Учреждение образования
«Белорусский государственный университет культуры и искусств»

Факультет культурологии и социально-культурной деятельности
Кафедра информационных технологий в культуре

СОГЛАСОВАНО
Заведующий кафедрой
_____/Т. С. Жилинская/
20.11. 2025 г.

СОГЛАСОВАНО
Декан факультета
_____/Н. Е. Шелупенко/
29.12. 2025 г.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

ОСНОВЫ ФРОНТЕНД РАЗРАБОТКИ

для специальности
6-05-0314-03 Социально-культурный менеджмент и коммуникации,
профилизации: Мультимедийные технологии и цифровые коммуникации

Составители:
Т.В. Бачурина, старший преподаватель кафедры информационных технологий в культуре учреждения образования «Белорусский государственный университет культуры и искусств»
О.М. Кунцевич, старший преподаватель кафедры информационных технологий в культуре учреждения образования «Белорусский государственный университет культуры и искусств»
С.А. Шатько, преподаватель кафедры информационных технологий в культуре учреждения образования «Белорусский государственный университет культуры и искусств»

Рассмотрено и утверждено на заседании
Совета факультета культурологии и социально-культурной деятельности
«29» 12 2025 г., протокол № 4

Рецензенты:
Кафедра программного обеспечения информационных систем и технологий учреждения образования «Белорусский национальный технический университет»;

Фролова Н.Ю., доцент кафедры коммуникативного дизайна учреждения образования «Белорусский государственный университет», кандидат культурологии, доцент

Рассмотрено и обсуждено на заседании кафедры информационных технологий в культуре
(протокол от 20.11.2025 № 3)

СОДЕРЖАНИЕ

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

В условиях современного общества неотъемлемым качеством квалифицированного специалиста сферы культуры и искусства становится высокий уровень владения интернет-технологиями для решения профессиональных задач по использованию, разработке, сопровождению и продвижению интернет-ресурсов в гипермедийном пространстве глобальной сети. Это предполагает знание основных технологий разработки сайтов, методов и инструментов их поискового и социального продвижения, умение применять их интегрировано для решения задач маркетинга и менеджмента.

Изучение учебной дисциплины «Основы фронтенд разработки» основывается на знаниях и умениях, полученных студентами в процессе освоения таких учебных дисциплин, как «Основы информационных технологий», «Информационная культура специалиста», «Системный анализ и моделирование информационных процессов», а также «Языки и системы программирования», «Технологии компьютерной графики», «Технологии создания баз данных сферы культуры», «Медиакультура специалиста».

Целью учебно-методического комплекса по учебной дисциплине «Основы фронтенд разработки» является обеспечение студентов учебно-методическими материалами для изучения учебной дисциплины, теоретическими знаниями и умениями в области веб-дизайна, навыками профессионального использования программных и технических средств веб-дизайна при создании и сопровождении сайтов в области культуры и искусств, что способствует усвоению в полном объеме учебного материала дисциплины через систематизацию, планирование и контроль собственной деятельности, дать рекомендации по подготовке к текущей и итоговой аттестации.

Учебно-методический комплекс составлен в соответствии с образовательным стандартом общего высшего образования по специальности 6-05-0314-03 Социально-культурный менеджмент и коммуникации и учебным планом учреждения высшего образования по специальности 6-05-0314-03 Социально-культурный менеджмент и коммуникации, профилизации «Мультимедийные технологии и цифровые коммуникации».

Учебно-методический комплекс включает следующие разделы: пояснительную записку, теоретический, практический, контроля знаний, вспомогательный. Теоретический раздел учебно-методического комплекса содержит материалы лекций. Практический раздел содержит задания лабораторных и практических занятий и рекомендации по их выполнению. Раздел контроля знаний включает в себя перечень требований к аттестации, критерии оценки результатов учебной деятельности, задания для контролируемой самостоятельной работы студентов, контрольные вопросы по темам, перечень вопросов к экзамену. Вспомогательный раздел учебно-методического комплекса содержит учебную программу, учебно-методические каты для дневной и заочной формы обучения, список основной и дополнительной литературы.

2. ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

2.1 Лекция 1. Введение. Технологии веб-дизайна

Основные вопросы:

1. Понятие веб-пространства Интернет.
2. Классификация интернет-ресурсов, критерии оценки.
3. Понятие веб-клиента и веб-сервера.
4. Технологические средства веб: URL-адресация, HTTP-протокол в контексте единой модели клиент-сервер.
5. Клиентские и серверные технологии веб-дизайна.
6. Этапы анализа и проектирования сайта.

Цель. Ознакомление с основными технологиями веб-дизайна. Интернет – всемирная система объединённых компьютерных сетей, построенная на базе протокола TCP/IP.

Интернет образует глобальное информационное пространство, служит физической основой для Всемирной паутины (World Wide Web, WWW) и множества других систем передачи данных.

День Интернета – 30 сентября.

3 июня 2011 года была принята резолюция ООН, признающая доступ в Интернет базовым правом человека. Отключение конкретных регионов от Интернета с июня 2011 года считается нарушением прав человека.

Ключевые принципы Интернета.

Интернет состоит из многих тысяч корпоративных, научных, правительственных и домашних компьютерных сетей. Объединение сетей разной архитектуры и топологии стало возможно благодаря протоколу IP (англ. Internet Protocol) и принципу маршрутизации пакетов данных.

Протокол IP был специально создан агностическим в отношении физических каналов связи. То есть любая система (сеть) передачи цифровых данных, проводная или беспроводная, для которой существует стандарт инкапсуляции в неё IP-пакетов, может передавать и трафик Интернета. Агностицизм протокола IP, в частности, означает, что компьютер или маршрутизатор должен знать тип сетей, к которым он непосредственно присоединён, и уметь работать с этими сетями; но не обязан (и в большинстве случаев не может) знать, какие сети находятся за маршрутизаторами.

На стыках сетей специальные маршрутизаторы (программные или аппаратные) занимаются автоматической сортировкой и перенаправлением пакетов данных, исходя из IP-адресов получателей этих пакетов. Протокол IP образует единое адресное пространство в масштабах всего мира, но в каждой отдельной сети может существовать и собственное адресное подпространство, которое выбирается исходя из класса сети. Такая организация IP-адресов позволяет маршрутизаторам однозначно определять дальнейшее направление для каждого пакета данных. В результате между

отдельными сетями Интернета не возникает конфликтов, и данные беспрепятственно и точно передаются из сети в сеть по всей планете и ближнему космосу.

Серф Винт и Роберт Кан – создатели протокола передачи данных TCP/IP, который по сей день является стандартом для передачи данных в Интернете.

Разница между понятиями Интернет и Веб (www).

Интернет: сеть компьютеров, объединенных каналами связи и использующих протоколы TCP/IP для связи.

Веб: сеть сайтов, использующих гиперссылки для переходов от страницы к странице.

*Веб-сайт (*от англ. *website*: *web* – «паутина», «сеть» и *site* – «место», буквально «место в сети») или просто сайт – в компьютерной сети объединённая под одним адресом (доменным именем или IP-адресом) совокупность документов частного лица или организации.

Второе определение веб-сайта подчеркивает его коммуникативную функцию.

Сайт – это набор информационных блоков и инструментов для взаимодействия с целевой аудиторией, которая может быть представлена реальными и потенциальными клиентами и партнерами, а также представителями средств массовой информации.

То, какая информация и как будет представлена на сайте, техническое оформление сайта находится в сильной зависимости от того, кто является целевой аудиторией и что сайт должен до нее донести и какие возможности предоставить.

Профессионально разработанный веб-сайт может служить как высокоэффективным инструментом ведения бизнеса, так и информационным или имиджевым ресурсом, рассказывающим о деятельности какой-либо общественной организации.

Задачи, которые решают сайты, многообразны – от организации внутренних или внешних коммуникаций компании (корпоративные ресурсы) и интернет-торговли (интернет-магазины) до самовыражения (сетевые дневники) и объединения людей по интересам (комьюнитиобразующие ресурсы).

Сайт – это структурированная информационная единица всемирной паутины.

Классификация сайтов.

Виды сайтов по технологиям, влияющим на дизайн и функциональность:

- статические сайты и веб-страницы;
- динамические сайты и веб-страницы;
- флэш-сайты.

По принадлежности сайты подразделяются на:

- личные (персональные) сайты;
- сайты коммерческих организаций;

- сайты некоммерческих организаций.

К официальному сайту предъявляются более жесткие требования в части информационного содержимого, графического дизайна, навигации, хостинга.

Официальный сайт обычно имеет следующие разделы:

- новостная информация;
- нормативные документы, положения;
- направления деятельности;
- структура учреждения;
- кадровый состав;
- контактная информация (список ответственных лиц, их должности, координаты и часы приема).

Виды сайтов *по величине*, по уровню решаемых ими задач:

- простые сайты, содержащие немного информации и состоящие из нескольких страничек («сайты-визитки», домашние странички и т.п.);
- тематические, узконаправленные сайты;
- многофункциональные сайты (порталы).

Виды сайтов *по назначению*:

- сайты, предоставляющие контент;
- сайты для онлайн-контактов и общения;
- сайты электронной коммерции;
- сайты, предоставляющие онлайн-сервисы.

По характеру предоставляемого контента можно выделить:

- информационно-тематические сайты;
- новостные сайты;
- развлекательные сайты;
- сайты-библиотеки;
- сайты-базы определённого рода документов;
- разнообразные сайты-справочники, онлайн-энциклопедии и словари,
- сайты-каталоги, обобщающие информацию о других сайтах и т.д.

По доступности сайты бывают:

- открытые – открыты для всех посетителей;
- полуоткрытые – часть информации открыта для всех, а часть скрыта, поэтому для просмотра скрытой информации на таком сайте необходима регистрация;
- закрытые.

http://archive.org/web/ - веб-архив сайтов с момента появления Веба.

Способы активного отображения информации во Всемирной паутине.

Информация в вебе может отображаться как пассивно (то есть пользователь может только считывать её), так и активно – тогда пользователь может добавлять информацию и редактировать её. К способам активного отображения информации относятся:

– гостевые книги,

– форумы,

– чаты,

– блоги,
– вики-проекты,
– социальные сети,
– системы управления контентом.

Перспективы развития Всемирной паутины.

В настоящее время наметились две тенденции в развитии Всемирной паутины: семантическая паутина и социальная паутина. Семантическая паутина предполагает улучшение связности и релевантности информации во Всемирной паутине через введение новых форматов метаданных. Социальная паутина полагается на работу по упорядочиванию имеющейся в Паутине информации, выполняемую самими пользователями Паутины. В рамках второго направления наработки, являющиеся частью семантической паутины, активно используются в качестве инструментов (RSS и другие форматы веб-каналов, микроформаты XHTML).

12 критериев качества ресурсов: прозрачность, эффективность, поддержка, доступность, ориентация на пользователя, реактивность, многоязычность, совместимость, управляемость, сохранность, производительность, посещаемость.

Персоналии веб-дизайна.

Сэр Тимоти Джон Бернерс-Ли (англ. Sir Timothy John «Tim» Berners-Lee; род. 8 июня 1955) – британский учёный, изобретатель URI, URL, HTTP, HTML, изобретатель Всемирной паутины (совместно с Робертом Кайо) и действующий глава Консорциума Всемирной паутины. Автор концепции семантической паутины. Автор множества других разработок в области информационных технологий.

Роберт Кайо (англ. Robert Cailliau) (родился 26 января 1947) совместно с сэром Тимом Бернерсом-Ли изобрёл технологию Всемирной паутины (World Wide Web). Место рождения: Тонгерен, Бельгия. В 1989 году независимо от Тима Бернерса-Ли предложил систему гипертекст для доступа к документации CERN. В 1990 году это привело к совместному предложению этой технологии, а затем и к созданию Всемирной паутины (World Wide Web).

17 мая 1991 более 20 лет назад сотрудник Европейской лаборатории по ядерным исследованиям CERN в Женеве, консультант по программному обеспечению Тим Бернес-Ли приклеил на один из компьютеров в своей лаборатории наклейку с надписью «This machine is a server, DO NOT POWER IT DOWN!!”. Этот первый в мире веб-сервер реализовал „большую технологическую тройку“: URL-адресацию, HTML-разметку и HTTP-протокол в контексте единой модели клиент-сервер.

Первый в мире веб-сайт появился в Интернете 6 августа 1991 года по адресу http://info.cern.ch/.

На этом сайте описывалось, что такое Всемирная паутина, как установить веб-сервер, как заполучить браузер и т. п. Этот сайт также являлся первым в мире интернет-каталогом, потому что позже Тим Бернерс-Ли разместил и поддерживал там список ссылок на другие сайты.

W3C. 1994 г. – основную работу по развитию Всемирной паутины взял на себя Консорциум Всемирной паутины (англ. World Wide Web Consortium, W3C), основанный и до сих пор возглавляемый Тимом Бернерсом-Ли. Консорциум разрабатывает и внедряет единые принципы и технологические стандарты для Интернет и Всемирной паутины (называемые «Рекомендáциями», англ. W3C Recommendations).

Миссия W3C: «Полностью раскрыть потенциал Всемирной паутины, путём создания протоколов и принципов, гарантирующих долгосрочное развитие Сети». Две другие важнейшие задачи Консорциума –обеспечить полную «интернационализáцию Сетй» и сделать Сеть доступной для людей с ограниченными возможностями.

Все Рекомендации Консорциума Всемирной паутины открыты, то есть не защищены патентами и могут внедряться любым человеком без всяких финансовых отчислений консорциуму.

Официальный сайт Консорциума Всемирной паутины (W3C) www.w3.org

О нескольких важных принципах:

Возможность редактировать информацию Паутины не менее важна, чем возможность просто путешествовать по ней. В этом смысле Бернерс-Ли очень рассчитывает на концепцию WYSIWYG, хотя Wiki –это тоже шаг в нужном направлении.

Компьютеры могут быть использованы для «фоновых процессов», помогающих людям работать сообща.

Каждый аспект Интернета должен работать как паутина, а не как иерархия. В этом смысле очень неприятным исключением является система имён доменов (англ. Domain Name System, DNS), управляемая организацией ICANN.

Учёные-компьютерщики несут не только техническую ответственность, но и моральную.

Развитие веб-дизайна шло по двум основным направлениям:

– с развитием *программных* и *инструментальных* средств:

1. HTML-дизайн; использование в дизайне таблиц для разметки;
2. CSS (каскадные таблицы стилей);
3. Flash (динамичекий дизайн);
4. CMS (система управления контентом);
5. HTML5, CSS3, Ajax (Asynchronous Javascript and XML).

– с развитием *технических* средств:

1. совершенствование средств вывода информации (мониторы, их разрешение и цветопередача);
2. увеличение пропускной способности каналов связи.

Браузеры. Текстовые документы, содержащие код на языке HTML (такие документы традиционно имеют расширение .html или .htm), обрабатываются специальными приложениями, которые отображают документ в его форматированном виде. Такие приложения, называемые браузерами, обычно предоставляют пользователю удобный интерфейс для

запроса веб-страниц, их просмотра (и вывода на иные внешние устройства) и, при необходимости, отправки введённых пользователем данных на сервер.

Веб-сервер –это сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, обычно вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными.

Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и непосредственно компьютер, на котором это программное обеспечение работает.

Клиент, которым обычно является веб-браузер, передаёт веб-серверу запросы на получение ресурсов, обозначенных URL-адресами. Ресурсы –это HTML-страницы, изображения, файлы, медиа-потоки или другие данные, которые необходимы клиенту. В ответ веб-сервер передаёт клиенту запрошенные данные. Этот обмен происходит по протоколу HTTP.

Наиболее распространённые веб-серверы:

– Apache – свободный веб-сервер, наиболее часто используемый в Unix-подобных операционных системах;

– IIS от компании Microsoft, распространяемый с ОС семейства Windows NT.

Технологии построения веб-приложений

– Клиентские средства разработки веб-приложений: HTML, CSS, JAVASCRIPT.

– Серверные технологии разработки веб-приложений: PERL, PHP, PYTHON, ASP и др.

Актуальные технологии при разработке веб-приложений: XHTML, микроформаты, фреймворки, AJAX, JSON, RSS, Ruby/Ruby on Rails, ASP.NET, Silverlight.

Технология AJAX (асинхронный JavaScript + XML) была известно давно. Однако, благодаря появлению термина AJAX, который ввел Джис Джеймс Гаррет (Jesse James Garrett) в феврале 2005г., она стала необычайно модной.

Суть технологии AJAX заключается в изменении содержимого загруженной веб-страницы без ее полной перезагрузки, благодаря чему достигается высокая динамичность сайтов. Технология основывается на разделении данных и подзагрузки тех или иных компонентов по мере необходимости.

AJAX технологией в строгом смысле слова не является. Это просто аббревиатура, обозначающая подход к созданию веб-приложений с помощью следующих технологий: стандартизированное представление средствами XHTML и CSS; динамическое отображение и взаимодействие с пользователем с помощью DOM; обмен и обработка данных в виде XML и XSLT; JavaScript; асинхронные запросы с помощью объекта XMLHttpRequest.

Преимущества:

– Экономия трафика. Вместо загрузки всей страницы достаточно загрузить только изменившуюся часть, как правило довольно небольшую.

– Уменьшение нагрузки на сервер. К примеру, на странице работы с почтой, когда вы отмечаете прочитанные письма, серверу достаточно внести изменения в базу данных и отправить клиентскому скрипту сообщение об успешном выполнении операции без необходимости повторно создавать страницу и передавать её клиенту.

– Ускорение реакции интерфейса. Поскольку нужно загрузить только изменившуюся часть, пользователь видит результат своих действий быстрее.

Недостатки:

– Отсутствие интеграции со стандартными инструментами браузера.

– Динамически создаваемые страницы не регистрируются браузером в истории посещения страниц, поэтому не работает кнопка «Назад», предоставляющая пользователям возможность вернуться к просмотренным ранее страницам, но существуют скрипты, которые могут решить эту проблему.

– Другой недостаток изменения содержимого страницы при постоянном URL заключается в невозможности сохранения закладки на желаемый материал. Частично решить эти проблемы можно с помощью динамического изменения идентификатора фрагмента (части URL после #), что позволяют многие браузеры.

– Динамически загружаемое содержимое недоступно поисковикам (если не проверять запрос, обычный он или XMLHttpRequest).

– Поисковые машины не могут выполнять JavaScript, поэтому разработчики должны позаботиться об альтернативных способах доступа к содержимому сайта.

– Старые методы учёта статистики сайтов становятся неактуальными. Многие сервисы статистики ведут учёт просмотров новых страниц сайта. Для сайтов, страницы которых широко используют AJAX, такая статистика теряет актуальность.

– Усложнение проекта. Перераспределяется логика обработки данных – происходит выделение и частичный перенос на сторону клиента процессов первичного форматирования данных. Это усложняет контроль целостности форматов и типов. Конечный эффект технологии может быть нивелирован необоснованным ростом затрат на кодинг и управление проектом, а также риском снижения доступности сервиса для конечных пользователей.

2.2 Лекция 2. Основы проектирования сайта

Основные вопросы

1. Этапы анализа сайта
2. Стратегии проектирования
3. Ограничения сайта, оценка их значимости
4. Этапы проектирования веб-сайта
5. Проектирование информационного наполнения
6. Концепция графического дизайна сайта
7. Принципы информационной архитектуры и usability веб-ресурсов
8. Онлайн-конструкторы сайтов
9. Законодательное регулирование белорусского сегмента сети интернет

Цель: дать студентам комплексное понимание процесса проектирования веб-ресурсов, от анализа и стратегий до правовых аспектов, и научить применять полученные знания на практике.

Этапы анализа сайта:

– определение назначения и критериев успешной разработки сайта;

– определение состава пользователей и способов использования ими сайта;

– определение и организация информационных тем;

– анализ ограничений и определение методов их преодоления;

– уточнение назначения, критериев работы и состава информации узла с учетом ограничений.

– анализ сайтов-«конкурентов».

Цели и задачи сайта. Главные и второстепенные цели. Критерии достижения цели (как оценивать степень успешной реализации поставленных целей).

Примеры задач:

– публикация информации об организации;

– расширение круга пользователей;

– предоставление информации о своих продуктах и услугах;

– обеспечение возможности пользователям присылать отзывы и предоставлять дополнительные сведения;

– прием заказов на продукты и услуги в оперативном режиме;

– в рамках интрасети (внутренней сети компании) предоставление доступа к административным распоряжениям, документации и переписке компании;

– в рамках экстрасети (внешней сети с ограниченным кругом пользователей) предоставление сторонним организациям доступа к части интрасети компании, например, для электронной коммерции.

Анализ состава пользователей сайта. Кто входит в состав пользователей сайта? Каковы потребности пользователей? Какая

информация требуется пользователям? Могут ли обращаться к сайту конкуренты?

Проанализировать каждую категорию пользователей по следующим критериям:

– интересы, потребности, навыки, способности и предпочтения;

– платформа, броузер, быстродействие коммутируемого соединения и степень подготовки и опыта работы в Интернет;

– описания платформ потенциальных пользователей, включая изготовителя оборудования, модели, оперативную память, жесткие дискии другое оборудование.

Определение информационной тематики. Организация информационных тем по категориям. Состояние разработки информационных тем.

Ограничения сайта, оценка их значимости:

– низкая пропускная способность;

– применение пользователями браузеров, предназначенных только для текста;

– недостаточная пропускная способность для применения графических и мультимедийных файлов:

– недостаточные знания разработчиков компании в области HTML и мультимедиа;

– большие затраты и трудоемкость разработки многоязычных версий.

Корректировка проекта веб-сайта с учетом ограничений.

Этапы проектирования веб-сайта.

– определение концептуальной модели сайта (описания назначения, темы и стиля);

– построение логической структуры сайта (блок-схемы навигации, на которой показана организация и пути навигации по сайту):

– проектирование средств навигации;

– проектирование информационного наполнения;

– построение или обновление прототипа (прототипов),

– разработка дизайна и програмирование сайта;

– тестирование сайта.

Стратегии проектирования – это совокупность подходов и методов, позволяющих эффективно разрабатывать и реализовывать проекты, учитывая внутренние ресурсы и внешние условия. Они включают аналитические инструменты (SWOT, PESTLE), школы стратегического мышления и этапы планирования, направленные на достижение устойчивого результата.

Стратегии проектирования занимают ключевое место в современном управлении и инженерной практике. Они позволяют системно подходить к разработке проектов, обеспечивая баланс между целями, ресурсами и внешними условиями. В основе лежит идея: правильная стратегия проектирования определяет успех реализации проекта.

Основные подходы к стратегическому проектированию.

Аналитические методы:

– *SWOT-анализ* – выявление сильных и слабых сторон, возможностей и угроз. Сильные стороны – простота, ясность, структурность. К ограничению применения можно отнести – субъективность оценок. Данный метод применяют на начальном этапе проектирования.

– *PESTLE-анализ* – оценка политических, экономических, социальных, технологических, правовых и экологических факторов. Сильные стороны – учет макросреды, комплексность. К ограничению применения можно отнести – необходимость использования большого объема данных. Данный метод применяют для долгосрочных проектов.

– Этапы *Strategium* – Сильные стороны – последовательность, интеграция в управление. К ограничению применения можно отнести высокую трудоемкость. Данный метод применяют для комплексных проектов.

Эти инструменты помогают формировать основу для выбора стратегического направления.

Этапы разработки стратегии:

– Анализ внешней и внутренней среды.

– Формулирование миссии и видения.

– Построение бизнес-модели.

– Определение стратегических целей.

– Разработка проектов и распределение ресурсов.

– Мониторинг и корректировка реализации.

Практическое значение данных методов:

Для бизнеса: стратегии проектирования помогают адаптироваться к изменениям рынка и конкуренции.

Для инженерии: обеспечивают рациональное использование ресурсов и минимизацию рисков.

Для образования: формируют у студентов системное мышление и навыки стратегического анализа.

Стратегии проектирования веб-ресурсов – это системный подход к созданию сайтов, включающий анализ целей и аудитории, выбор технологий, разработку дизайна и обеспечение удобства и безопасности. Они помогают превратить веб-ресурс из набора страниц в эффективный инструмент бизнеса и коммуникации. Веб-ресурс сегодня – это не просто сайт, а полноценная платформа для взаимодействия с клиентами, партнёрами и обществом. Стратегия проектирования определяет, каким образом ресурс будет выполнять свои функции: информировать, продавать, обучать или продвигать бренд.

Основные стратегии проектирования

1. Определение целей и задач.

Первостепенная задача при разработке веб-сайта – это четкое определение его целей и задач. Этот этап является ключевым для создания эффективного ресурса, который будет отвечать всем требованиям заказчика и потребностям его аудитории. Необходимо понимать, какие задачи должен

решать сайт: увеличение продаж, информирование клиентов, сбор контактных данных потенциальных клиентов или же укрепление имиджа компании. От того, насколько точно будут определены эти аспекты, зависит успешность всего проекта. Без четкого понимания целей, процесс разработки может превратиться в бесконечный цикл корректировок и изменений, что неизбежно приведет к увеличению сроков и бюджета проекта.

Пример: интернет-магазин –цель: продажи; образовательный портал – цель: обучение.

2. Анализ целевой аудитории и конкурентов.

Определение и анализ целевой аудитории являются ключевыми аспектами в процессе проектирования веб-сайта. Это позволяет создать продукт, который максимально отвечает потребностям и ожиданиям пользователей. Для эффективного анализа необходимо изучить следующие параметры:

– демографические характеристики (возраст, пол, доход и т.д.);

– поведенческие факторы (интересы, увлечения, привычки в интернете);

– потребности и проблемы, которые ваш сайт может решить.

Анализ конкурентов также играет важную роль в разработке успешного веб-сайта. Он помогает определить сильные и слабые стороны конкурирующих сайтов, выявить незанятые ниши и предложить пользователям уникальное ценностное предложение. Важно сосредоточиться на следующих аспектах:

– дизайн и удобство использования сайтов конкурентов;

– контент и способы взаимодействия с аудиторией;

– маркетинговые стратегии и каналы привлечения трафика.

Эти данные помогут вам выстроить эффективную стратегию развития веб-сайта и обеспечить его конкурентное преимущество на рынке.

Пример: сайт для подростков – яркий дизайн, мобильная адаптация; сайт для госуслуг – простота и надёжность.

3. Разработка структуры и прототипирование сайта

Эффективность веб-сайта напрямую зависит от его структуры и удобства использования. Прототипирование – ключевой этап, позволяющий визуализировать будущий продукт и тестировать его функциональность до начала разработки. Сравнение различных инструментов для прототипирования, таких как Adobe XD, Sketch и Figma, показывает их уникальные возможности и области применения. Например, Adobe XD идеально подходит для сложных проектов с динамическими элементами, Sketch – для работы с векторной графикой, а Figma выделяется возможностью коллективной работы в реальном времени. Выбор инструмента зависит от специфики проекта и предпочтений команды разработчиков.

Инструмент	Особенности	Преимущества
Adobe XD	Динамические элементы, интеграция с другими	Мощные инструменты для создания интерактивных

	продуктами Adobe	прототипов
Sketch	Работа с векторной графикой, плагины для расширения функционала	Идеален для дизайна интерфейсов и векторной графики
Figma	Коллективная работа в реальном времени, обширная библиотека компонентов	Упрощает командную работу и обеспечивает высокую скорость проектирования

4. Выбор технологий и платформы для разработки

Выбор подходящих технологий и платформы является ключевым моментом в процессе разработки веб-сайта. Это решение влияет на скорость разработки, безопасность, масштабируемость (использование CMS (WordPress, Joomla, Drupal) для удобного управления контентом) и возможность интеграции с другими сервисами (интеграция CRM, платёжных систем и сервисов для расширения функционала).

Например, для создания динамических веб-приложений часто используются JavaScript-фреймворки, такие как React или Angular, в то время как для более простых сайтов может подойти система управления контентом (CMS) вроде WordPress или Joomla. Важно учитывать специфику проекта и потребности целевой аудитории при выборе технологий.

Пример: WordPress – для малого бизнеса, Drupal – для крупных порталов.

Технология/ Платформа	Простота использования	Гибкость	Стоимость	Поддержка сообщества
WordPress	Высокая	Средняя	Бесплатно (за исключением стоимости хостинга и домена)	Очень высокая
React	Средняя	Высокая	Бесплатно	Высокая
Angular	Средняя	Высокая	Бесплатно	Высокая
Joomla	Высокая	Средняя	Бесплатно (за исключением стоимости хостинга и домена)	Высокая

В таблице учтены такие параметры, как простота использования, гибкость, стоимость и поддержка сообщества. Например, WordPress идеально подходит для блогов и корпоративных сайтов благодаря своей простоте и большому выбору готовых тем и плагинов. В то же время, для разработки сложных веб-приложений с высокой производительностью и уникальным пользовательским интерфейсом лучше использовать React или Angular.

4. Дизайн и UX.

Дизайн веб-сайта: ключевые принципы и тренды

Создание привлекательного и функционального дизайна веб-сайта требует глубокого понимания текущих трендов и основных принципов визуального оформления. Удобство использования (usability) и адаптивность являются неотъемлемыми аспектами, обеспечивающими высокую вовлеченность пользователей и их удовлетворенность. Современные тенденции, такие как минимализм, использование кастомных иллюстраций и анимаций, а также внедрение темной темы, значительно повышают эстетическую привлекательность и удобство веб-сайтов. Особое внимание стоит уделить выбору цветовой палитры и типографики, поскольку эти элементы в значительной степени влияют на восприятие бренда пользователями. Важно помнить, что каждый элемент дизайна должен служить определенной цели и способствовать достижению общих бизнес-целей веб-сайта.

Адаптивная верстка и мобильная оптимизация

С учетом того, что большинство пользователей сегодня используют мобильные устройства для доступа к интернету, адаптивная верстка и мобильная оптимизация становятся ключевыми аспектами в проектировании веб-сайтов. Эти подходы обеспечивают корректное отображение сайта на различных устройствах, улучшая пользовательский опыт и повышая общую доступность контента. Среди преимуществ адаптивной верстки можно выделить улучшенную скорость загрузки страниц на мобильных устройствах и повышение позиций сайта в поисковой выдаче, поскольку поисковые системы, такие как Google, предпочитают оптимизированные под мобильные устройства сайты. Однако, стоит учитывать и некоторые сложности: разработка адаптивного дизайна требует дополнительных ресурсов и времени, а также более тщательного тестирования на разных устройствах, что может увеличить общую стоимость проекта. Несмотря на эти недостатки, инвестиции в адаптивность и мобильную оптимизацию окупаются за счет увеличения удовлетворенности пользователей и конверсии.

Удобство и адаптивность – ключ к удержанию пользователей: адаптивный дизайн позволяет одинаково комфортно пользоваться сайтом на смартфоне и ПК. Прототипирование интерфейса, удобная навигация, акцент на пользовательский опыт.

5. SEO и оптимизация.

Сайт должен быть видимым в поисковых системах: оптимизация скорости загрузки повышает позиции в Google; включение ключевых слов, настройка мета-тегов, оптимизация скорости загрузки; подготовка ресурса к продвижению в поисковых системах.

Разработка контента и его SEO-оптимизация

Создание качественного контента является ключевым аспектом в процессе разработки веб-сайта. Он должен быть не только информативным и интересным для целевой аудитории, но и оптимизирован для поисковых систем. SEO-оптимизация играет важную роль в продвижении сайта в топ выдачи, что напрямую влияет на его посещаемость и популярность. В процессе создания контента важно соблюдать следующие принципы:

– Подбор ключевых слов: Использование актуальных и часто искомых запросов помогает улучшить видимость страниц.

– Уникальность материала: Контент должен быть оригинальным и полезным, что повышает его ценность для пользователей и поисковых систем.

– Структурированность текста: Применение заголовков, подзаголовков и списков улучшает читабельность и восприятие информации.

– Адаптация под мобильные устройства: С учетом растущей популярности смартфонов, контент должен корректно отображаться на всех типах устройств.

Эти элементы способствуют не только улучшению пользовательского опыта, но и повышению ранжирования сайта в поисковых системах.

6. Тестирование и запуск.

Проверка ресурса на разных устройствах и браузерах, проверка на баги, ошибки, удобство использования; постоянный мониторинг и улучшение после запуска.

Пример: тестирование интернет-магазина на корректность работы корзины и оплаты.

Тестирование и отладка функционала сайта

На этапе тестирования и отладки функционала сайта особое внимание уделяется обеспечению его стабильной и корректной работы на всех типах устройств и в различных браузерах. Этот процесс включает в себя не только проверку выполнения запланированных функций, но и выявление потенциальных уязвимостей. Основные направления тестирования включают:

– Функциональное тестирование – проверка работы всех предусмотренных функций сайта.

– Тестирование совместимости – убеждаемся, что сайт корректно отображается и работает в различных браузерах и на разных устройствах.

– Тестирование производительности – оценка скорости загрузки страниц и обработки запросов.

– Тестирование безопасности – выявление уязвимостей, которые могут быть использованы для атак на сайт.

После завершения всех тестов и исправления найденных ошибок, важно провести регрессионное тестирование, чтобы убедиться, что внесенные изменения не повлияли на уже существующий функционал. Этот этап критически важен для обеспечения качества веб-сайта перед его запуском. Также не стоит забывать о пользовательском тестировании, которое поможет понять, насколько удобен сайт для конечного пользователя. Включение реальных пользователей в процесс тестирования может выявить проблемы, не замеченные разработчиками, и помочь сделать сайт максимально удобным и функциональным.

Запуск сайта и анализ его эффективности

Запуск сайта – это лишь начало пути в его продвижении и развитии. Важно не только успешно разместить ресурс в сети, но и обеспечить его

постоянный анализ и оптимизацию. Для этого используются различные инструменты и методики, позволяющие оценить, насколько хорошо сайт соответствует целям бизнеса и нуждам его целевой аудитории. Среди ключевых показателей эффективности сайта:

– Поведенческие факторы – время на сайте, глубина просмотра, отказы;

– Конверсии – количество совершенных целевых действий по отношению к общему числу посетителей;

– Позиции в поисковой выдаче по ключевым запросам, связанным с бизнесом.

После запуска сайта крайне важно уделить внимание сбору и анализу обратной связи от пользователей. Это позволит выявить потенциальные проблемы в удобстве использования сайта, скорости его работы и других аспектах, которые могут отталкивать посетителей. Регулярное обновление контента, улучшение интерфейса и технической части сайта в соответствии с полученными данными способствуют повышению его общей эффективности.

Использование инструментов веб-аналитики, таких как Google Analytics, Яндекс.Метрика, помогает в реализации стратегии постоянного улучшения сайта. Они предоставляют подробную информацию о том, как посетители взаимодействуют с сайтом, что в свою очередь позволяет оптимизировать его структуру, контент и функциональность для достижения лучших результатов. Тестирование различных версий страниц (A/B тестирование) также является эффективным способом улучшения конверсии и удовлетворенности пользователей.

Сравнительная таблица стратегий

Стратегия	Преимущества	Ограничения	Применение
CMS-платформа	Простота управления, гибкость	Ограничения по кастомизации	Малый и средний бизнес
Адаптивный дизайн	Удобство на всех устройствах	Требует дополнительных ресурсов	Все типы сайтов
SEO-ориентированная	Высокая видимость в поиске	Долгосрочный эффект	Коммерческие проекты
UX-прототипирование	Повышение конверсии	Высокая трудоёмкость	Интернет-магазины, сервисы

Практическое значение:

– для бизнеса: рост продаж и узнаваемости бренда.

– для образования: удобные платформы для дистанционного обучения.

– для государства: прозрачные и доступные сервисы для граждан.

Стратегии проектирования веб-ресурсов – это комплексный процесс, включающий исследование, планирование, дизайн, оптимизацию и поддержку. Их грамотное применение превращает сайт в эффективный

инструмент, способный адаптироваться к изменениям рынка и потребностям пользователей.

Типы контента сайта, выбор информационных шаблонов и концепция графического дизайна

Современные веб-ресурсы – это не просто набор страниц, а комплексные системы коммуникации, маркетинга и обслуживания пользователей. Их эффективность напрямую зависит от того, какой контент используется, какие шаблоны выбраны и как реализован графический дизайн.

Правила хорошего веб-дизайна:

1. Привлекает. Цель любого сайта - получить реакцию пользователя. Это значит, что основная задача сайта - показать и рассказать о Вашем товаре или услуге, продемонстрировать самые ценные качества, удовлетворить потребности клиента в Вашей продукции и вызвать желание задержаться на сайте. А также сделать сайт таким, чтобы его образ, цветовая гамма или отдельные элементы были запоминающимися

2. Делает сайт понятным. Удобство взаимодействия клиента с сайтом зависит от его логично выстроенной структуры и продуманного дизайна. Понятная навигация, графические элементы, которые не напрягают глаза, приятные сочетания цветов - не только повышают лояльность пользователя к компании, но и повышают поведенческие факторы, такие как время на сайте, глубина просмотра, процент отказов и конверсию

3. Не кричит. Сайт - это один из инструментов маркетинга. И в отличии от эпатажного искусства, которое должно шокировать и вызывать бурю впечатлений, сайт должен вызывать только положительные эмоции. Главная функция сайта - быть понятным целевой аудитории через простые, сбалансированные решения, дополненные визуальным оформлением. Сайт должен продавать, а не провоцировать

4. Продает. Казалось бы, сайт - это всего лишь сочетание текста и графических элементов. Однако, правильное расположение их и есть ключ к успеху. Дизайнер способен выстроить структуру сайта таким образом, чтобы пользователь прошел определенный путь и в конечном итоге совершил целевое действие на сайте: позвонил, отправил форму обратной связи, купил товар или написал в чат-бот и т.д.

Типы контента сайта:

– Текстовый контент: Статьи, новости, описания товаров и услуг. Обеспечивает информирование и продвижение. Пример: корпоративный блог компании.

– Мультимедийный контент: Фото, видео, аудио. Повышает вовлечённость и делает сайт более привлекательным. Пример: видеопрезентации на образовательных порталах.

– Интерактивный контент: Формы обратной связи, калькуляторы, опросы. Способствует взаимодействию с пользователем. Пример: онлайн-заявка на сайте банка.

– Динамический контент: Автоматически обновляемые данные: ленты новостей, базы данных, погодные сервисы. Пример: новостные порталы.

Выбор информационных шаблонов сайта

Шаблон – это структура, определяющая расположение элементов и логику взаимодействия.

– Статический шаблон: фиксированная структура страниц, подходит для визиток и небольших сайтов.

– Динамический шаблон: гибкая структура, интеграция с CMS, позволяет легко обновлять и расширять сайт.

– Модульный шаблон: состоит из блоков (меню, баннеры, карточки). Удобен для интернет-магазинов и порталов.

Пример: интернет-магазин использует модульный шаблон с каталогом, корзиной и системой отзывов.

Концепция графического дизайна сайта

Графический дизайн формирует визуальное восприятие ресурса и влияет на доверие пользователей.

Принципы дизайна:

– единый стиль и фирменная айдентика;

– читаемость и визуальная иерархия;

– адаптивность для разных устройств.

Элементы дизайна:

– цветовая палитра;

– типографика;

– иконки, изображения.

Пример: образовательный портал использует спокойные цвета и минималистичный дизайн для удобства чтения, а коммерческий сайт – яркие акценты для привлечения внимания к товарам.

Веб-ресурсы Беларуси демонстрируют разные подходы к контенту, шаблонам и дизайну: от крупных порталов (Onliner.by) до специализированных сервисов (Relax.by, Citydog.by). Их примеры позволяют показать, как типы контента, выбор шаблонов и концепция графического дизайна влияют на эффективность сайта.

Современные белорусские веб-ресурсы – это не только источники информации, но и полноценные платформы для бизнеса, досуга и коммуникации. Рассмотрим три ключевых аспекта проектирования сайтов: контент, шаблоны и графический дизайн, подкрепив их примерами популярных ресурсов Беларуси.

1. Типы контента сайта

Текстовый контент – статьи, новости, описания: Belta.by, новостной портал, где текстовые материалы составляют основу.

Мультимедийный контент – фото, видео, аудио: Onliner.by активно использует фотообзоры и видеоконтент для обзоров техники и автомобилей.

Интерактивный контент – формы, калькуляторы, опросы: Relax.by предлагает сервисы бронирования ресторанов и доставки еды, что делает сайт интерактивным.

Динамический контент – постоянно обновляемые данные: Pogoda.by (Белгидромет) публикует прогнозы погоды и метеоновости, обновляемые ежедневно.

2. Выбор информационных шаблонов сайта

Статический шаблон – простая структура, редко обновляемый контент: сайт Argo Minsk критикуется за устаревший статический шаблон и хаотичное оформление.

Динамический шаблон – гибкая структура, интеграция с CMS: Citydog.by как городской журнал использует динамические шаблоны для публикации статей и фотоисторий.

Модульный шаблон – набор блоков (каталог, корзина, отзывы): Onliner.by сочетает новостной портал и маркетплейс, где модульная структура позволяет совмещать статьи, объявления и каталог товаров.

3. Концепция графического дизайна сайта

Принципы: единый стиль, читаемость, адаптивность. Успешные примеры:

Silverweb.by демонстрирует современный UX/UI-дизайн, где акцент сделан на удобство и визуальную гармонию.

B-R.by (Brand Room) создаёт уникальные сайты без шаблонов, уделяя внимание фирменному стилю и айдентике.

Неудачные примеры:

Pogoda.by критикуется за перегруженность текстом и слабую графическую структуру.

Argo Minsk – пример устаревшего дизайна с хаотичной цветовой гаммой.

Эффективный веб-ресурс строится на трёх взаимосвязанных основах: контент, шаблон, графический дизайн. Грамотное сочетание этих элементов превращает сайт в мощный инструмент коммуникации, маркетинга и образования. Белорусские сайты показывают, что успех веб-ресурса зависит от баланса контента, шаблона и дизайна. Современные проекты (Onliner.by, Silverweb.by, B-R.by) используют динамические и модульные шаблоны, мультимедийный контент и продуманный UX/UI. Устаревшие ресурсы (Argo Minsk, Pogoda.by) демонстрируют, как игнорирование этих принципов снижает эффективность.

Онлайн-конструкторы сайтов и система доменных имен

Современные технологии позволяют создавать веб-ресурсы не только профессиональным разработчикам, но и пользователям без специальных знаний. Важную роль в этом играют онлайн-конструкторы сайтов, а также понимание работы системы доменных имен (DNS).

Онлайн-конструкторы сайтов – это сервисы, позволяющие создавать веб-страницы без знания языков программирования. Основаны на принципе drag-and-drop: пользователь перетаскивает готовые блоки (текст, изображения, формы).

Преимущества: простота, скорость, доступность.

Ограничения: меньшая гибкость по сравнению с ручной разработкой, зависимость от платформы.

Создание веб-сайтов с помощью онлайн-конструкторов

Этапы работы:

– регистрация на платформе;

– выбор шаблона (визитка, интернет-магазин, блог);

– настройка дизайна: цвета, шрифты, структура;

– добавление контента: тексты, фото, видео;

– подключение домена и публикация;

– SEO-настройка и интеграция с соцсетями.

Пример: создание интернет-магазина на Shopify или Tilda занимает несколько часов и не требует знаний HTML/CSS.

Обзор популярных конструкторов сайтов:

– Wix – удобный интерфейс, множество шаблонов, интеграция с маркетингом.

– Tilda Publishing – популярна в Беларуси и СНГ, акцент на дизайн и адаптивность.

– Squarespace – сильная визуальная составляющая, подходит для портфолио и креативных проектов.

– Shopify – специализированный конструктор для интернет-магазинов.

– WordPress.com – сочетает простоту конструктора и возможности CMS.

Система доменных имен (DNS- Domain Name System*)* – система, которая переводит удобные для человека имена сайтов (например, example.com) в IP-адреса, понятные компьютерам, обеспечивает доступность сайтов в интернете, упрощает навигацию.

Структура доменного имени:

– корневой домен («.»);

– домен верхнего уровня (TLD) – .by, .com, .org.;

– домен второго уровня – название сайта (onliner.by);

– поддомены – дополнительные разделы (shop.onliner.by).

Пример: сайт belta.by использует национальный домен .by, что подчёркивает его принадлежность к Беларуси.

Онлайн-конструкторы сайтов сделали процесс создания веб-ресурсов доступным для широкой аудитории. Они позволяют быстро запускать проекты, но имеют ограничения по функциональности. Система доменных имен обеспечивает удобство и глобальную доступность сайтов. В совокупности эти инструменты формируют основу современного веб-пространства.

Законодательное регулирование белорусского сегмента сети интернет:

ЗАКОН РЕСПУБЛИКИ БЕЛАРУСЬ 10 ноября 2008 г. № 455-З Об информации, информатизации и защите информации

Источник: https://pravo.by/document/?guid=3871&p0=h10800455 – Национальный правовой Интернет-портал Республики Беларусь

УКАЗ ПРЕЗИДЕНТА РЕСПУБЛИКИ БЕЛАРУСЬ 1 февраля 2010 г. № 60 О мерах по совершенствованию использования национального сегмента сети Интернет

Источник: https://pravo.by/document/?guid=3871&p0=p31000060 – Национальный правовой Интернет-портал Республики Беларусь

ГОСУДАРСТВЕННАЯ ПРОГРАММА "ЭЛЕКТРОННАЯ БЕЛАРУСЬ"

УТВЕРЖДЕНО Постановление Совета Министров Республики Беларусь 27.12.2002 № 1819 ГОСУДАРСТВЕННАЯ ПРОГРАММА информатизации Республики Беларусь на 2003-2005гг. и на перспективу до 2010 года «Электронная Беларусь»

Источник: https://nces.by/wp-content/uploads/progr-elektr-belarus.pdf

2.3 Лекция 3. Язык HTML. Технология CSS

Основные вопросы

1. Язык HTML: история создания, версии.
2. Элементы веб-страницы.
3. Синтаксис языка HTML.
4. Методы реализации динамических страниц:
5. Основы поисковой оптимизации, SMM.
6. Особенности поискового продвижения ресурсов культуры и искусства в Интернет.

Цель: Изучение структуры документа и синтаксиса языка HTML, знакомство с методами поискового продвижения: поисковой оптимизацией и маркетингом в социальных сетях, особенностями поискового продвижения веб-ресурсов культуры и искусства.

Основные определения.

Веб-страница – гипермедийный HTML-документ.

HTML (Hyper Text Markup Language) – язык гипертекстовой разметки – язык создания веб-страниц.

Браузер – программа, предназначенная для просмотра веб-страниц.

Веб-сайт – группа веб-страниц, связанных единой темой, схемой оформления и гипертекстовыми ссылками.

Язык HTML.

Язык, предназначенный для создания форматированного текста, который насыщен изображениями, звуком, анимацией и ссылками на другие объекты, например, гипертекстовые документы, графические файлы и т.д., разбросанные по всему пространству веб-серверов Интернет/Интранет.

Версии HTML:

HTML – данный стандарт не использовался, но стал основой для других версий языка;

HTML+ – разработан в 1993 г.;

HTML 2.0 – данная спецификация появилась в 1994 г.;

HTML 3.0 – проект данной спецификации был опубликован в 1995 г., но в качестве стандарта реализован не был;

HTML 3.2 – первая публикация в 1996 г.;

HTML 4.0.

HTML 5 – с 2014 г.

Веб-страница с точки зрения файловой структуры.

Веб-страница – это: HTML-документ (.htm) + Файлы мультимедиа + Активные компоненты.

Структура сайта:

Логическая – информационная организация.

Физическая –алгоритм размещения файлов по подкаталогам корневой папки сайта.

Имена файлов – html-страниц и папок: только латинские символы, цифры, некоторые спецсимволы, без пробелов и никакой кириллицы! Примеры: index.htm, dom.html, Lelik-2.htm.

Именование главной (первой) страницы сайта: index.html, index.htm.

Абсолютный и относительный путь:

xyz.html – файл HTML с именем xyz.html находится в текущем каталоге;

abc/xyz.html – файл HTML с именем xyz.html находится в подкаталоге abc текущего каталога;

http://myDomain.by/abc/xyz.html – файл HTML с именем xyz.html находится в каталоге abc на сервере MyDomain.by;

../abc/xyz.html – файл HTML с именем xyz.html находится в подкаталоге abc каталога, лежащего на один уровень выше текущего;

../../abc/xyz.html – файл HTML с именем xyz.html находится в подкаталоге abc каталога, лежащего на два уровня выше текущего.

Элементы веб-страницы.

Заголовок. Текст, отображающийся в строке заголовка браузера при просмотре страницы. Первым появляется при загрузке, служит закладкой на страницу, имеет больший вес при индексации.

Текст.

Фон. Фоновый цвет. Фоновый звук. Фоновый рисунок.

Графика. В HTML-документе присутствует лишь адрес файла с графическим изображением:

D:\site\image\ris.gif (ошибка: абсолютная адресация на файл локального компьютера)

D:\мои рисунки\ris.gif (еще одна ошибка: кириллица)

..\..\image\ris.gif (правильно)

Форматы графических файлов в Интернет: растровые: gif, png, jpg и векторные: swf - флэш, svg.

Размер изображения задается в пикселях или % от ширины окна браузера.

Альтернативный текст – отображается в режиме отключения графики или неграфическими браузерами.

Функции графики: фон, миниатюры, графические карты, визуал, навигация.

Элементы веб-страницы: гиперссылки. Гиперссылка – это средство установки связей в веб-пространстве.

Элементы гиперссылки: указатель ссылки, адресная часть.

Указатель гиперссылки: выделенный фрагмент текста, графические изображение, кнопка, изображение-карта.

Адресная часть гиперссылки: ведет на определенную область страницы (закладка), иную страницу того же сайта или документ другого типа на данном компьютере, страницу внешнего сайта (URL), другой сервис Интернет (адрес e-mail, телефон).

Элементы веб-страницы: таблицы, задаются в пикселях или % окна, выполняют функции метода представления данных и средства компоновки страниц.

Динамические компоненты: счетчик посещений, рекламный баннер, формы, бегущая строка, анимационные эффекты при переходе к другой странице, полиморфная кнопка и т.д.

Фрейм – область окна браузера, в которой осуществляется просмотр отдельного HTML-документа.

Установочный файл фреймов – отдельный документ HTML, задающий способ раскладки и форматирования фреймов, и альтернативное представление для браузеров, не поддерживающих фреймы.

Зачем изучать HTML: гибкость; глубина понимания; упрощение отладки; цена; независимость.

Синтаксис языка HTML.

Тег (дескриптор) – инструкция языка HTML. Тег может быть открывающий (<тег>) и закрывающий (</тег>), причем открывающий тег может иметь атрибуты (параметры), влияющие на его поведение.

Контейнер (блок)– структура, состоящая из открывающего или открывающего и закрывающего тегов.

Примеры:

<ТЕГ параметр1=значение1 параметр2=значение2 ...>
текст
другие конструкции
</ ТЕГ>
<ТЕГ параметр1=значение1...> текст другие конструкции</ТЕГ>
<ТЕГ параметр1=значение1...> текст другие конструкции

Кодировки текста - однобайтовые – 256 символов: ASCII; КОИ8 (код обмена информацией) (KOI8-R); CP1251; CP866; ISO 8859-5; двухбайтовые – 65636 символов: UNICODE; четырехбайтовые: ISO 10646; другие: UTF-8 – является комбинацией ISO 10646 и UNICODE.

Структура документов HTML:

<!DOCTYPE HTML PUBLIC “-//W3//DTD HTML 4.0//EN”
<HTML>
<HEAD>
<TITLE>
….заголовок
</TITLE>
</HEAD>
<BODY>
... тело документа
</BODY>
</HTML>

Документ HTML является блочным элементом и сам состоит из блоков. Два основных блока – это HEAD (определяет свойства документа) и BODY (определяет тело документа). Каждый из них включает другие блоки. HTML-

документ состоит из стандартных элементов разметки: заголовки, списки, таблицы, параграфы и т.д., - которые разделены на два типа: строчные и блочные. К блочным относятся параграф, список, таблица. К строчным относятся начертание, текст гипертекстовых ссылок.

Содержимое области HEAD. В области заголовка документа могут содержаться как теги, описывающие свойства документа, так коды программ (сценариев), вызываемых в теле документа. Примеры тегов:

TITLE – определяет текст, отображаемый в заголовке браузера;

META – определяет различные свойства документа, например, тип кодировки, ключевые слова, описание документа, автор и т.д.;

base href – указывает адрес документа;

link – определяет ссылку на другой файл.

Пример организации области HEAD:

<HEAD>

<TITLE>Моя страничка</TITLE>

<META name=“keywords” content=“список терминов”>

<META name=“description” content=“краткое описание содержания...”>

<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-1251">

<META NAME="Author" CONTENT=«Svetlana Hancharova">

<META NAME="GENERATOR" CONTENT="Mozilla/6.05 [en] (WinNT; I) [Netscape]">

<base href=“http://www.buk.by/itk.html”>

<link rel=“StyleSheet” type=“text/css” href=“http://www.buk.by/site.css”>

</HEAD>

Элемент BODY. Тег <BODY> определяет тело документа, имеет различные атрибуты для определения цвета фона документа, цвета текста, цвета ссылок и т.д. Пример записи:

<BODY параметр1=данные1, параметр2=данные2...>

текст документа

</BODY>

Параметры элемента BODY:

bgcolor – определяет цвет фона для тела документа;

text – определяет цвет, используемый при выводе на экран текста из данного документа;

background – определяет адрес URL, откуда будет браться изображение для фона текущего документа;

link – определяет цвет, который будет использоваться при выводе на экран текста из еще не выбранных вами гипертекстовых связей;

vlink – определяет цвет, который будет использоваться при выводе на экран текста из уже проверенных вами гипертекстовых связей;

alink – задает цвет, которым будут выделяться в тексте гипертекстовые связи в тот момент, когда пользователь щелкает по ним клавишей мыши.

Элементы, задающие шрифт:

<FONT size=… color=...> текст </FONT>

size - устанавливает размер шрифта. Изменяется в пределах от 1 до 7. Размер по умолчанию – 3. Может быть указан относительный размер, например size="+1" или size="-2";

color – указывает цвет, которым будет выделен данный текст. Цвета задаются в виде RGB-значения с шестнадцатеричной нотацией, либо выбирается один из 16 стандартных цветов, описанных в элементе BODY при рассмотрении атрибута BGCOLOR.

Элементы форматирования на уровне блоков:

Параграф <p align=(left|center|right|justify)>

Заголовок <H1|H2|H3|H4|H5|H6 align=...>

Горизонтальная линия <hr align=... size=... width=...>

Перевод каретки (новая строка)

Элементы, задающие начертание:

<TT> телетайпный текст

<I> *стиль с наклонным шрифтом*

<B> **стиль с жирным шрифтом**

<U> стиль с подчеркиванием текста

<BIG> печать текста шрифтом увеличенного размера

<SMALL> печать текста шрифтом уменьшенного размера

<SUB> печать текста со сдвигом вниз (нижний индекс)

<SUP> печать текста со сдвигом вверх (верхний индекс)

<STRIKE> ~~стиль с перечеркиванием текста~~

Маркированные списки:

```
<UL>
<LI> ... первый пункт списка
<LI> ... второй пункт списка
...
</UL>
```

Нумерованные списки:

```
<OL>
<LI> ... первый пункт списка
<LI> ... второй пункт списка
...
</OL>
```

Таблица

```
<TABLE BORDER=...  WIDTH=...  >
<TR>
<TD параметры=… > 1-я клетка 1-ой строки </TD>
<TD параметры=... > 2-я клетка 1-ой строки </TD>
</TR>
<TR>
<TD> 1-я клетка 2-ой строки </TD>
...
</TR>
</TABLE>
```

Типы связей. Связь, устанавливаемая между гипертекстовыми документами, называется ссылкой (link) или гиперссылкой (hyperlink). Существуют следующие типы связей:

– якорь (anchor, target) или закладка (bookmark) – установка метки перед определенным блоком документа;

– ссылка на закладку (якорь) – обращение к установленной метке;

– относительная ссылка – путь к файлу относительно корневого каталога узла;

– абсолютная ссылка – полный путь к файлу с указанием доменного адреса узла и пути к файлу на данному узле.

Ссылки:

 текст или указатель на объект

Примеры:

Щелкните по этой ссылке

 Белый кораблик!

Письмо

Якорь (anchor).

Определение якоря – Maryia

Обращение к якорю –Maryia

Изображения:

<img src=... alt=... align=... width=... height=... border=... hspace=... vspace=...>

<img src="ship.gif" alt="Корабль моей мечты" align=left width=70 height=50 border=2 hspace=3 vspace=3>

Описание атрибутов тега img:

src – путь к файлу;

alt – текст подписи;

align – выравнивание;

width – ширина;

height – высота;

border – толщина контура;

hspace – отступ текста по горизонтали от изображения;

vspace – отступ текста по вертикали от изображения.

Формы:

<FORM METHOD=POST ACTION=«email»>

<INPUT TYPE=CHECKBOX> - в <form> создается элемент ввода «опция»;

<INPUT TYPE=FILE> - в <form> создается элемент ввода «выбор файла»;

<INPUT TYPE=HIDDEN> - в форме создается скрытый элемент;

<INPUT TYPE=IMAGE> - создается элемент ввода «изображение»;

<INPUT TYPE=PASSWORD> - создается элемент ввода текста с обеспечением защиты содержимого;

<INPUT TYPE=RADIO> - создается элемент ввода «селекторная кнопка»;

<INPUT TYPE=RESET> - создается кнопка сброса;

<INPUT TYPE=SUBMIT> - создается кнопка передачи;

<INPUT TYPE=TEXT> - создается элемент ввода текста;

тег OPTION – с помощью конструкции тегов <option> определяется меню внутри элемента <select>;

тег SELECT – начальный и конечный теги определяют меню с несколькими вариантами выбора или список;

тег TEXTAREA – многострочная область ввода текста в форме;

</FORM>

Технология CSS (Cascading Style Sheets).

Говоря о дизайне html-документов, основной упор делается на то, что элементы документа должны точно размещаться относительно друг друга в окне браузера. Элементами документа являются текстовые блоки, графика и другие объекты, для каждого из которых определяются свои настройки. Чтобы избежать повторения настроек для различных блоков с одинаковым оформлением, была предложена технология CSS (Cascading Style Sheets – каскадные таблицы стилей). Технология CSS позволяет определять форму представления документа. Спецификация CSS level 1 (CSS1) стала стандартом в 1996 г., CSS2 – в 1998 г., а CSS3 – в 2014.

Использование элемента <STYLE> - один из основных способов внедрения таблицы стилей в HTML-документ. Элемент <STYLE> управляет как отображением элементов разметки, так и описывает стилевые свойства элементов. Позволяет определить стиль отображения для: стандартных элементов разметки (тегов); произвольных классов; объектов HTML.

Пример:

<style><!--

TAG {параметр:значение; ……} – определение стиля для конкретного тега;

TAG.Class {параметр:значение; ……} – определение класса элементов, которые будут отображаться одинаково для группы элементов разметки;

.Class {параметр:значение; ……} – определение класса элементов, которые будут отображаться одинаково;

--></style>

XML (eXtensible Markup Language). Язык XML (eXtensible Markup Language – расширяемый язык разметки) разделяет содержание и представление документа. Позволяет пользоваться любыми тегами, создаваемыми автором в процессе разработки документа. В рамках XML разрабатываются фиксированные системы разметки документов, например, MathML – язык разметки математических текстов, ChemML – язык разметки химических текстов. Об XML можно сказать, что это HTML+семантическая разметка.

Методы реализации динамических страниц:

JavaScript – модули интегрируются в файл HTML как подпрограмма;

Java – модули существуют как самостоятельные приложения (апплеты .class);

PHP (Personal Home Page) - сценарии вставляются в тот участок документа, где необходимо разместить интерактивный элемент; позволяет организовать счетчик посещений, статистику обращений к разделам сайта, защищать доступы к файлам паролями и т.д.

ASP (Active Server Page) - скрипт исполняется на сервере, пользователю отсылается готовый HTML-документ с результатами работы ASP. Программное обеспечение пользователя не имеет значения.

Flash – для создания интерактивной анимации. Разрабатывается с использованием приложения Adobe Flash. На компьютере пользователя инсталлируется надстройка к браузеру Adobe Flash Player.

DHTML – расширение стандарта HTML - использование сценариев («скриплетов»), обрабатываемых браузером совместно с кодом HTML. Позволяет создать движущиеся объекты, выпадающие меню, подсвечивающиеся кнопки и др.

XML (eXtensible Markup Language). Язык XML (eXtensible Markup Language - расширяемый язык разметки) разделяет содержимое и представление документа. Позволяет пользоваться любыми тегами, создаваемые автором в процессе разработки документа. В рамках XML разрабатываются фиксированные системы разметки документов, например, MathML - язык разметки математических текстов, ChemML - язык разметки химических текстов. Об XML можно сказать, что это HTML + семантическая разметка.

Средства создания веб-страниц:

– простейшие текстовые редакторы - Блокнот, Far Editor, Norton Editor;

– специализированные HTML-редакторы - AceHTML или Notepad++;

– редакторы визуального проектирования WYSIWYG – MS VisualStudio.NET., Adobe Muse и т.д.

– системы управления сайтом (веб-контентом).

Content management systems (CMS) – программный комплекс для создания, публикации, редактирования и организации содержимого, настройки и администрирования сайта.

Среди основных функций:

– хранение содержимого сайта в БД и/или в файловой системе, наличие средств для управления информационным содержимым;

– стандартизация представления информации на сайте, использование шаблонов, позволяющих централизованное редактирование;

– масштабируемость по функциональности и по нагрузке;

– управление пользователями, разделение ролей.

Преимущества CMS:

– уменьшение трудозатрат на создание веб-сайтов, т.е. сроков и стоимости разработки;

– возможность концентрироваться в ходе разработки на обеспечении удобства для целевых пользователей;

– повышение качества информационного продукта;

– снижение трудоемкости и стоимости поддержки информации, снижение требований к квалификации персонала – с системами может работать и неспециалист в сфере ИТ (контент-менеджер, вебмастер, дизайнер);

– улучшение возможностей дальнейшего развития продукта – за счёт модульной архитектуры, разделения данных и их представления.

Особенности и недостатки CMS:

– необходимость работать с чужими кодом и архитектурой системы;

– требовательность к компетенциям программиста (специализация);

– более низкая производительность по сравнению со специализированными программными решениями (универсальность не бывает «бесплатной»);

– требовательность к программным ресурсам: PHP; MySQL, PostgreSQL; ASP.NET, C#, Java, VB.NET, Python и др.; MSSQL, Oracle

– необходимость настройки и доработки под конкретные задачи, которая не всегда осознаётся неспециалистами (CMS легко принять за готовое решение).

Основы поискового продвижения.

Чтобы увеличить аудиторию пользователей, ежедневное количество посетителей необходимо популяризировать ресурс (сайт), то есть произвести его оптимизацию и продвижение в ведущих поисковых системах Интернет.

Под продвижением сайта понимается комплекс мер по обеспечению посещаемости сайта целевыми посетителями. Целевые посетители – это посетители, которые заинтересованы в информации товаре или услуге, представленной на продвигаемом сайте.

Одним из важнейших этапов продвижения сайта является поисковая оптимизация (Search engine optimization), которая представляет собой комплекс мер по повышению позиций сайта в поисковых системах, и, таким образом, позволяет увеличить его целевую посещаемость. Это длительный и комплексный процесс сбора всех возможных усилий на популяризацию конкретного ресурса.

Аудитория радио, телевидения, печатных СМИ, наружной рекламы и Интернета качественно отличается. Отличие не только в том, что Интернет является общей средой, но и в том, что продвижение сайта, в отличие от роликов, магистральных щитов и модулей в газетах – именно **целевая** реклама.

Существуют белые, серые и черные методы продвижения.

«Белые» методы (также называют «Белое SEO») являются законными с точки зрения поисковых систем, но продвижение сайта белыми методами занимает больше времени, и требует большего денежного вложения ввиду своей трудоемкости.

«Серые» методы можно трактовать как что-то среднее между черными и белыми методами продвижения, так как они условно законные. В сущности, эти способы продвижения сайта являются полулегальными и поэтому не лишены риска, что продвигаемый сайт не «забанят».

Под «черными» методами продвижения сайта понимают применение методов, запрещенных правилами поисковых систем, в частности, нарушающие лицензию Яндекса и Google, а также нарушающие правила участия в рейтинге сайтов (например, TOP-100). Использование таких методов – прямая дорога сайту в «бан» (исключение из индекса поисковой системы).

Методы продвижения сайта делятся на две категории:

1. Внутренняя оптимизация (работа с сайтом: выбор ключевых слов, прописание мета-тэгов, оптимизация html-кода, перелинковка страниц, оригинальные тексты, регулярное обновление конткнта, RSS-рассылки, файлы robots.txt и sitemap.xml и др.).

2. Работа с внешними факторами (внешние ссылки, упоминания на других интернет-ресурсах, регистрация в поисковых системах и каталогах, реклама ресурса на тематических форумах, участие сайта в рейтингах, продвижение сайта в социальных медиа, продвижение сайта в закладочных сервисах, контекстная реклама, обмен банерами, создание рассылок и др.).

Особенности продвижения сайтов общественных и некоммерческих организаций.

Некоммерческие цели – когда сайты созданы не для того, чтобы приносить прибыль, предлагать, продвигать, рекламировать или продавать товары, продукты, товары и услуги. Такие интернет-проекты называются некоммерческими. Тематические ресурсы, сайты государственных учреждений сферы культуры и искусства, общественных организаций, благотворительных фондов и др., занимающиеся различными социальными проектами, рзличные клубы по интересам, образовательные учреждения.

Основная цель некоммерческих сайтов или сайтов общественных организаций – информирование определенной группы людей, следовательно, деятельность по продвижению таких сайтов должна быть направлена на обновление контента сайта (добавление новостей, событий, объявлений, касающихся самой организации).

Одной из особенностей продвижения таких сайтов является широкое использование методов офлайн-рекламы. Организации, которые занимаются активной общественной деятельностью, проводя конференции, семинары, акции, мероприятия, могут продвигать свой сайт в офлайн режиме, распространяя брошюры, листовки, бюллетени с адресом на собственный веб-сайт, а также указывая его на баннерах и растяжках в помещениях, где проводят свои мероприятия.

Достаточно эффективным методом продвижения является размещение пресс-релизов. Пресс-релизы представляют собой короткие (одна-две страницы) официальные информационные сообщения для прессы, рассказывающие о различных мероприятиях, событиях в жизни общественной организации или учреждения. Их можно размещать на специализированных сайтах, как правило, бесплатно. Размещение пресс-релизов дает многосторонний эффект. Во-первых, их могут прочитать посетители тех ресурсов, где вы их размещаете. Во-вторых, сайтами, где есть пресс-релизы, пользуются многие журналисты. Ваш пресс-релиз может стать основой для публикаций в сетевых и печатных изданиях. В-третьих, разместив в пресс-релизах гиперссылки на свои ресурсы, вы сможете повысить их цитируемость.

Для некоммерческого сайта в Интернет можно предложить следующие методы продвижения: внутренняя оптимизация; регистрация в поисковых системах; обмен баннериками с другими сайтами; обмен ссылками с другими сайтами; участие в рейтинговых системах, создание рассылки сайта (RSS-рассылки), рекламирование на форумах, продвижение в социальных сетях.

3 ПРАКТИЧЕСКИЙ РАЗДЕЛ

3.1 Практическое занятие 1. Технологии веб-дизайна

Цель урока: Ознакомиться с правовыми основами функционирования интернета в Беларуси, изучить технические компоненты публикации сайта, понять основы его продвижения и аналитики.

1. Законодательное регулирование белорусского сегмента сети интернет

Ключевым документом является Закон Республики Беларусь от 17 июля 2018 г. № 130-З «О внесении изменений и дополнений в Закон Республики Беларусь «Об информации, информатизации и защите информации». Он ввел понятие «оператор интернет-ресурса» и установил ряд требований.

Основные требования:

– Обязательная идентификация пользователей при использовании онлайн-сервисов (комментарии, форумы).

– Регистрация интернет-ресурсов в Министерстве информации РБ (для некоторых категорий, например, СМИ, онлайн-магазинов).

– Сбор и хранение данных о пользователях оператором ресурса.

– Запрет на распространение информации, наносящей вред национальным интересам.

Полезные источники:

– Эталон-онлайн (https://etalonline.by/) (официальный источник).

– Разъяснения Министерства информации: [mininform.gov.by] (https://mininform.gov.by/ru/) (раздел «Деятельность» -> «Интернет-пространство»).

2. Хостинг сайтов

Хостинг – услуга по предоставлению вычислительных мощностей для размещения информации на сервере, постоянно находящемся в сети Интернет.

Требования к хостингу в РБ: Оператор хостинга должен быть резидентом РБ и соблюдать законодательство, в частности, обеспечивать возможность ограничения доступа к ресурсу по требованию уполномоченных органов.

Серверные платформы:

– Unix-подобные хостинги (Linux, FreeBSD): Наиболее популярны благодаря стабильности, безопасности и бесплатности. Поддерживают PHP, Perl, MySQL.

– Windows-хостинги: Используются для сайтов на ASP.NET и с базами данных MS SQL Server.

Веб-серверы (ПО, которое обрабатывает запросы от браузеров):

– Apache: Самый распространенный, гибкий, с поддержкой .htaccess.

– Официальный сайт Apache (https://httpd.apache.org/)

– IIS (Internet Information Services): Веб-сервер от Microsoft, тесно интегрирован с ОС Windows.

– Официальная документация IIS (https://www.iis.net/)

– Nginx: Высокопроизводительный сервер, часто используется как обратный прокси-сервер или для раздачи статического контента. Официальный сайт Nginx (https://nginx.org/ru/)

Задачи администрирования веб-сервера: установка и настройка ПО, обеспечение безопасности, мониторинг нагрузки, обновление программного обеспечения, резервное копирование.

Белорусские провайдеры (хостеры):

– active.by – крупнейший национальный регистратор и хостинг-провайдер;

– hosting.by – популярный белорусский хостинг-провайдер;

– reg.by – еще один крупный игрок на рынке.

Стоимость хостинга: Зависит от ресурсов (CPU, RAM, дискового пространства), типа хостинга (виртуальный, VPS, выделенный сервер) и дополнительных услуг. Стартует от ~5-10 BYN/мес. за виртуальный хостинг.

Специфика бесплатного хостинга:

Недостатки: реклама на вашем сайте, ограниченные ресурсы, низкая скорость и надежность, часто отсутствует поддержка собственного домена, нет гарантий сохранности данных.

Не подходит для коммерческих и серьезных проектов.

3. Система доменных имен (DNS)

DNS (Domain Name System) – это «телефонная книга» интернета, которая преобразует человеко-читаемые доменные имена (например, `google.com`) в машинно-читаемые IP-адреса (например, `142.251.42.14`).

Выбор доменного имени: должно быть коротким, запоминающимся, отражать тематику сайта, легко набираться на клавиатуре.

Регистрация доменов: Производится через аккредитованных регистраторов (например, те же `active.by`, `hosting.by`). Для домена .BY требуется предоставить данные владельца.

Регламентация доменных имен госорганов РБ: Для государственных организаций установлен домен верхнего уровня. GOV.BY. Использование регулируется Оператором АСУ «Государственный регистр информационных ресурсов и информационных систем».

Источник: [Положение о порядке регистрации доменов. GOV.BY] (http://www.programmy.by/docs/doc_ru/index_1_3.html)

4. Публикация веб-ресурса

Способы публикации:

1. Загрузка файлов через FTP (File Transfer Protocol) или его защищенную версию SFTP.

2. Использование панели управления хостингом (например, cPanel, ISPmanager) с встроенным файловым менеджером.

3. Автоматическая публикация из систем управления контентом (CMS) или Git-репозиториев.

FTP-клиенты – программы для передачи файлов между компьютером и сервером.

FileZilla: Бесплатный кроссплатформенный клиент (https://filezilla-project.org/)

WinSCP: Популярный клиент для Windows с поддержкой SCP и SFTP (https://winscp.net/).

5. Онлайн-конструкторы сайтов

Онлайн-конструктор – это сервис, позволяющий создать сайт без знания языков программирования, используя визуальный редактор и готовые шаблоны.

Преимущества: скорость создания, простота, не требует навыков программирования, все технические вопросы решает провайдер конструктора.

Недостатки: ограниченная функциональность, привязка к платформе, затруднен перенос сайта на другой хостинг.

Обзор популярных конструкторов:

Tilda Publishing: Мощный конструктор для создания лендингов и блогов с богатой библиотекой блоков (https://tilda.cc/)

Wix: Конструктор с большим количеством шаблонов и гибким дизайнером (https://www.wix.com/)

WordPress.com: Упрощенная и хостинговая версия популярной CMS WordPress (https://wordpress.com/)

6. Поисковое продвижение (SEO)

SEO (Search Engine Optimization) – это комплекс мер по внутренней и внешней оптимизации сайта для поднятия его позиций в результатах выдачи поисковых систем (Яндекс, Google) по определенным запросам.

Белые, серые и черные методы SEO.

Белые: Полное соответствие правилам поисковых систем (качественный контент, семантическая разметка, естественные ссылки).

Серые: Методы на грани правил (например, покупка ссылок в статьях, но без прямого указания на это).

Черные: Прямое нарушение правил (клоакинг, дорвеи, скрытый текст). Приводят к бану (исключению сайта из индекса) поисковой системы.

Внутренняя оптимизация: Работа над самим сайтом.

– Качественный уникальный контент.

– Мета-теги (title, description).

– Структура URL, заголовки (H1-H6).

– Оптимизация изображений.

– Файлы `robots.txt` (указания для поисковых роботов) и `sitemap.xml` (карта сайта для роботов).

– Повышение скорости загрузки.

Внешняя оптимизация: Работа над авторитетом сайта в интернете.

– Регистрация в каталогах (например, каталог Яндекса).

– Естественное получение ссылок с других сайтов.

– Социальные закладки (упоминания в соцсетях).

– Комментирование в блогах/форумах (осмысленное, с релевантным ресурсом).

– Контекстная и баннерная реклама (не прямое SEO, но инструмент привлечения трафика).

– RSS-рассылки для удержания постоянной аудитории.

Полезные источники по SEO:

– Помощь Google для веб-мастеров: [Руководство по поисковой оптимизации (SEO)] (https://developers.google.com/search/docs/fundamentals/seo-starter-guide?hl=ru)

– Блог Яндекса для веб-мастеров: [Yandex Webmaster] (https://yandex.ru/support/webmaster/)

7. Инструменты аналитики

Веб-аналитика – это измерение, сбор, анализ и отчетность о данных о посещаемости веб-сайта с целью понимания и оптимизации его использования.

Инструменты аналитики веб-разработчика:

– Яндекс.Метрика: Мощный инструмент от Яндекса с детальной статистикой по аудитории, поведению и источникам трафика (https://metrika.yandex.ru/);

– Google Analytics: Мировой стандарт веб-аналитики. Глубокий анализ пользовательского поведения и интеграция с другими сервисами Google (https://analytics.google.com/);

– Hotjar, Pendo.io, LogRocket: Инструменты для анализа поведения пользователей (тепловые карты, записи сессий, опросы) (https://www.hotjar.com/);

– Mixpanel, Amplitude: Платформы для продукт-аналитики, фокусируются на действиях пользователей (событиях) (https://mixpanel.com/);

– Matomo (ранее Piwik): Открытое ПО для веб-аналитики, которое можно разместить на своем сервере (https://matomo.org/);

– SimilarWeb: Анализ трафика и статистики не только своего, но и конкурентных сайтов (https://www.similarweb.com/).

Инструменты аналитики хостинга: Это, как правило, логи веб-сервера (Apache, Nginx), которые анализируются с помощью специальных программ (например, AWStats, Webalizer), предоставляющих данные о файлах, посетителях, ссылающихся страницах и т.д. Часто интегрированы в панель управления хостингом.

Задание в классе:

1. Найти на сайте Министерства информации РБ перечень зарегистрированных интернет-ресурсов.

2. Сравнить тарифы двух белорусских хостинг-провайдеров по 3 параметрам: цена, дисковое пространство, поддержка PHP/MySQL.

3. Используя публичные данные SimilarWeb или Яндекс.Метрики (бенчмаркинг), проанализировать примерный трафик любого известного белорусского сайта.

3.2 Практическое занятие 2. Основы проектирования сайта

Цель урока: Сформировать у учащихся системное понимание процессов анализа и проектирования веб-сайта, познакомить с ключевыми понятиями, инструментами и лучшими практиками.

Тщательное планирование разработки сайта – залог его успеха и экономии ресурсов в будущем. Без четкого плана сайт превращается в свалку информации, где невозможно найти нужное, и пользователи уходят.

Этап 1: Анализ и стратегия – Определить, «зачем» и «для кого» создают сайт.

1. Анализ предметной области и назначение сайта

Вопросы для анализа:

– О чем сайт? (e.g., интернет-магазин электроники, блог о путешествиях, корпоративный сайт университета).

– Какова его главная цель? (Продавать, информировать, развлекать, собирать заявки).

Пример: Назначение сайта IKEA – продавать мебель и вдохновлять на решения для дома. Назначение сайта Wikipedia – предоставлять свободную энциклопедическую информацию.

[Сайт IKEA] (https://www.ikea.com/) – пример коммерческого сайта с четким назначением.

[Wikipedia] (https://www.wikipedia.org/) – пример информационного ресурса.

2. Пользователи сайта и их характеристика

«Целевая аудитория» (ЦА) – это группа людей, для которых создается сайт.

Методы: Создание User Personas (Персоны пользователей) – вымышленных, но реалистичных портретов типичных пользователей.

Пример Персоны:

– Имя: Мария, 35 лет.

– Цель: Быстро найти и купить детскую кровать онлайн с доставкой на следующий день.

– Сценарий: Заходит на сайт с телефона, использует поиск, фильтры по цене и наличию, смотрит фото и отзывы.

– Изображение: Пример визуализации пользовательской персоны.

[Пример User Persona на Dribbble] (https://dribbble.com/shots/2283452-User-Persona-Template/attachments/2283452-User-Persona-Template?mode=media)

3. Определение задач сайта – это конкретные действия, которые должны выполнять пользователи для достижения цели сайта.

– Для IKEA: 1) Найти товар; 2) Добавить в корзину; 3) Оформить заказ.

– Для блога: 1) Прочитать статью; 2) Подписаться на рассылку; 3) Поделиться в соцсетях.

Этап 2: Проектирование структуры и навигации

1. Проектирование логической структуры (Информационная архитектура – это организация информации на сайте для ее простого и интуитивного восприятия)

Инструмент: Сайтмап (Card Sorting) – схема иерархии всех страниц сайта.

Пример: Структура простого сайта-портфолио:

– Главная

– Обо мне

– Услуги

– Портфолио (с подразделами: Веб-дизайн, Графика)

– Блог

– Контакты

– Изображение: Пример схемы сайта (Site Map).

[Пример Site Map на Visual Paradigm] (https://www.visual-paradigm.com/guide/ux-design/what-is-website-sitemap/)

2. Принципы и типы навигации

Главное правило: Пользователь всегда должен понимать, «где он находится», «куда он может пойти» и «как вернуться назад».

Типы навигации:

– Глобальная (Основное меню): Постоянный элемент на всех страницах. Обычно в шапке сайта. Пример – https://www.apple.com/ – меню всегда вверху.

– Локальная (Вспомогательное меню): Меню для раздела (e.g., «Категории» в блоге).

– «Хлебные крошки»: Показывают путь от главной страницы до текущей. «Главная > Блог > Веб-дизайн > Этапы проектирования». Пример – https://www.amazon.com/ – «хлебные крошки» под основным меню.

– Футер (подвал): Содержит повторяющиеся ссылки (О нас, Контакты, Политика конфиденциальности).

Этап 3: Проектирование контента и визуальной части

1. Проектирование информационного наполнения (Контент-стратегия)

Типы контента: Текст, изображения, видео, аудио, формы, товары.

Выбор информационных шаблонов: Стандартные схемы расположения контента на странице.

– Шаблон «Hero Section»: Большое изображение/видео + заголовок + призыв к действию (CTA) на главной. [Пример Hero Section на Spotify] (https://www.spotify.com/).

– Шаблон «Блог/Новости»: Сетка из карточек с превью статей.

– Шаблон «Карточка товара»: Фото, название, цена, кнопка «В корзину».

2. Концепция графического дизайна

На основе анализа: Стиль сайта должен соответствовать бренду и целевой аудитории.

Составляющие:

– Цветовая палитра. Пример: Кальмовая палитра для спа-салона, яркая – для детского центра. [Инструмент для подбора цветов - Coolors.co] (https://coolors.co/).

– Типографика (Шрифты). Не более 2-3 шрифтов на сайте.

– Фирменный стиль (Логотип, иконки).

3. Средства прототипирования

Что такое прототип? Интерактивный «черновик» сайта без дизайна, показывающий структуру и основные сценарии.

Инструменты:

– Figma (бесплатный для отдельных пользователей) – самый популярный инструмент. [Пример интерактивного прототипа в Figma Community] (https://www.figma.com/community/file/888593391096937625).

– Adobe XD, Sketch, Balsamiq (для более схематичных "wireframes").

– Изображение: Сравнение Wireframe (каркас) и Mockup (готовый дизайн). [Пример Wireframe vs Mockup на UX Planet](https://miro.com/guides/wireframe/what-is-a-wireframe).

Этап 4: Реализация и тестирование

Программирование (Верстка и разработка): Front-end (то, что видит пользователь) и Back-end (логика на сервере).

Наполнение контентом: Перенос текстов и изображений в CMS (систему управления контентом, например, WordPress).

Тестирование: Проверка сайта на ошибки.

– Кроссбраузерность: Корректное отображение в разных браузерах, например: Chrome, Firefox, Safari.

– Адаптивность: Корректное отображение на мобильных устройствах.

– Юзабилити-тестирование: Приглашение людей из ЦА для выполнения задач на сайте и сбор обратной связи.

[Пример плохого и хорошего UX] (https://www.userpeek.com/blog/bad-website-design-examples)

Ключевые принципы: Usability и веб-тексты.

Важнейшие принципы Usability (Юзабилити) – «10 Эвристик Якоба Нильсена: Классические правила удобства интерфейса».

1. Видимость статуса системы: Пользователь должен всегда видеть, что происходит (e.g., загрузка, успешная отправка формы).

2. Соответствие между системой и реальным миром: Используйте понятные слова, а не тех. жаргон.

3. Свобода и контроль: У пользователя должна быть кнопка «Отмена» и «Назад».

4. Единообразие и стандарты: Кнопки и элементы должны выглядеть и вести себя одинаково на всем сайте.

Источник: [10 эвристик юзабилити от Якоба Нильсена] (https://www.nngroup.com/articles/ten-usability-heuristics/).

Особенности подготовки текстов для веб-страниц.

Пользователи не читают, а сканируют. Они бегло просматривают страницу.

Правила веб-текстов:

– Использовать подзаголовки (как в этом конспекте).

– Делить текст на короткие абзацы (до 5-7 строк).

– Применять маркированные списки.

– Выделять ключевые слова.

– Писать информацию просто и по делу.

Пример: Сравните два описания товара. «Данный кухонный комбайн обладает высокой производительностью и многофункциональностью» или «Комбайн с 12 насадками. Измельчит, перемешает и взобьет за 30 секунд».

Задание в классе: Выберите любой известный вам сайт и проанализируйте его по пунктам: целевая аудитория, структура (попробуйте нарисовать схему), тип навигации, соблюдение принципов юзабилити.

Итог предоставить в виде доклада с помощью программных средств Microsoft PowerPoint или Word.

3.3 Практическое занятие 3. Язык HTML

Цель урока: Познакомить учащихся с базовыми понятиями HTML, научить создавать простые веб-страницы, содержащие текст, ссылки и основные элементы форматирования.

1. Введение: Что такое HTML?

HTML (HyperText Markup Language) – это не язык программирования, а язык разметки гипертекста. Он определяет структуру и содержание веб-страницы.

Аналогия: Если представить веб-страницу как человека, то:

– HTML – это скелет и органы (структура и содержание).

– CSS – это кожа и одежда (внешний вид).

– JavaScript – это мышцы и мозг (поведение и интерактивность).

Декларирование типа документа (`<!DOCTYPE>`)

Эта строка помогает браузеру правильно отображать страницу. Для современного HTML5 она очень проста:

```html
<!DOCTYPE html>
```

Кодировка текста

Важно указывать кодировку (обычно UTF-8), чтобы браузер правильно отображал кириллицу и другие символы.

```html
<meta charset="UTF-8">
```

Ссылка на материал: [Документация по `<!DOCTYPE>` на MDN](https://developer.mozilla.org/ru/docs/Glossary/Doctype)

2. Синтаксис HTML: Теги и Элементы

Тег – это элемент языка, заключенный в угловые скобки `< >`.

Элемент (Контейнер) – обычно состоит из открывающего тега, содержимого и закрывающего тега.

```html
<открывающий_тег>содержимое</закрывающий_тег>
```

Атрибуты – дополнительные параметры, которые настраивают элементы. Указываются в открывающем теге.

```html
<тег атрибут="значение">содержимое</тег>
```

Примеры:

Тег абзаца: `<p>Это текст внутри абзаца.</p>`

Тег с атрибутом: `<p class="important">Это важный абзац.</p>`

3. Базовая структура HTML-документа

```html
<!DOCTYPE html>
<html lang="ru">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Моя первая веб-страница</title>
</head>
<body>
<h1>Это главный заголовок страницы!</h1>
<p>А это первый абзац текста.</p>
<!-- Это комментарий, его не видно на странице -->
</body>
</html>
```

Разбор структуры:

1. `<html>`: Корневой элемент всей страницы. Атрибут `lang` помогает поисковым системам и скринридерам.

2. Область `<head>`: Служебная информация для браузера и поисковых систем. Её содержимое не отображается на самой странице.

`<meta charset="UTF-8">` – кодировка.

`<meta name="viewport">` – настройки для корректного отображения на мобильных устройствах.

`<title>` – заголовок страницы во вкладке браузера. Крайне важен для SEO!

3. Область `<body>`: Тело документа. Здесь размещается всё содержимое, которое видят пользователи (заголовки, текст, изображения и т.д.).

Ссылка на материал: [Структура HTML-документа на MDN] (https://developer.mozilla.org/ru/docs/Learn/HTML/Introduction_to_HTML/Getting_startedанатомия_html_документа)

4. Основные элементы для работы с текстом

Заголовки (блочные элементы). Имеют 6 уровней, от самого важного `<h1>` до наименее важного `<h6>`.

```html
<h1>Главный заголовок (должен быть один на странице)</h1>
<h2>Заголовок раздела</h2>
<h3>Подраздел</h3>
```

Элементы форматирования текста (строчные элементы):

```html
<p>Это <b>жирный</b> текст, а это <strong>важный</strong> текст (семантически).</p>
<p>Это <i>курсив</i>, а это <em>акцентированный</em> текст.</p>
<p>Это <mark>выделенный</mark> текст.</p>
<p>Это <small>маленький</small> текст.</p>
<p>Формула воды: H<sub>2</sub>O.</p>
<p>Степень: 10<sup>2</sup> = 100.</p>
```

Списки:

```html
<!-- Маркированный список -->
<ul>
<li>Первый пункт</li>
<li>Второй пункт</li>
</ul>
<!-- Нумерованный список -->
<ol>
<li>Первый пункт</li>
<li>Второй пункт</li>
</ol>
```

Ссылка на материал: [HTML-элементы для текста на MDN] (https://developer.mozilla.org/ru/docs/Web/HTML/Elementосновное_введение)

5. Ссылки и пути

Тег ссылки `<a>`: Создает гиперссылку.

Атрибут `href` (адресная часть) – определяет URL, на который ведет ссылка.

Содержимое тега (указатель ссылки) – текст или изображение, которые кликает пользователь.

```html
<a href="https://www.google.com">Это внешняя ссылка на Google</a>
```

Абсолютный и относительный путь:

Абсолютный путь: Полный URL, включающий протокол, домен и путь.

`https://site.com/images/photo.jpg`

Относительный путь: Путь относительно текущего местоположения файла.

`about.html` – файл в той же папке.

`pages/contacts.html` – файл в папке `pages`.

`../images/logo.jpg` – файл в папке `images`, которая находится на уровень выше.

Примеры относительных ссылок:

```html
<!-- Ссылка на страницу в той же папке -->
<a href="about.html">О нас</a>
<!-- Ссылка на страницу в другой папке -->
<a href="projects/index.html">Наши проекты</a>
<!-- Ссылка, ведущая на два уровня вверх, а затем в папку assets -->
<a href="../../assets/style.css">Файл стилей</a>
```

Ссылка на материал: [Что такое URL? на MDN](https://developer.mozilla.org/ru/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL)

6. Изображения, фон и таблицы

Изображения (`<img>`) – одиночный тег.

```html
<img src="path/to/image.jpg" alt="Описание изображения" width="400">
```

`src` – путь к изображению.

`alt` – обязательный альтернативный текст для доступности и SEO.

Фон – задается с помощью CSS, но упомянем для связи с планом.

```html
<body style="background-color: lightblue;">
```

Таблицы (`<table>`) – для представления табличных данных.

```html
<table border="1">
<tr> <!-- строка (table row) -->
<th>Имя</th> <!-- ячейка заголовка (table header) -->
<th>Возраст</th>
</tr>
<tr>
<td>Анна</td> <!-- ячейка данных (table data) -->
<td>25</td>
```

</tr>
</table>
```

Ссылка на материал: [Тег `<img>` на MDN] (https://developer.mozilla.org/ru/docs/Web/HTML/Element/img)

7. Формы

Формы (`<form>`) – используются для сбора данных от пользователя.

```html
<form action="/submit" method="post">
<label for="name">Ваше имя:</label>
<input type="text" id="name" name="user_name"><br><br>

<label for="email">E-mail:</label>
<input type="email" id="email" name="user_email"><br><br>

<input type="submit" value="Отправить">
</form>
```

<label>` – подпись для поля.

`<input>` – поле ввода. Атрибут `type` определяет его тип (текст, email, пароль, кнопка и т.д.).

`action` – URL, куда отправляются данные формы.

`method` – HTTP-метод (GET или POST).

Ссылка на материал: [Руководство по HTML-формам на MDN](https://developer.mozilla.org/ru/docs/Learn/Forms)

8. Инструменты для создания веб-страниц

Простые редакторы: Блокнот, Notepad++.

Редакторы кода (рекомендуется): VS Code (бесплатный, мощный, с большим количеством расширений), Sublime Text, WebStorm.

Визуальные редакторы (WYSIWYG): WordPress (через админ-панель), Adobe Dreamweaver.

Ссылка на материал: [Скачать VS Code] (https://code.visualstudio.com/)

Задание в классе:

Задание: Создать HTML-файл `index.html` со следующей структурой:

1. Корректная структура документа (doctype, html, head, body).

2. Заголовок страницы (`<title>`).

3. Один заголовок `<h1>` и несколько абзацев `<p>` с разным форматированием (жирный, курсив).

4. Нумерованный или маркированный список.

5. Две ссылки: одна внешняя (на любой сайт), одна относительная (можно создать пустой файл `about.html` рядом).

6. Изображение (скачайте любое или используйте ссылку на изображение из интернета).

7. Простая форма с полем «Имя» и кнопкой «Отправить».
```

3.4 Практическое занятие 4. Технология CSS

Цель урока: познакомиться с механизмами создания адаптивной страницы.

Задание:

1. Ознакомиться с готовым проектом – https://games.of.by/adaptiv/.
2. Отредактировать адаптивную страницу с бургер-меню:

2.1. Создать страницу по образцу (рисунок 1 – адаптивный дизайн, рисунок 2 – наполнение файлов HTML/CSS/JS-код).

2.2 Добавить в полученный проект минимум по 2 изображения, чтобы они корректно отображались на всех версиях устройств.

2.3. Подобрать различные шрифты для описания.

3. Результат предоставить для проверки.

Рисунок 1.1 – Десктопная версия страницы сайта

Рисунок 1.2 – Отображение на планшете

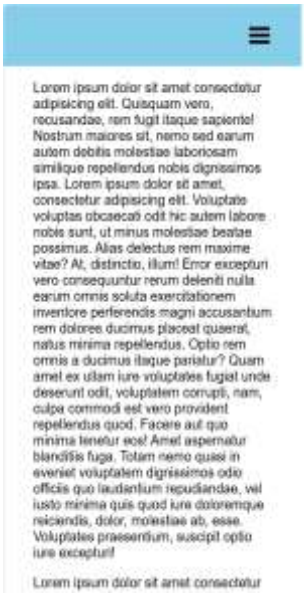


Рисунок 1.3 – Мобильная версия

Рисунок 1.4 – Отображение меню в мобильной версии

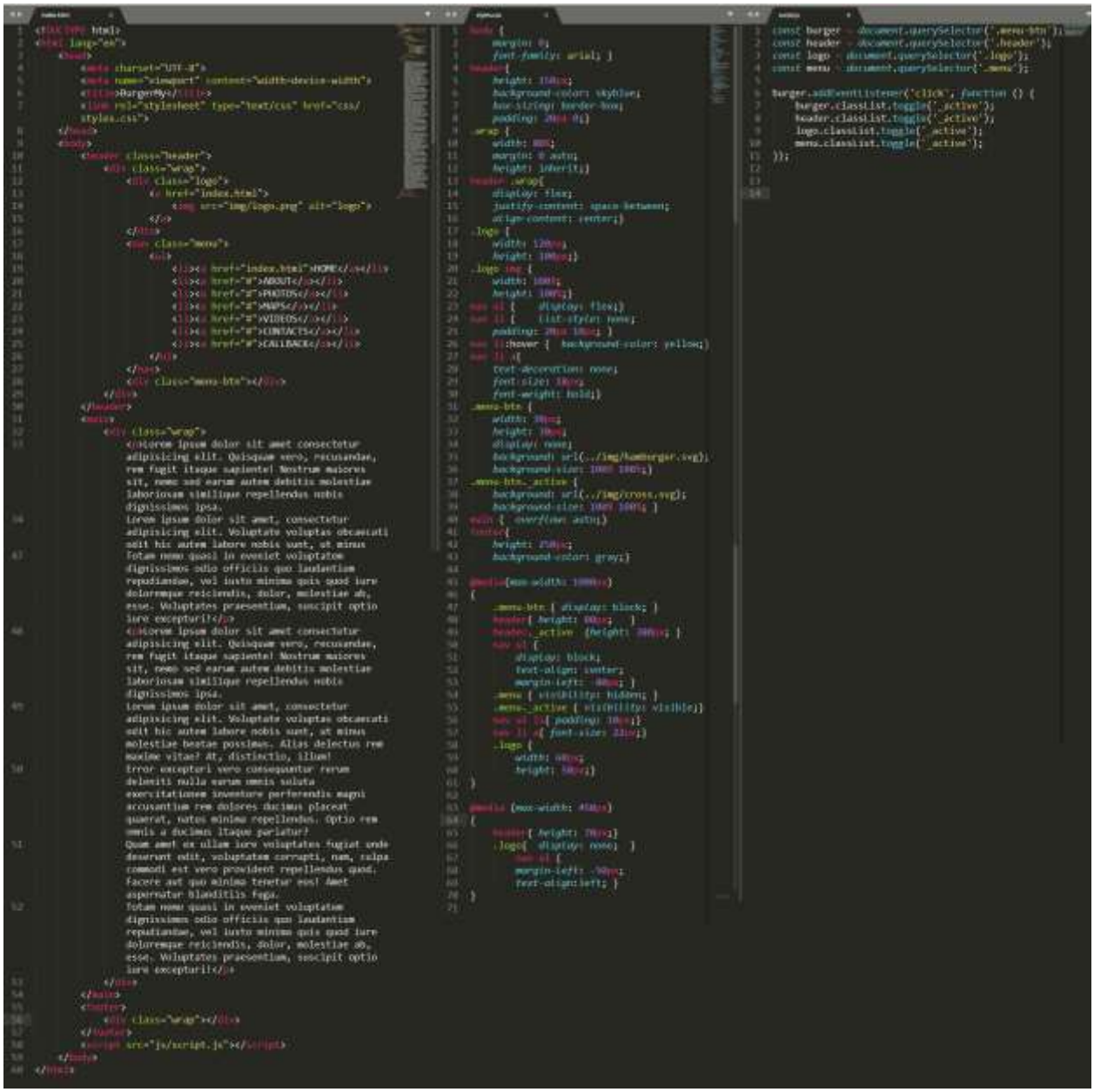


Рисунок 2 – Наполнение файлов проекта HTML/CSS/JS-код

3.5 Практическое занятие 5. Графические и звуковые элементы

Цель урока: изучить способы создания многостраничного сайта, и освоить работу с изображениями и ссылками.

Задание:

1. Ознакомиться с готовым проектом – https://games.of.by/gifs/. Главная страница (рисунок 3) состоит из 9-ти кликабельных gif-изображений. Каждое из них ведёт на свою страницу с навигацией по сайту (рисунок 4).

Ознакомиться и повторить в своем проекте: код главной страницы и стилизация (рисунок 5); файловая структура проекта (рисунок 6); код отдельной страницы для всех изображений структурно идентичный с небольшими изменениями в скрипте (рисунок 7).

2. Создать страницу с 9 gif-изображениями.

3. Реализовать переходы на отдельные страницы изображений.

4. Добавить ещё один ряд изображений (всего будет 12 изображений) в полученный проект.

5. Результат показать преподавателю.

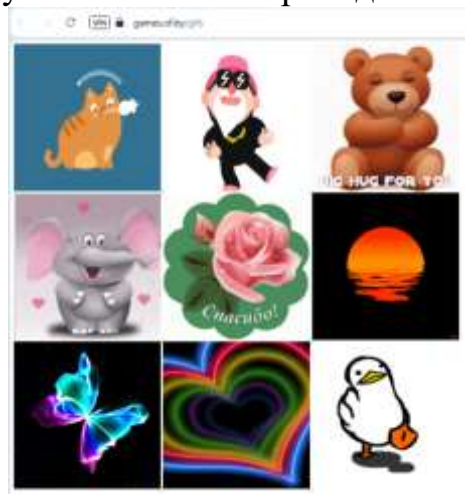


Рисунок 3 – Главная страница проекта

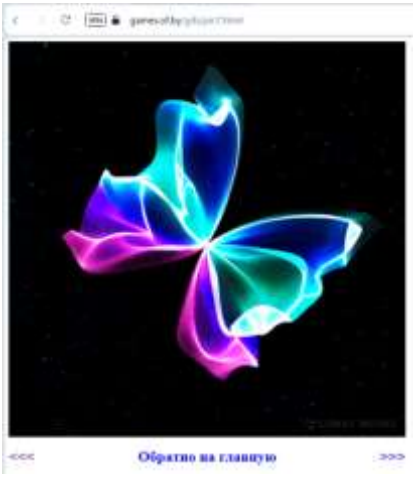


Рисунок 4 – Отдельная страница изображения с навигацией по сайту

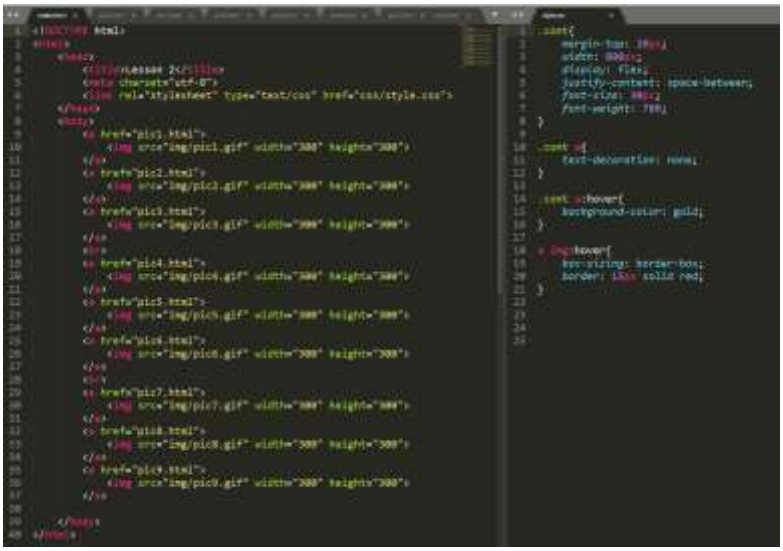


Рисунок 5 – Пример кода главной страницы и стилизации (css)

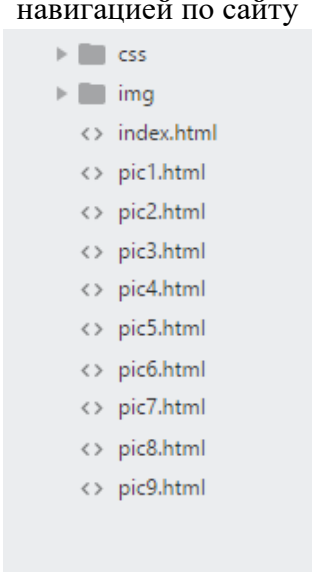


Рисунок 6 – Пример файловой структуры проекта

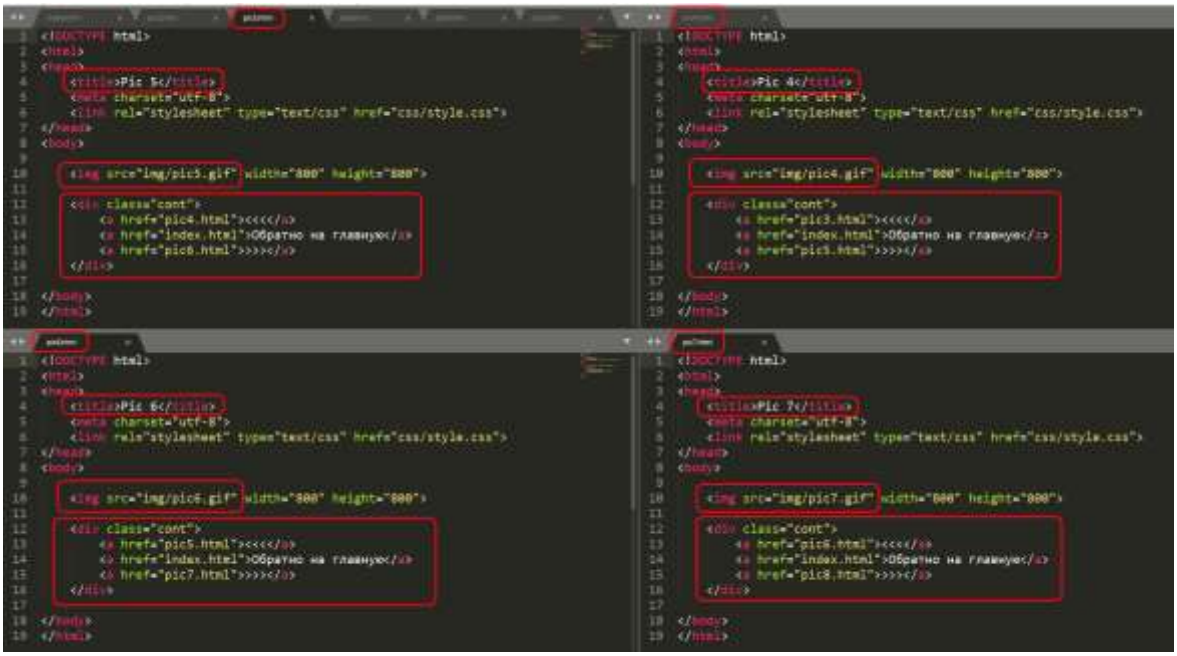


Рисунок 7 – Пример кода отдельной страницы для каждого изображения

3.6 Практическое занятие 6-7. Верстка сайта

Цель урока: изучить способы создания многостраничного сайта со статическим фоном, логотипом и контактными данными.

Задание:

1. Ознакомиться с готовым проектом – https://games.of.by/seasons.
2. Ознакомиться со скриптами готового проекта (рисунок 8-9).
3. Создать папку проекта под своим именем. Повторить все страницы из представленного проекта.
4. С готового проекта можно скопировать изображения.
5. Самостоятельно добавить 5-ю страницу с изображениями космоса.
6. Результат показать преподавателю

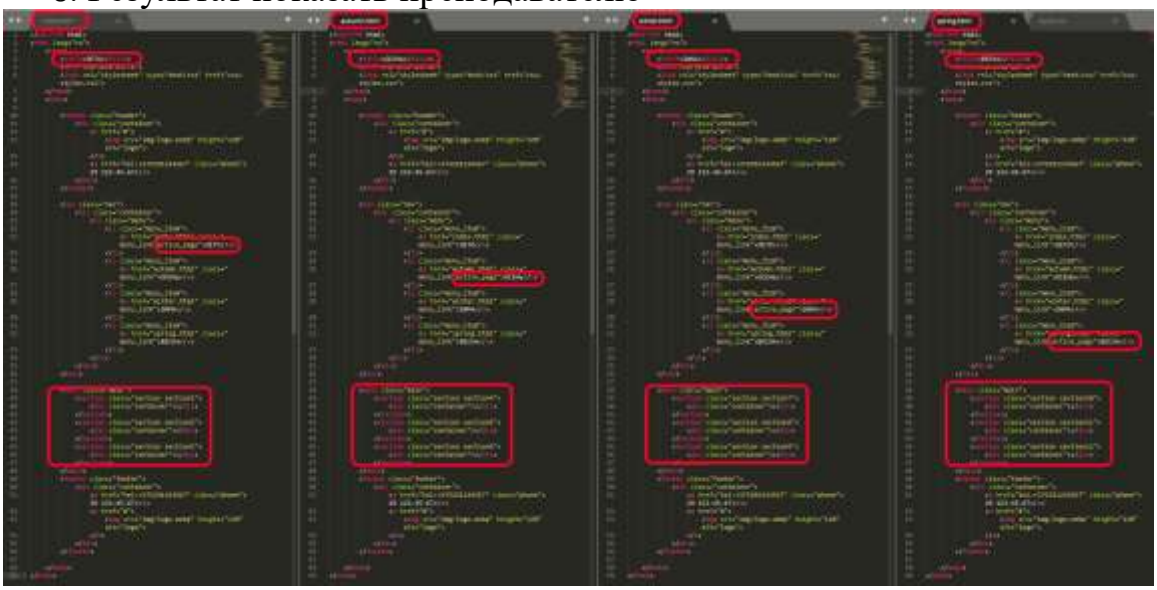


Рисунок 8 – Пример кода всех страниц готового проекта

styles.css

```css
body { margin: 0; }
ul { margin: 0; padding: 0; }
a { text-decoration: none; }
.container {
    max-width: 1024px;
    margin: 0 auto;
    height: inherit; }
.header {
    height: 150px;
    background-color: gold;
    box-sizing: border-box; }
.header .container img, .footer .container img { border-radius: 15px; }
.header .container, .footer .container {
    padding: 0 50px;
    display: flex;
    align-items: center;
    justify-content: space-between;}
.phone {
    font-family: sans-serif;
    font-size: 24px;
    font-weight: 700;
    color: #333; }
.phone:hover { opacity: 0.7; }
.nav { height: 50px; background-color: pink; }
.menu { padding-top: 13px; }
.menu_item {    display: inline; }
.menu_link {
    font-family: arial;
    font-size: 24px;
    font-weight: 700;
    padding: 13px 20px 10px;
    color: blue; }
.menu_link:hover { background-color: yellow; color: darkblue; }
.active_page { background-color: #ffa5006e; }
.main { height: 3000px; }
.footer { height: 250px; background-color: grey; }
.section { height: 1000px; }
.section1  { background: url(../img/summer1.webp) no-repeat fixed center; background-size: cover; }
.section2  { background: url(../img/summer2.webp) no-repeat fixed center; background-size: cover; }
.section3  { background: url(../img/summer3.webp) no-repeat fixed center; background-size: cover; }
.section4  { background: url(../img/autumn1.webp) no-repeat fixed center; background-size: cover; }
.section5  { background: url(../img/autumn2.webp) no-repeat fixed center; background-size: cover; }
.section6  { background: url(../img/autumn3.webp) no-repeat fixed center; background-size: cover; }
.section7  { background: url(../img/winter1.webp) no-repeat fixed center; background-size: cover; }
.section8  { background: url(../img/winter2.webp) no-repeat fixed center; background-size: cover; }
.section9  { background: url(../img/winter3.webp) no-repeat fixed center; background-size: cover; }
.section10 { background: url(../img/spring1.webp) no-repeat fixed center; background-size: cover; }
.section11 { background: url(../img/spring2.webp) no-repeat fixed center; background-size: cover; }
.section12 { background: url(../img/spring3.webp) no-repeat fixed center; background-size: cover; }
```

Рисунок 9 – Пример скрипта файла css готового проекта

3.7 Лабораторная работа 1. Основы проектирования сайта

Цель работы: освоение студентами способа создания WEB-сайта на платформе TILDA.CC.

Задание – создать посадочную страницу кафедры информационных технологий в культуре на основе шаблона в TILDA.CC

Методические рекомендации:

1. Зарегистрироваться на платформе TILDA.CC (рисунок 10, 11)
2. Создать новый сайт «Кафедра информационных технологий в культуре» (рисунок 12).
3. Создать первую страницу, выбрать шаблон (рисунок 13).
4. Просмотреть обучающее видео по работе на платформе (рисунок 14).
5. Отредактировать шаблон: отредактировать текст (рисунок 15), добавить изображения (рисунок 16), добавить новые блоки (рисунок 17).
6. Сохранить созданную посадочную страницу.

Рисунок 10 – Стартовая страница платформы TILDA

Рисунок 11 – Форма регистрации на платформе TILDA

Рисунок 12 – Форма создания сайта

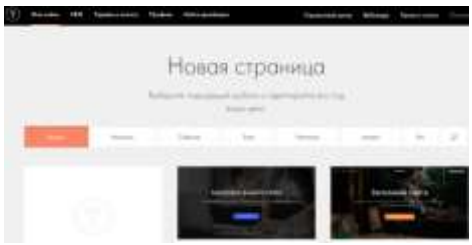


Рисунок 13 – Выбор фаблона для первой страницы сайта

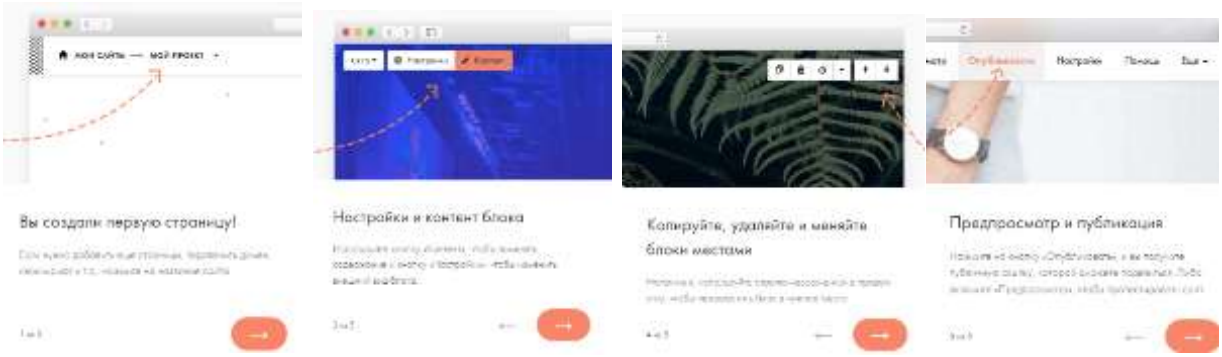


Рисунок 14 – Обучающее видео для работы на платформе TILDA

Рисунок 15 – Инструменты для редактирования текста

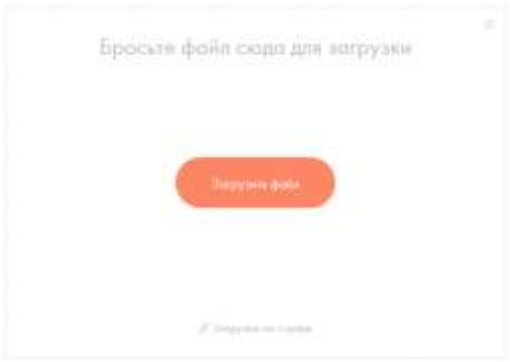


Рисунок 16 – Окно для загрузки нового изображения

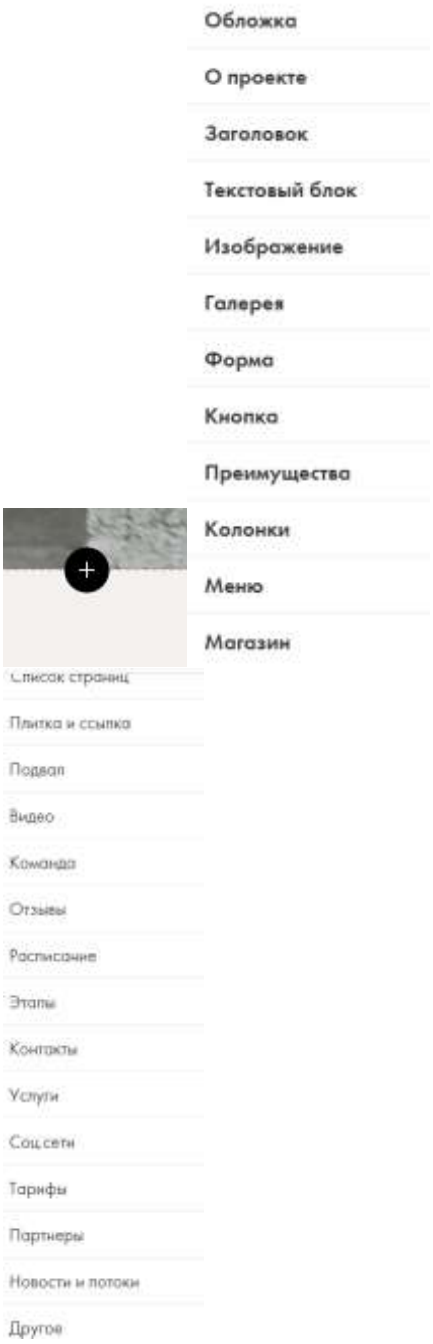


Рисунок 17 – Добавление новых блоков

3.8 Лабораторная работа 2. Язык HTML

Цель: Создание и сохранение html документа в папке проекта.

Задание: знакомство с HTML. Создание первого документа, сверстанного на языке HTML.

1. Создать в папке html-документ.

– Открыть Sublime Text, нажать File / New File.

– Новый документ, необходимо правильно переименовать и сохранить, для этого нажать File / Save As… (рисунок 18).

– В появившимся окне, в поле «Имя файла» пишем ОБЯЗАТЕЛЬНО имя «index.html».

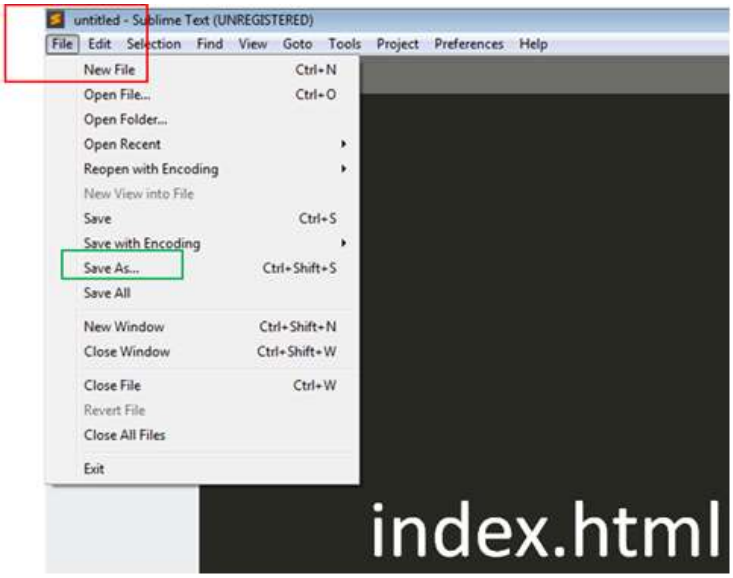


Рисунок 18 – Создание и сохранение нового файла в программе Sublime Text

2. Ознакомится со структурой HTML-документа (рисунок 19).

HTML-документ состоит из элементов. Тег и информация внутри него называется HTML-элемент (рисунок 20).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Заголовок документа</title>
  </head>
  <body>
    <h1>Заголовок</h1>
    <p>Здесь содержание документа...</p>
  </body>
</html>
```

Рисунок 19 – Структура документа HTML

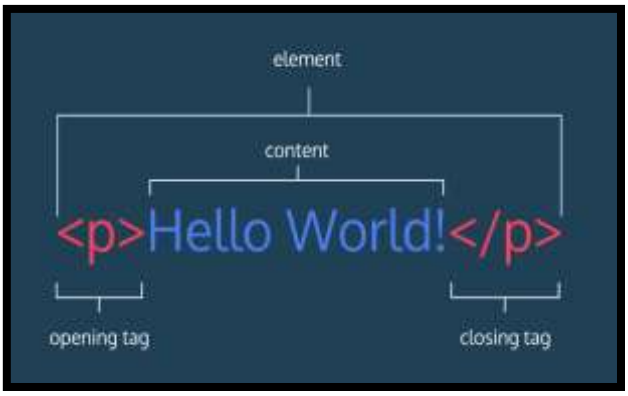


Рисунок 20 – Структура HTML-элемента

Рассмотрим основную структуру HTML-документа, без которой не обходится ни один сайт:

<!DOCTYPE html> - определяет тип документа и версию HTML.

Тег <html> определяет начало HTML-файла, внутри него хранится заголовок (<head>) и тело документа (<body>).

Заголовок документа, как еще называют блок <head>, может содержать текст и теги, но содержимое этого раздела не показывается напрямую на странице, за исключением <title>(отображается на вкладке).

Тело документа <body> предназначено для размещения тегов и содержательной части веб-страницы, все, что будет отображено на сайте, хранится внутри этого тега.

3. Ознакомиться с основными тегами HTML-документа (рисунок 21).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Заголовок документа</title>
  </head>
  <body>
    <h1>Заголовок 1</h1>
    <h2>Заголовок 2</h2>
    <h3>Заголовок 3</h3>
    <h4>Заголовок 4</h4>
    <h5>Заголовок 5</h5>
    <h6>Заголовок 6</h6>

    <p> Hello World!</p>

  </body>
</html>
```

Рисунок 21 – Теги заголовков и параграфа в HTML-документе

– <h1></h1>.....<h6></h6>– теги заголовков, от самого большого к самому маленькому.

– –параграф.

– – расставление акцентов в тексте путём выделения его фрагментов полужирным начертанием.

– – выделение текста курсивом.

–
 – обрыв строки.

– <!--...--> – тег для добавления комментариев в документ. Помещённые внутри него теги не интерпретируются браузером. Создаем документ, используя указанные теги.

4. Внести информацию по образцу в свой документ.

5. Сохранить файл.

3.9 Лабораторная работа 3. Язык HTML

Цель: научиться описывать теги вставки изображения, ссылки и списки.

Задание:

1. Открыть свой файл «index.html» в программе Sublime Text.
2. Создать разные по видам и уровням списки.
3. Вставить в документ ссылки (внутри страницы и внешние).
4. Вставить изображения.
5. Сохранить файл.

1. Списки

Существует два вида списков – упорядоченные и неупорядоченные.

Создание маркированного списка (рисунок 22) в HTML осуществляется с помощью блочного элемента списка <ul>. Каждый отдельный пункт в списке размечается с помощью элемента <li>. Именно тег <li> (сокращенно от «list» – список) указывает на то, что элемент является частью списка, тег <ul> (сокращенно от «unordered list» – неупорядоченный список) указывает на неупорядоченный список.

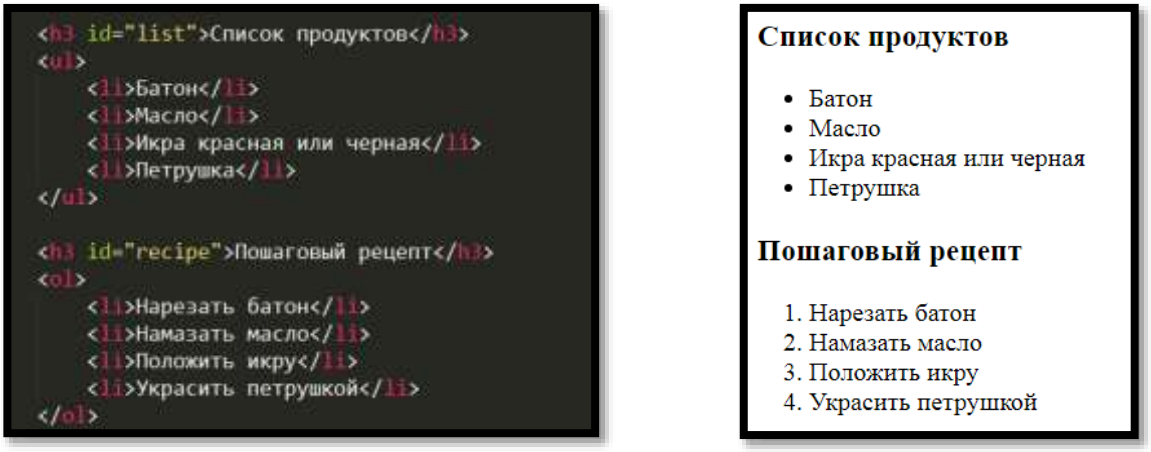


Рисунок 22 – Маркированный и нумерованный (упорядоченный) списки

Нумерованный или упорядоченный список (рисунок 22) элементов <ol> очень похож на маркированный список, отдельные пункты списка создаются таким же образом. Основным различием между списками является то, что для упорядоченного списка важен порядок представления пунктов. Поскольку порядок имеет значение, вместо точки в качестве маркера по умолчанию в нумерованном списке применяются номера.

2. Атрибуты

Открывающие теги могут содержать атрибуты (рисунок 23). С их помощью можно менять свойства объекта, которому соответствует тег.

Атрибуты записываются внутри открывающего тега через пробел в виде ключевого слова (имя), знака “=” и параметра (значения атрибута).

Порядок следования атрибутов в теге не важен. Разделителем атрибутов также является пробел.

Рисунок 23 – Пример написания тега с атрибутом

3. Изображение

HTML-изображения добавляются на веб-страницы с помощью тега <img> (самозакрывающийся) (рисунок 24).

Тег <img> имеет обязательный атрибут src, значением которого является абсолютный или относительный путь к изображению.

Атрибут ALT - это важный элемент для всех картинок на странице. Он помогает пользователям узнать, что изображено на картинке, если она не

отображается по какой-либо причине. Если картинка используется просто как элемент дизайна, тогда все равно добавляйте пустой атрибут - alt=" ". Отсутствие данного атрибута считается плохой практикой.

```
<img src="https://st.vkuso.ru/data/cache/thumb/
e4/7bd86b0380e94e4_600x400.jpg" alt="sandwich"/>
```

Рисунок 24 – Пример вставки изображения с атрибутом «alt»

4. Ссылки

Атрибут «href» определяет URL (Universal Resource Locator, универсальный указатель ресурса), иными словами, адрес документа, на который следует перейти, а содержимое контейнера <a> является ссылкой (рисунок 25). Текст, расположенный между тегами <a> и , по умолчанию становится синего цвета и подчеркивается.

```
<p><a href="https://www.youtube.com/watch?v=eMd3nRezhcA"
target="_blank">Рецепт</a></p>
```

Рисунок 25 – Пример создания внешней ссылки на текст

Заметим, что <a> является инлайновым элементом, поэтому ссылка будет отображаться справа от картинки, однако если мы поместим его в блочный тег <p>, то ссылка отобразится снизу от картинки.

По умолчанию, при переходе по ссылке документ открывается в текущем окне. При необходимости, это условие может быть изменено атрибутом target со значением «_blank».

Ссылка внутри страницы (якорь)

Случается, что необходимо сделать ссылку от начала страницы на ее конец, или с оглавления на определенную главу, а листать всю страницу не хочется, тогда можно создать ссылки внутри страницы (рисунок 26).

ШАГ 1. Добавим после оглавления (после <h1>..</h1>) меню в виде списка и укажем ссылку.

ШАГ 2. К каждому заголовку добавляем уникальный идентификатор через атрибут id.

```
<ul>
    <li><a href="#list">Список продуктов</a></li>
    <li><a href="#recipe">Рецепт</a></li>
    <li><a href="#media">Медиа</a></li>
</ul>
```

```
<h3 id="list">Список продуктов</h3>
<ul>
    <li>Батон</li>
    <li>Масло</li>
    <li>Икра красная или черная</li>
    <li>Петрушка</li>
</ul>

<h3 id="recipe">Пошаговый рецепт</h3>
<ol>
    <li>Нарезать батон</li>
```

Рисунок 26 – Пример написания тегов для создания ссылки внутри страницы

3.10 Лабораторная работа 4. Язык HTML

Цель: научится применять встроенные в тэг стили.

Задание:

1. Создать новый html-файл в программе Sublime Text.

2. Ознакомиться и повторить пример написания встроенного стиля в свой файл (рисунок 27).

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <title>Lesson 2</title>
</head>
<body>

    <p>Hello</p>
    <p>I am a
        <span style="
            margin-left: 30px; /* отступ слева (внешний) */
            font-size: 30px; /* размер шрифта */
            color: red; /* цвета шрифта */
            font-family: arial; /* шрифт */
            font-weight: 700; /* вес шрифта (жирность) */
            background-color: gold; /* цвет фона */
            border: 3px solid green; /* рамка (толщина тип цвет) */
            padding: 20px;"> <!-- отступ со всех сторон (внутренний) -->

        PROGER</span>
    </p>

</body>
</html>
```

Рисунок 27 – Пример написания встроенного стиля

Тэги <p> и работают разным способом:

– Тэг «р» - блочный, то есть - переносит содержимое на новую строчку.

– Тэг «спан» - строчный, структурно ничего не меняет в содержимом, но может отделить слово или фразу для применения стилей, которые пишутся в первом тэге.

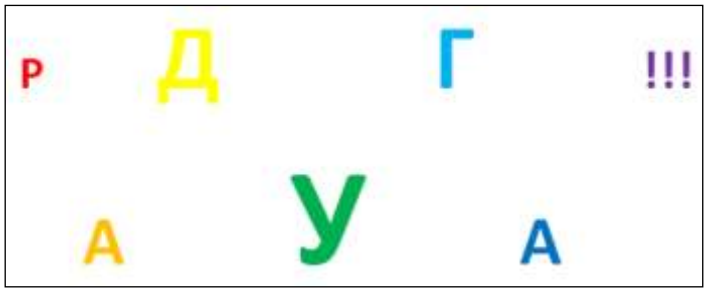
– Комментарии указывают, что делают атрибуты стиля.

3. Используя полученный пример выполнить все задания, представленные на рисунке 28.

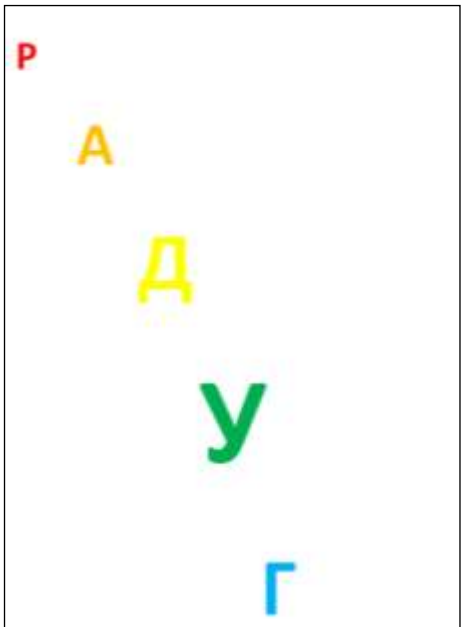
4. Вставить изображения.

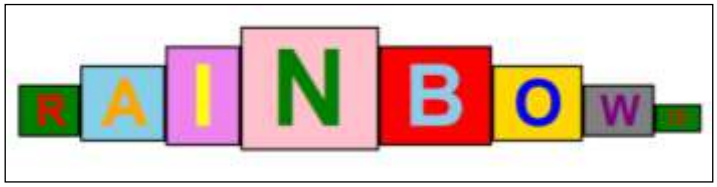
5. Сохранить файл.

28.1



28.2





28.4

28.3

Рисунок 28 – Самостоятельное задание по применению встроенного стиля

3.11 Лабораторная работа 5. Язык HTML

Цель: рассмотреть возможности работы с таблицами.

Задание:

1. Создать новый html-файл в программе Sublime Text.
2. Ознакомиться с особенностями применения таблиц.
3. Повторить пример работы с таблицей (рисунок 29-31).

Прежде всего, содержимое любой таблицы заключается между двумя тегами: <table></table> (рисунок 29).

Для наглядности, поместим внутри тега <table> атрибут border=”2”, чтобы видеть границы.

Самым маленьким контейнером в таблице является ячейка, она создается элементом <td> ('td' - сокращение от 'table data').

Чтобы новые ячейки перешли на вторую строку, необходимо использовать элемент <tr> ('tr' - сокращение от 'table row').

Для объединения двух и более ячеек в одну используются атрибуты «colspan» и «rowspan» тега <td> (рисунок 30). Атрибут «colspan» устанавливает число ячеек объединяемых по горизонтали (объединяет столбики, column - колонка). Аналогично работает и атрибут «rowspan», но объединяет ячейки по вертикали (объединяет строчки, row - строка).

```
<table border="2">
    <tr>
        <td>1 ячейка</td>
        <td>2 ячейка</td>
        <td>3 ячейка</td>
    </tr>
    <tr>
        <td>4 ячейка</td>
        <td>5 ячейка</td>
        <td>6 ячейка</td>
    </tr>
</table>
```

1 ячейка	2 ячейка	3 ячейка
4 ячейка	5 ячейка	6 ячейка

Рисунок 29 – Пример вставки таблицы

```
<table border="2">
    <tr>
        <td colspan="2">1 ячейка + 2 ячейка</td>
        <td rowspan="2">3 ячейка + 6 ячейка</td>
    </tr>
    <tr>
        <td>4 ячейка</td>
        <td>5 ячейка</td>
    </tr>
</table>
```

1 ячейка + 2 ячейка		3 ячейка + 6 ячейка
4 ячейка	5 ячейка	

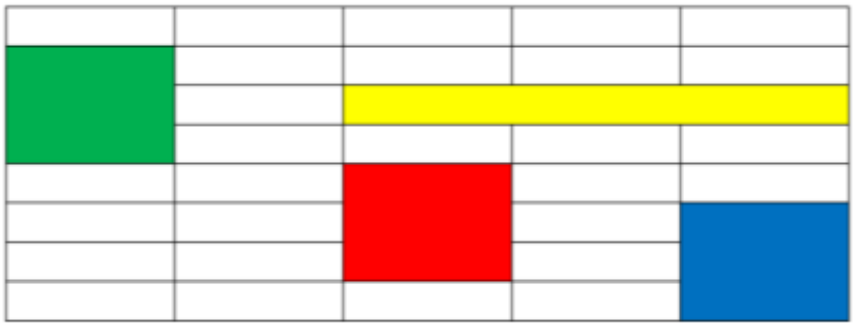
Рисунок 30 – Пример объединения ячеек

Также можно применять встроенные стили в таблицы, способы для оформления ячеек в коде (Рисунок 31).

```
<table border="1" cellspacing="0" cellpadding="0">
    <tr>
        <td colspan="2">111</td>
        <td bgcolor="#a8c3e0">222</td>
    </tr>
    <tr>
        <td width="100px" height="50px">444</td>
        <td rowspan="2">555</td>
        <td>666</td>
    </tr>
```

Рисунок 31 – Пример применения встроенного стиля для оформления ячеек

4. Используя полученный знания самостоятельно выполнить все задания, представленные на рисунке 32.



32.1

расходы	2020 год		
	январь	февраль	март
Машина	500	350	100
Квартира	50	55	Нет информации
Питание	780	600	

32.2

Название	Население млн. чел.		Плотность чел. на кв. км.		Площадь, млн. кв. км.
	1970 г	1989 г	1970 г	1989 г	
Австралия и Океания	19	26	2	3	8,5
Африка	361	628	12	21	30,3
Европа	642	701	61	67	10,5
Южная Америка	190	291	11	16	17,8
Северная и Центральная Америка	320	422	13	17	24,3
Азия	2161	3133	49	71	44,4
Весь мир	3693	5201	27	38	135,8

32.3

Рисунок 32 – Самостоятельное задание для применения таблиц

5. Сохранить файл.

3.12 Лабораторная работа 6. Язык HTML

Цель: получить практические навыки по работе с формами.

Задание:

1. Создать новый html-файл в программе Sublime Text.

2. Ознакомиться и повторить пример написания кода для вставки формы (рисунок 33).

Основу любой формы составляет элемент <form>...</form>.

Если мы хотим создать поле ввода в форме, нам понадобится <input>элемент. Когда мы создаем <input> элемент с «type="text"», он отображает текстовое поле, в котором пользователи могут вводить информацию «name=" "» – определяет имя формы. Без атрибута name информация не будет отправлена.

Для того, чтобы пользователь правильно понял что требуется ввести, используются метки - <label>. Чтобы связать <label>и <input>, <input> необходим id атрибут, а <label> - атрибут for. Еще одним преимуществом использования <label>элемента является то, что при щелчке по этому элементу соответствующий элемент <input> выделяется/выбирается.

Для отправки введенных данных на сервер используем кнопку (submit).

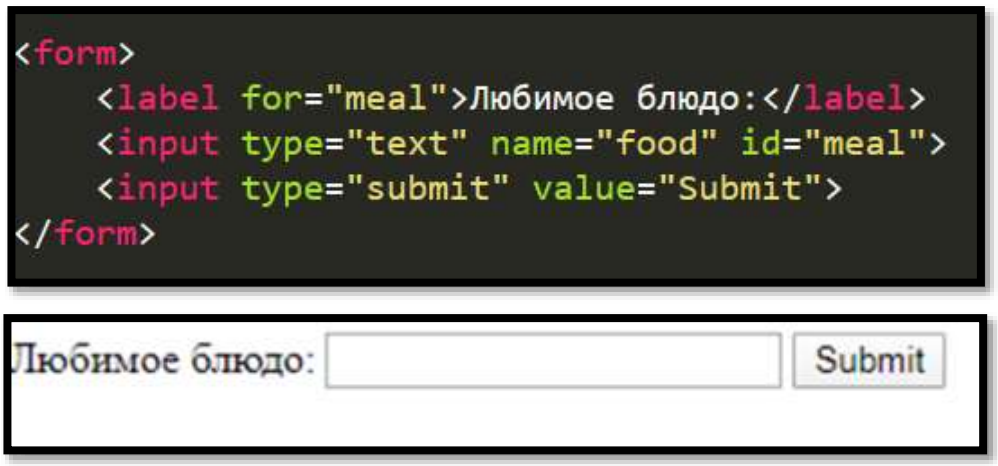


Рисунок 33 – Пример написания кода для вставки формы
и её отображение в браузере

4. Переписать пример кода в свой файл, используя пример на рисунке 34.

5. Исходя из полученных знаний выполнить самостоятельно задания по составлению разных типов формы регистрации (рисунок 35-36).

6. Сохранить файл.

```html
<!DOCTYPE html>
<html lang="ru">
<head>
    <title>Forms</title>
    <meta charset="utf-8">
</head>
<body>
    <form>
        <label>Name:</label><br>
        <input type="text" name="name"><br>
        <label>Password:</label><br>
        <input type="password" name="pass"><br>
        <hr>
        <input type="radio" name="sex">M<br>
        <input type="radio" name="sex">X<br>
        <input type="radio" name="sex">?<br>
        <hr>
        <input type="checkbox" name="games">Games<br>
        <input type="checkbox" name="music">Music<br>
        <input type="checkbox" name="sport">Sport<br>
        <input type="checkbox" name="coding">Coding<br>
        <input type="checkbox" name="chess">Chess<br>
        <hr>
        <input type="button" value="BUTTON">
        <input type="submit" value="SUBMIT">
        <input type="reset" value="RESET">
        <hr>
        <textarea rows="10" cols="45" name="text"></textarea>
        <br>
        <select>
            <option value="2000">2000</option>
            <option value="2001">2001</option>
            <option value="2002">2002</option>
            <option value="2003">2003</option>
            <option value="2004">2004</option>
            <option value="2005">2005</option>
        </select>

    </form>

</body>
</html>
```

Рисунок 34 – Пример написания кода для вставки формы регистрации

Индивидуальное задание

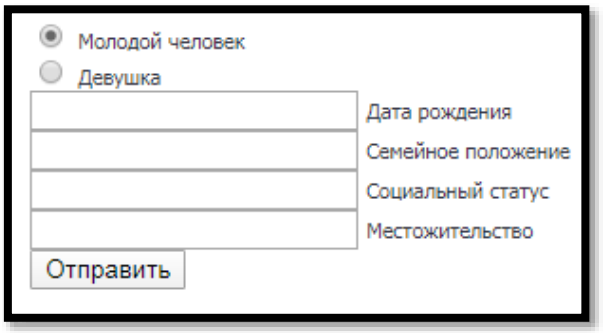


Рисунок 35– Форма регистрации 1

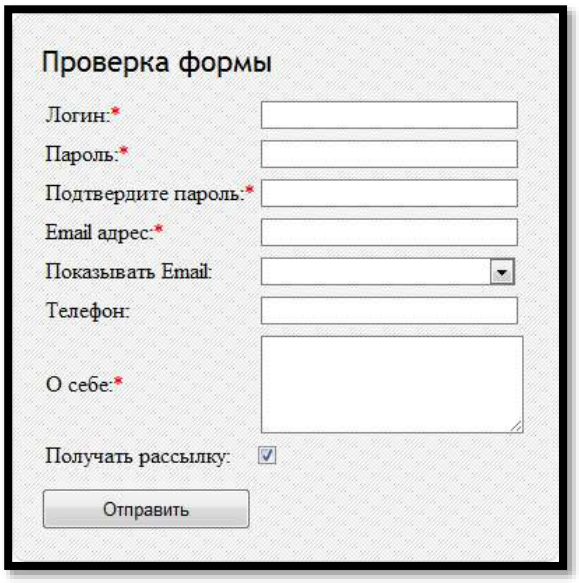


Рисунок 36– Форма регистрации 2

3.13 Лабораторная работа 7. Технология CSS

Цель: научиться работать с применением внешнего файла CSS, который позволяет прикреплять стиль к структурированным документам.

Задание:

1. Открыть любой текстовый html-файл в программе Sublime Text.
2. Добавить в <head>…</head> тег с метаданными, для применения кодировки (рисунок 37).
3. Создать новый файл, сохранить под именем «stylesheet.css».
4. В файле ‘html’ добавить в <head>…</head> тег <link> (рисунок 38).

Рисунок 37– Назначение метаданных кодировки

Рисунок 38– Применение ссылки для связи html-файла с файлом css

Необходимо связать два файла (index.html и style.css), в противном случае файл HTML не сможет найти код CSS, и стили не будут применены. Для этого используем тег <link> в теге <head>…</head>.

5. Ознакомиться с типами селекторов, особенностями их оформления, специфичностью, с применением цепочки селекторов (рисунок 39-41).

Селекторы бывают 3-х типов: теги (рисунок 39), классы (рисунок 40), уникальные идентификаторы (id) (рисунок 41).

Рисунок 39 – Селектор «тег»

Рисунок 40 – Селектор «класс»

Рисунок 41 – Селектор «ID»

Специфичность – это способ, с помощью которого браузеры определяют, какие значения свойств CSS наиболее соответствуют элементу и, следовательно, будут применены. Специфичность основана на правилах соответствия, состоящих из селекторов CSS различных типов. В следующем списке типы селекторов расположены по возрастанию специфичности (приоритета):

– селекторы типов элементов (теги, например, h1) и псевдоэлементов (например: .before).

– селекторы классов (классы, например, .example), селекторы атрибутов (например, [type="radio"]) и псевдоклассов (например, :hover).

– селекторы идентификаторов ID (например, #example).

На рисунке 42, исходя из специфичности, к элементу применится свойство «color: firebrick;».

Несколько селекторов. В CSS можно группировать селекторы. Допустим, нужно сделать цвет заголовка и абзаца одного цвета – красным (рисунок 43). Этот код браузер понимает так: «применить красный цвет текста ко всем абзацам и заголовкам первого уровня». Два разных селектора ОБЯЗАТЕЛЬНО разделить запятой.

Цепочка селекторов - к одному элементу можно применить несколько классов и идентификаторов. Данный стиль (рисунок 44) будет применен ТОЛЬКО к элементу с тегом <p> И классом class=”red-text”. Заметим, что в данном случае селекторы НЕ разделяют запятой.

```
<h1 class="headline">Breaking News</h1>

h1 {
  color: red;
}

.headline {
  color: firebrick;
}
```

Рисунок 42 – Пример назначения специфичности стиля

```
p, h1 {
    color: #FF0000;
}
```

Рисунок 43 – Пример назначения стиля нескольким селекторам

```
html
    <p class="red-text">Красный параграф</p>

css
    p.red-text {
        color: #FF0000;
    }
```

Рисунок 44 – Пример применения цепочки селекторов

6. Исходя из полученных знаний выполнить самостоятельно задания по оформлению веб-страницы (рисунок 45), с назначением внешнего файла стилей с разными типами селекторов.

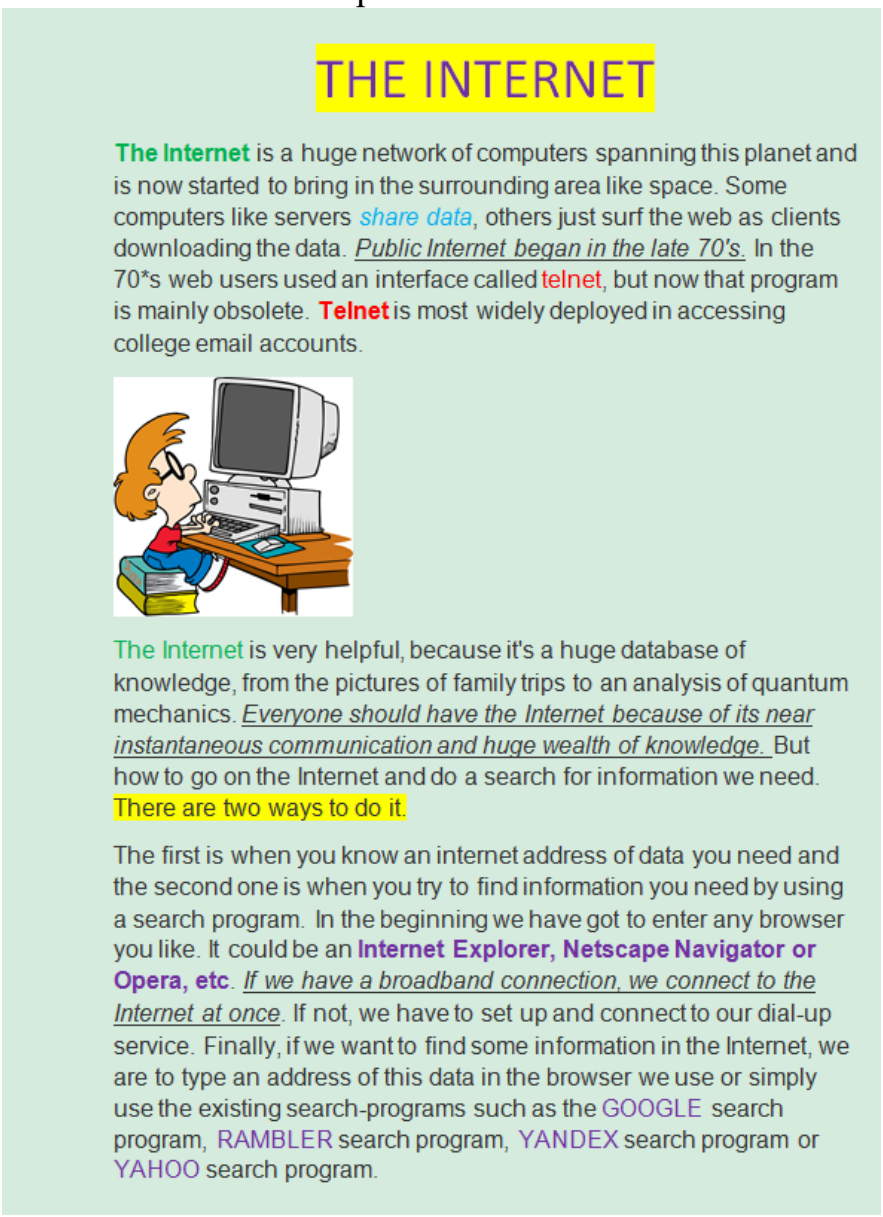


Рисунок 45 – Пример итого результата по выполнению самостоятельного задания

7. Сохранить файл.

3.14 Лабораторная работа 8. Технология CSS

Цель: ознакомиться с особенностями применения блочной модели.

Задание:

1. Познакомиться с вёрсткой основных зон страниц сайта.

Блочная модель

Есть два типа HTML-элементов: строчные и блочные. Одной из особенностей блочных элементов является то, что они занимают всю площадь контейнера, в котором находятся. Если вы не указали иное с помощью CSS, они растягиваются, чтобы занять максимум доступного места, сдвигая другие элементы под ними (рисунок 46). Кроме того, блочные элементы могут содержать другие блочные или строчные элементы и автоматически регулируют свою высоту, чтобы уместить своё содержимое. В их число входят заголовки, контейнеры, списки и другие элементы.

Рисунок 46 – Пример блочной верстки веб-страницы

Блок состоит из нескольких слоёв, которыми можно независимо управлять с помощью CSS. Это позволяет разместить элементы относительно друг друга и оформить их разными способами (рисунок 47).

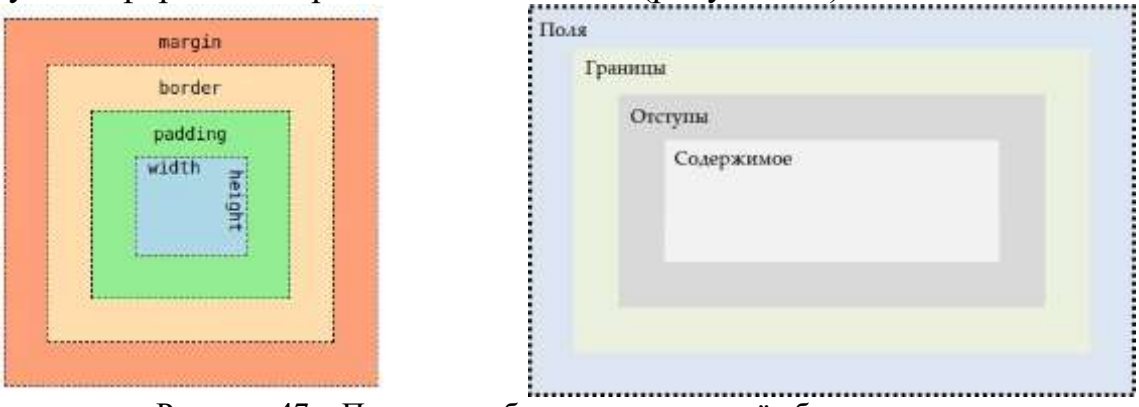


Рисунок 47 – Пример отображения всех слоёв блока

– Ширина (height) – ширина площади содержимого элемента. Для блочных элементов значение по умолчанию равно 100%. У строчных элементов ширина зависит от содержимого.

– Высота (width) – определяет высоту элемента. Как правило, она зависит от внутреннего содержимого, но при желании можно указать конкретную высоту. Опять же, это работает только с блочными элементами.

– Границы (border) – границы есть у каждого элемента, даже если вы их не видите. У них может быть разный размер, цвет и оформление.

– Отступы (padding) – они определяют расстояние между границей элемента и его содержимым. Их можно использовать, например, для того, чтобы текст внутри элемента оставался читаемым.

– Поля (margin) – они определяют расстояние между границей элемента и тем, что его окружает.

2. Создать новые html- и css –файлы в своей папке (рисунок 48).

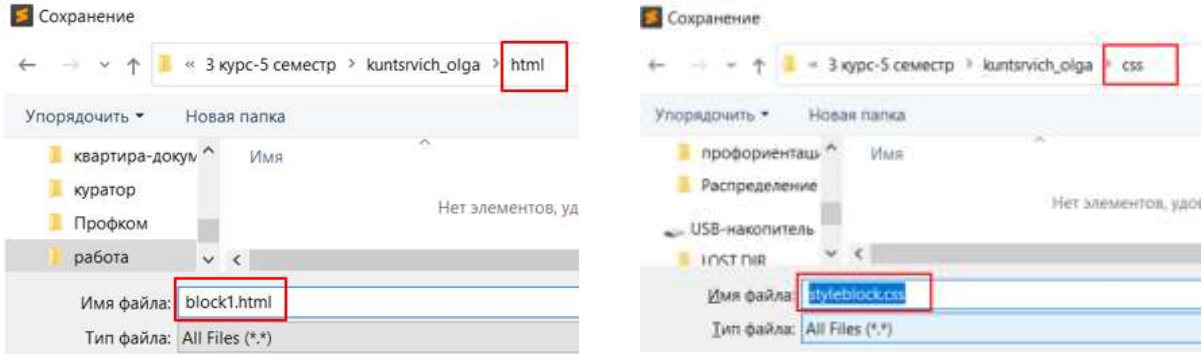


Рисунок 48 – Пример создания новых файлов

3. В файле html прописать: структуру документа, название страницы, метаданные, присоединить файл css к новой странице.

В основной части документа <body> ввести новый блочный элемент , назначить ему класс «.exemple».

4. В файле css ввести параметры стиля класса «.exemple» (рисунок 49).

Рисунок 49 – Пример параметров класса для блока и его отображение в браузере

5. Ознакомиться с примерами редактирования блоков. Повторить по примеру в своем файле.

5.1. Расположить элемент по центру – «Auto» – указывает, что размер отступов будет автоматически рассчитан браузером (рисунок 50).

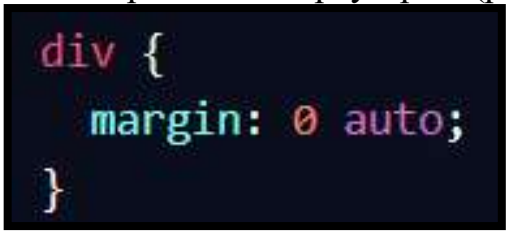


Рисунок 50 – Центрирование элемента div

Это наиболее распространенный способ использования auto для отступов. Если мы задаем auto для левого и правого отступов одного элемента, они равномерно займут все доступное в контейнере по горизонтали пространство. Таким образом элемент расположится по центру.

Это работает только для горизонтальных отступов. Но не будет работать для плавающих и строчных элементов. А также для абсолютно и фиксировано позиционированных элементов.

5.2 Разные значения для разных сторон (для границ, отступов, полей)

Для разных сторон можно установить свои значения с помощью «дополненных свойств» (рисунок 51):

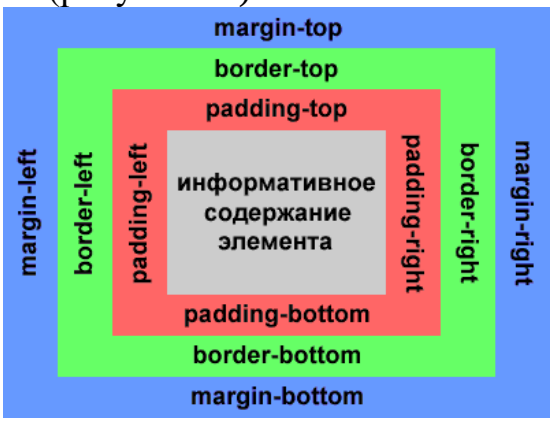


Рисунок 51 – Свойства блока для разных сторон

– Установить разные типы границ для каждой стороны с помощью свойств: border-top, border-right, border-bottom и border-left.

– Установить разные отступы: padding-top, padding-right, padding-bottom и padding-left.

– Установить разные поля: margin-top, margin-right, margin-bottom и margin-left.

особенности назначения отдельных значений для свойств блока:

– Установить разные отступы (или границы, или поля) через 1 строку в коде, на примере свойства «padding» с 4 значениями (рисунок 52):

– Первое значение - top

– Второе значение - right

– Третье значение - bottom

– Четвертое значение – left

Т.е. от верхнего значения по часовой стрелке.

Рисунок 52 – Пример применения разных значений для одного свойства блока

– Если нужно, чтобы сверху и снизу был один отступ, а слева и справа – другой, тогда можно использовать свойство padding, но с 2 значениями (рисунок 53).

– Первое значение – top и bottom

– Второе значение – right и left

```
index.html   style.css
1  .example {
2      background-color: #68ca8f;
3      width: 300px;
4      height: 300px;
5      border: 15px solid #357d33;
6      padding: 16px 200px;
7      margin: 25px;
8  }
```

Рисунок 53 – Пример применения повторяющихся значений для одного свойства

5.3 Box-sizing

Когда речь заходит о размерах элементов, стоит принять во внимание, что учитываются все части блочной модели. Например, если вы установили ширину равной 200px, это значение считается только для области с содержимым.

Любые границы, отступы и поля также вносят свой вклад в горизонтальный размер элемента. То же касается и высоты. В нашем случае также учитываются все слои блочной модели. Есть способы обхода такого поведения вроде установки свойства элемента box-sizing равным border-box. В этом случае ширина будет включать в себя границы и всё содержимое подстроит свои размеры, чтобы уместиться в этих рамках (рисунок 54).

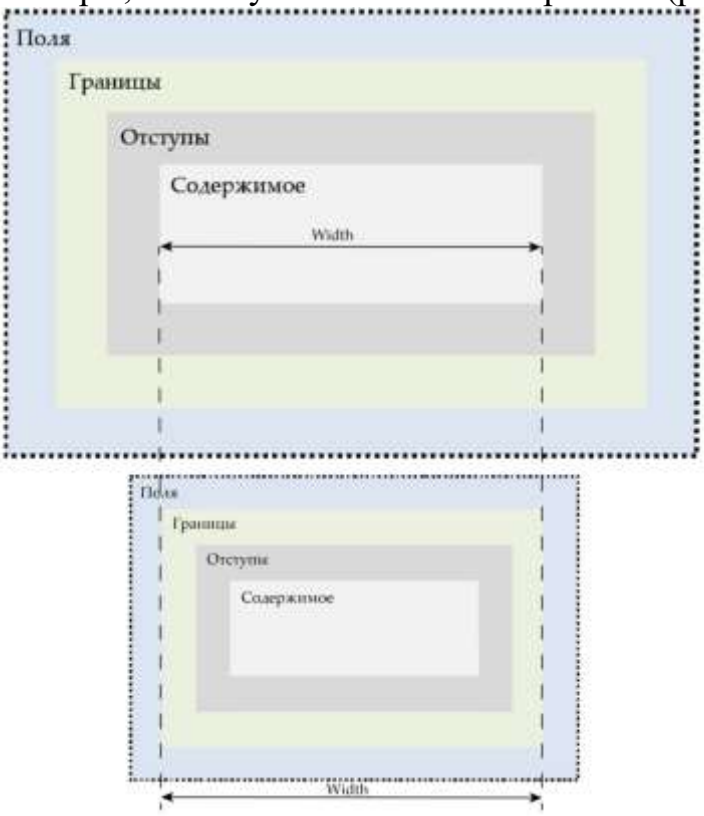


Рисунок 54 – Пример назначения свойства box-sizing

5.4 Min/Max

Поскольку веб-страницу можно просматривать с помощью дисплеев с различным размером экрана, содержимое веб-страницы может пострадать от этих изменений размера. Чтобы избежать этой проблемы, CSS предлагает два свойства, которые могут ограничивать, насколько узким или широким может быть размер блока элемента (рисунок 55):

– min-width - это свойство обеспечивает минимальную ширину поля элемента.

– max-width - это свойство обеспечивает максимальную ширину поля элемента.

Также можно ограничить минимальную и максимальную высоту элемента:

– min-height - это свойство обеспечивает минимальную высоту для поля элемента.

– max-height - это свойство обеспечивает максимальную высоту поля элемента.

5.5 Сброс настроек

Чтобы обнулить отступы и поля у всех элементов на странице, используйте стилевые свойства margin и padding с нулевыми значениями, добавляя их к универсальному селектору «*» (рисунок 56).

```
p {
  min-width: 300px;
  max-width: 600px;
}
```

Рисунок 55 – Пример ограничения по минимальной и максимальной ширине

```
* {
  margin: 0;
  padding: 0;
}
```

Рисунок 56 – Сброс настроек

6. Исходя из полученных знаний выполнить самостоятельно задания по оформлению веб-страницы (рисунок 57).

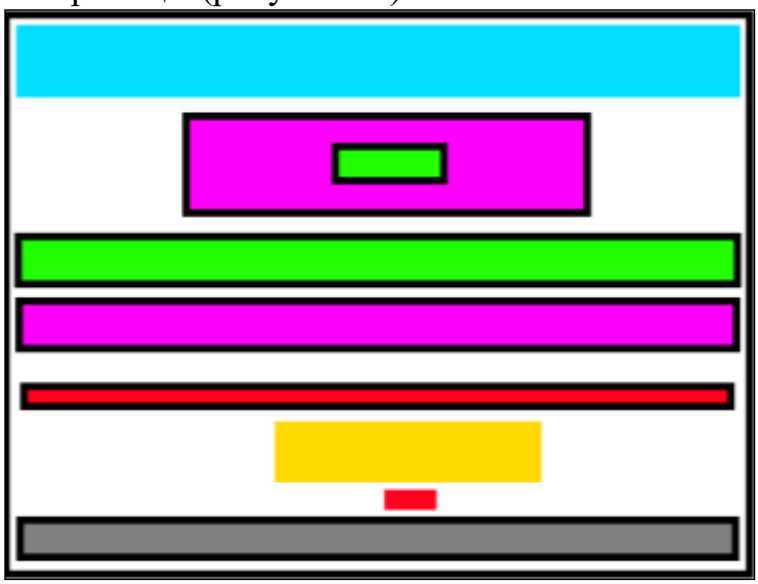


Рисунок 57 – Образец для выполнения самостоятельного задания

7. Сохранить файл.

3.15 Лабораторная работа 9. Технология CSS

Цель: научиться работать с позиционированием содержимого.

Задание:

1. Изучить основные команды позиционирования содержимого сайта при использовании блочной модели.

Свойство – *Position* (рисунок 58)

– Static – иначе называется «без позиционирования». В явном виде задается только если надо переопределить другое правило CSS.

– Relative – сдвигает элемент относительно текущего места. Противоположные границы left/right (top/bottom) одновременно указать нельзя. Окружающие элементы ведут себя так, как будто элемент не сдвигался.

– Absolute –визуально переносит элемент на новое место, которое вычисляется по координатам left/top/right/bottom относительно ближайшего позиционированного родителя. Если такого родителя нет, то им считается окно брайзера. Ширина элемента по умолчанию устанавливается по содержимому. Можно указать противоположные границы left/right (top/bottom). Элемент растянется. Окружающие элементы заполняют освободившееся место.

– Fixed – подвид абсолютного позиционирования, при котором элемент привязывается к координатам окна, а не документа. При прокрутке он остаётся на том же месте.

– Z-index контролирует как далеко «назад» или как далеко «вперед» элемент должен быть на веб-странице, когда элементы перекрывают друг друга. Как слои в фотошопе. z-index принимает целые значения. В зависимости от своих значений, целые числа указывают браузеру порядок, в котором элементы должны отображаться на веб-странице. Это работает только для горизонтальных отступов. Но не будет работать для плавающих и строчных элементов. А также для абсолютно и фиксировано позиционированных элементов.

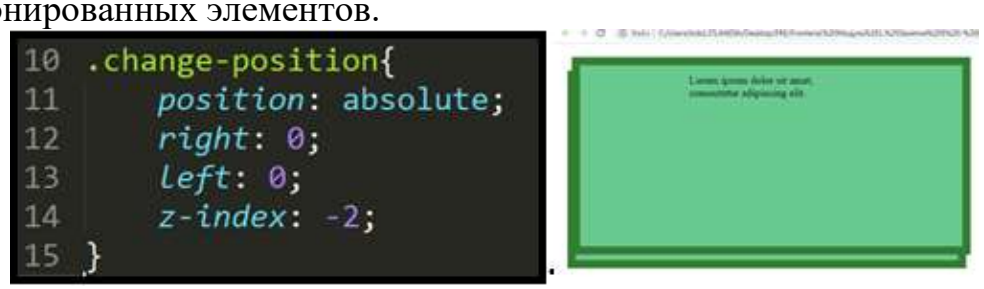


Рисунок 58 – Пример назначения свойства Position-Absolute, Z-index

Свойство *display* имеет много разных значений. Обычно, используются только три из них: none, inline и block, потому что когда-то браузеры другие не поддерживали. Но после ухода IE7-, стало возможным использовать и другие значения тоже. Рассмотрим здесь весь список.

– Значение none – самое простое значение. Элемент не показывается, вообще. Как будто его и нет.

– Значение block – блочные элементы располагаются один над другим, вертикально (если нет особых свойств позиционирования, например float). Блок стремится расшириться на всю доступную ширину. Можно указать ширину и высоту явно. Это значение display многие элементы имеют по умолчанию: , заголовок <h1>, параграф <p> (блочные элементы). Блоки прилегают друг к другу вплотную, если у них нет margin.

– Значение inline– элементы располагаются на той же строке, последовательно. Ширина и высота элемента определяются по содержимому. Поменять их нельзя. Например, инлайновые элементы по умолчанию: , <a>.

Содержимое инлайн-элемента может переноситься на другую строку. При этом каждая строка в смысле отображения является отдельным прямоугольником («line box»). Так что инлайн-элемент состоит из объединения прямоугольников, но в целом, в отличие от блока, прямоугольником не является. Больше информации https://learn.javascript.ru/display

– Значение inline-block – означает элемент, который продолжает находиться в строке (inline), но при этом может иметь важные свойства блока: 1) располагается в строке; 2) размер устанавливается по содержимому; 3) элемент всегда прямоугольный; 4) работают свойства width/height.

Значение inline-block используют, чтобы отобразить в одну строку блочные элементы, в том числе разных размеров (свойство vertical-align позволяет выровнять такие элементы внутри внешнего блока) (рисунок 59).

Дополнительная информация https://learn.javascript.ru/display

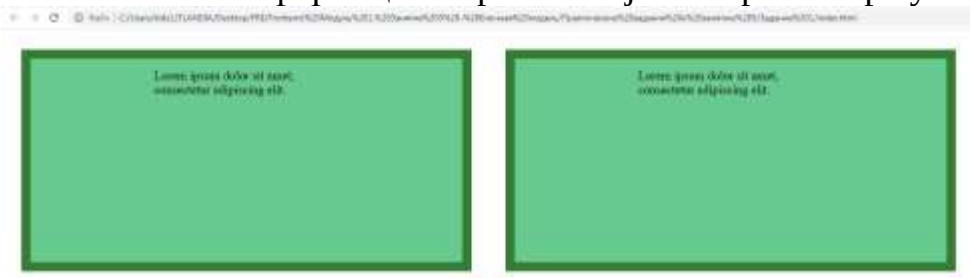


Рисунок 59 – Пример отображения в браузере свойства inline-block

Свойство *float* – элемент позиционируется как обычно, а затем вынимается из документа потока и сдвигается влево (для left) или вправо (для right) до того, как коснётся либо границы родителя, либо другого элемента с float. Если пространства по горизонтали не хватает для того, чтобы вместить элемент, то он сдвигается вниз до тех пор, пока не начнёт помещаться. Другие непозиционированные блочные элементы без float ведут себя так, как будто элемента с float нет, так как он убран из потока. Строки (inline-элементы), напротив, «знают» о float и обтекают элемент по сторонам.

– Элемент при наличии float получает display:block. То есть, указав элементу, у которого display:inline свойство float: left/right, мы автоматически сделаем его блочным. В частности, для него будут работать width/height.

– Исключением являются некоторые редкие display наподобие inline-table и run-in.

– Ширина float-блока определяется по содержимому.

– Вертикальные отступы margin элементов с float не сливаются с отступами соседей, в отличие от обычных блочных элементов.

Значения, которые можно назначить для элемента со свойством float:

– None - Отсутствие обтекания. Значение по умолчанию.

– Left - Элемент перемещается влево, содержимое обтекает плавающий блок по правому краю.

– Right - Элемент перемещается вправо, содержимое обтекает плавающий блок по левому краю.

– Inherit - Наследует значение свойства от родительского элемента.

Больше информации: https://learn.javascript.ru/float

Свойство *clear* CSS указывает, может ли элемент быть рядом с плавающими floating элементами, которые предшествуют ему или должны быть перемещены вниз (очищены) под ними. Свойство clear применяется как к плавающим, так и к неплавающим элементам.

Когда несколько плавающих элементов имеют разную высоту, это может повлиять на их макет на странице. В частности, элементы могут «сталкиваться» друг с другом и не позволять другим элементам, правильно перемещаться влево или вправо. Clear определяет, как элементы должны вести себя, когда они сталкиваются друг с другом на странице.

Больше информации:

https://learn.javascript.ru/float

https://developer.mozilla.org/ru/docs/Web/CSS/clear

2. Выполнить самостоятельно макеты страниц по требованиям:

Рисунок 60.1 – Макет 1

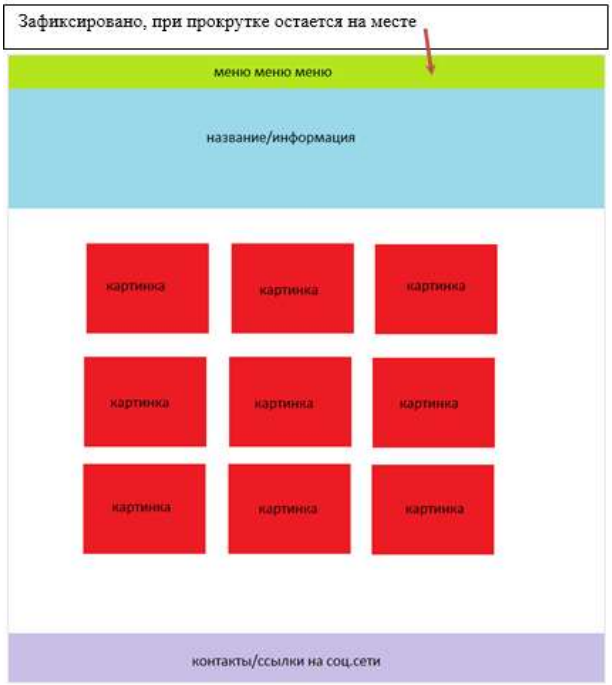


Рисунок 60.2 – Макет 2

Рисунок 60 – Пример выполнения задания

Макет 1 (рисунок 60.1):

– макет страницы состоит из 6 элементов по 3 в ряд (можно использовать изображения),

– 3 элемента в верхнем ряду зафиксированы и при прокрутке страницы остаются на месте.

Макет 2 (рисунок 60.2):

– макет страницы состоит из 12 элементов: 3 горизонтальных блока и 9 блоков по центру (3 х 3),

– элемент в верхнем ряду зафиксирован и при прокрутке страницы остается на месте.

3.16 Лабораторная работа 10. Технология CSS

Цель: приобрести навыки применять стили для определения свойства текста и шрифтов на странице сайта.

Задание 1: Ознакомиться с особенностями настройки блоков с большим объемом контента.

1. Ознакомиться с особенностями применения свойства overflow.

Свойства overflow задает поведение контейнера при переполнении:

– Visible - умолчанию, содержимое вылезает за границы блока.

– Hidden - переполняющее содержимое невидимо.

– Auto - полоса прокрутки только в случае переполнения.

– Scroll - полоса прокрутки всегда.

Можно указать поведение блока при переполнении по горизонтали в overflow-x и по вертикали – в overflow-y.

Дополнительная информация: https://learn.javascript.ru/overflow

2. Открыть файл. Повторить все параметры свойства overflow.

3. Сохранить результат.

Задание 2: Ознакомиться с особенностями настройки и подключения шрифтов при создании веб-страниц.

1. Ознакомиться с особенностями назначения шрифтов на веб-странице:

– Шрифт, указанный в таблице стилей, должен быть установлен на компьютере пользователя, чтобы этот шрифт отображался при посещении пользователем веб-страницы. Далее мы узнаем, как это обойти.

– Шрифт по умолчанию для многих браузеров - Times New Roman.

– Рекомендуется ограничить количество шрифтов, используемых на веб-странице, до 2 или 3.

– Когда имя шрифта состоит из более чем одного слова, оно должно быть заключено в двойные кавычки (в противном случае оно не будет распознано).

2. Ознакомиться с возможностью подключения дополнительных шрифтов через Google Fonts – это сервис, с помощью которого можно подключить более 700 шрифтов на свой сайт: https://fonts.google.com/

3. Подключить к своей странице шрифты:

– перейти на официальный сайт Google Fonts,

– найти в коллекции шрифтов те, которые подходят. В меню можно сузить круг, задав язык, начертание и популярность шрифта. Чтобы сервис показал шрифты с поддержкой русского языка, в пункте Languages выберите Cyrillic.

– выбрать понравившийся шрифт, после нажать Select this style, а затем кнопку сверху в правом углу (рисунок 61).

– нажать кнопку Embed, и скопировать в HTML в <head> ссылку на шрифты, а в CSS вставить полное название шрифта.

Рисунок 61 – Пример назначения индивидуального шрифта

Задание 3: Ознакомиться с особенностями применения свойств font-size и line-height

1. Ознакомиться с теоретической частью.

– font-size – размер шрифта, в частности, определяющий высоту букв.

Размер шрифта. Если сделать блок такой же высоты, как шрифт, то хвосты букв будут вылезать из-под него.

– line-height – высота строки (межстрочный интервал).

Размер строки, обычно больше размера шрифта. При установке множителем рассчитывается каждый раз относительно текущего шрифта, при установке в единицах измерения – фиксируется.

Можно одновременно задать размер, высоту строки и, собственно, сам шрифт (рисунок 62).

Дополнительная информация https://learn.javascript.ru/font-size-line-height.

```
font: italic bold 20px/1.5 Arial,sans-serif;
```

Рисунок 61 – Пример назначения параметров: размер, высота строки и шрифт

2. Применить рассмотренные параметры шрифта к своему документу.

3. Сохранить результат.

Задание 4: Самостоятельно ознакомиться с дополнительными параметрами на ресурсе Codeacademy: https://www.codecademy.com/courses/learn-css/lessons/css-typography/exercises/font-family.

3.17 Лабораторная работа 11. Технология CSS

Цель: приобрести навыки по верстке веб-страницы.

Задание 1: Создание локальной папки сайта.

1. Создайте новую папку для работы под своим именем.
2. Создайте внутри локальной папки папки для html, css и фото.
3. Создайте и сохраните начальные файлы: index.html и style.css.

Задание 2: Блок «header» (шапки сайта) (рисунок 62).

1. Ознакомиться с теоретической частью.
2. Повторить в своем документе.

– Заголовок обычно находится в верхней части сайта (или прямо под верхним меню навигации). Он часто содержит логотип или название сайта (рисунок 62.1). Заметим, что в CSS присутствует цепочка селекторов «.header h1», это значит, что данное свойство font-size применимо только к элементу с такой комбинацией селекторов (рисунок 62.2).

html

Рисунок 62.1 – Пример кода для заголовка

Рисунок 62.3 – Итоговый результат блока «header»

css

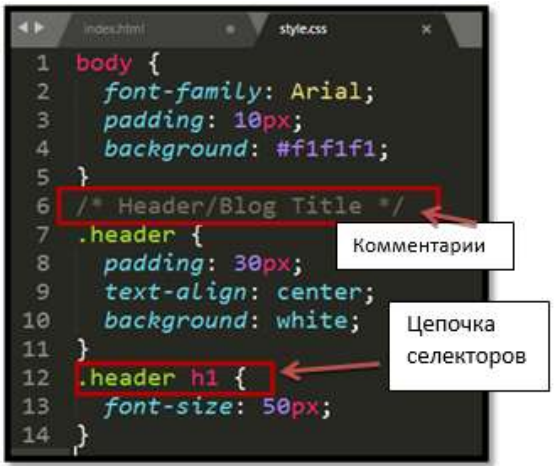


Рисунок 62.2 – Пример стиля для заголовка

Задание 3: «Панель навигации» (рисунок 63).

1. Создать блок (div) со списком ссылок (меню) - помогут посетителям перемещаться по сайту.
2. Повторить стиль для панели навигации.

html

Рисунок 63.1 – Пример кода

Рисунок 63.3 – Пример отображения «панели навигации» в браузере

css

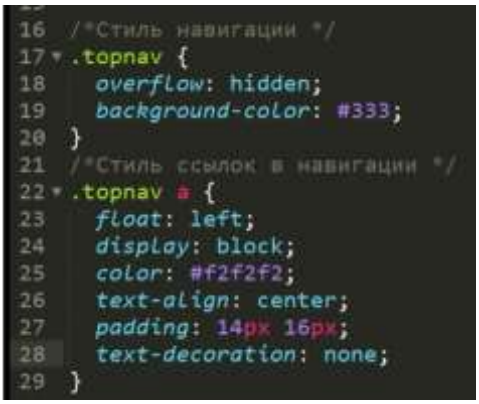


Рисунок 63.2 – Пример стиля

Задание 3: Область «Содержание» (рисунок 64).

1. Определить раскладку для содержания.

Внешний вид основного содержания зависит может быть различный, наиболее распространенная раскладка:

– 1 колонка (часто используется для мобильных браузеров);

– 2 колонки (часто используется для планшетов и ноутбуков);

– 3 колонки (используется только для настольных компьютеров).

2. Сохранить изображения в папку «img».

3. Создать блок (div). Поместить в него контент (рисунок 64):

– заголовок,

– изображение,

– общий текст,

* повторить информацию еще раз (только вставить другое фото).

** Проверить результат.

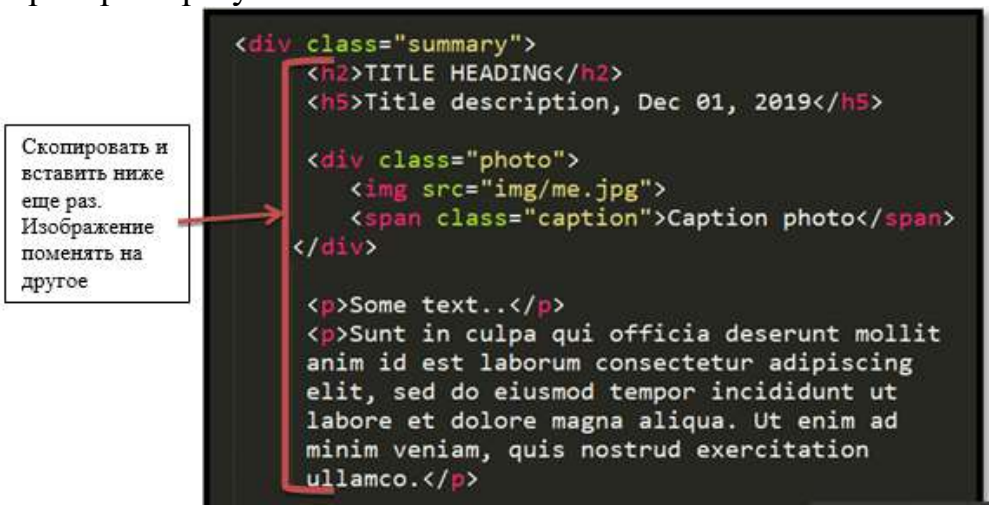


Рисунок 64 – Пример вставки блока «Содержание»

4. Исправить обтекание текста – добавить свойство «clear: left» для h2 (рисунок 65). Теперь, каждый добавленный контент, будет начинаться с новой строки (рисунок 66). Второй способ: использовать пустой блок с данным свойством где необходима очистка – .

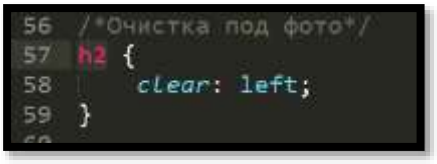


Рисунок 65 – Пример добавления свойста «clear»

Рисунок 66 – Пример отображения в браузере «без очистки» и «с очисткой»

Задание 4: Область «Footer» (подвал) (рисунок 67).

Создать новый блок «Footer» – содержит контактную информацию и авторские права. Добавить свойство «clear: left».

html

```
<div class="footer">
    <h2>Contacts</h2>
    <p>+375(25)000-00-00</p>
</div>
```

Рисунок 67.1 – Пример кода «Footer»

css

```
/* Footer */
.footer {
    clear: left;
    padding: 20px;
    text-align: center;
    background: #ddd;
    margin-top: 20px;
}
```

Рисунок 67.2 – Пример кода «Footer»

Рисунок 67.3 – Пример отображения «Footer»

Задание 5: Свойство «Box-sizing» (рисунок 68).

В файл css добавить стиль для всех объектов «*» (всего документа) свойство «box-sizing: border-box». Это обозначает, что указанная ширина относится к элементу полностью, включая border и padding.

style.css index.html

```
* {
    box-sizing: border-box;
}
```

Рисунок 68 – Свойство «box-sizing» для всех элементов

Задание 6: Самостоятельная работа

1. По примеру дополнить файлы «index.html» «style.css» (рисунок 69).

2. Дополнить страницу информацией о себе: вставить текст, картинки, ссылки (могут быть на социальные сети или ссылки внутри страницы), контакты.

3. Подписать все свойства в CSS (за что отвечает каждое?) с помощью комментариев (/**/).

4. Закрепить <heder>, чтобы при прокрутке он оставался на месте.

5. Создать кнопку «Наверх», чтобы при прокрутке можно было вернуться наверх страницы.

index.html

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Summary</title>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>

        <div class="header">
            <h1>My Website</h1>
        </div>

        <div class="topnav">
            <a href="#">Link</a>
            <a href="#">Link</a>
            <a href="#">Link</a>
            <a href="#">Link</a>
        </div>

        <div class="summary">
            <h2>TITLE HEADING</h2>
            <h5>Title description, Dec 01, 2019</h5>

            <div class="photo">
                <img src="img/me.jpg">
                <span class="caption">Caption photo</span>
            </div>

            <p>Some text..</p>
            <p>Sunt in culpa qui officia deserunt mollit anim id est
            laborum consectetur adipiscing elit, sed do eiusmod tempor
            incididunt ut labore et dolore magna aliqua. Ut enim ad
            minim veniam, quis nostrud exercitation ullamco.</p>

            <h2>TITLE HEADING</h2>

            <div class="photo">
                <img src="img/friends.jpg">
                <span class="caption">Caption photo</span>
            </div>

            <p>Some text..</p>
            <p>Sunt in culpa qui officia deserunt mollit anim id est
            laborum consectetur adipiscing elit, sed do eiusmod tempor
            incididunt ut labore et dolore magna aliqua. Ut enim ad
            minim veniam, quis nostrud exercitation ullamco.</p>
        </div>

        <div class="footer">
            <h2>Contacts</h2>
            <p>+375(25)000-00-00</p>
        </div>

    </body>
</html>
```

style.css

```css
* {
    box-sizing: border-box;
}
body {
    font-family: Arial;
    padding: 10px;
    background: #f1f1f1;
}
/* Header/Blog Title */
.header {
    padding: 30px;
    text-align: center;
    background: white;
}
.header h1 {
    font-size: 50px;
}

/*Стиль навигации */
.topnav {
   overflow: hidden;
   background-color: #333;
}
/*Стиль ссылок в навигации */
.topnav a {
    float: left;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

/*Стиль основного содержания */
.summary{
    padding: 0 10%;
}

/*Стиль фотографий */
.photo {
    width: 30%;
    padding: 1.11rem;
    border-radius: 5px;
    margin: 0 20px 1.5rem auto;
    background-color: #aaa;
    float: left;
}
.photo img {
    width: 100%;
}
.photo .caption {
    font-size: 14px;
    font-style: italic;
}

/*Очистка под фото*/
h2 {
    clear: left;
}

/* Footer */
.footer {
    clear: left;
    padding: 20px;
    text-align: center;
    background: #ddd;
    margin-top: 20px;
}
```

Рисунок 69 – Пример готовых файлов «index.html» «style.css»

3.18 Лабораторная работа 12. Графические и звуковые элементы

Цель: научиться добавлять переходы и анимацию.

Задание 1: Псевдокласс «hover»

1. Создайте новую папку для работы и 2 файла (html и css).
2. Ознакомьтесь со списком переходов и анимации (рисунок 70)

Рисунок 70 – Список переходов и анимации

Чтобы создать эффект перехода, нужно указать две вещи:

– свойство CSS, к которому вы хотите добавить эффект;

– продолжительность эффекта. *Если часть длительности не указана, переход не будет иметь никакого эффекта, поскольку значение по умолчанию будет равно «0».*

3. Повторить код и стиль по примеру (рисунок 71).

Проверить результат. Должно получиться, что при наведении курсора мыши на элемент размером 100*100 пикселей, его ширина (width) будет увеличиваться на протяжении 2 секунд до 300 пикселей.

* Можно так же добавить изменения для высоты.

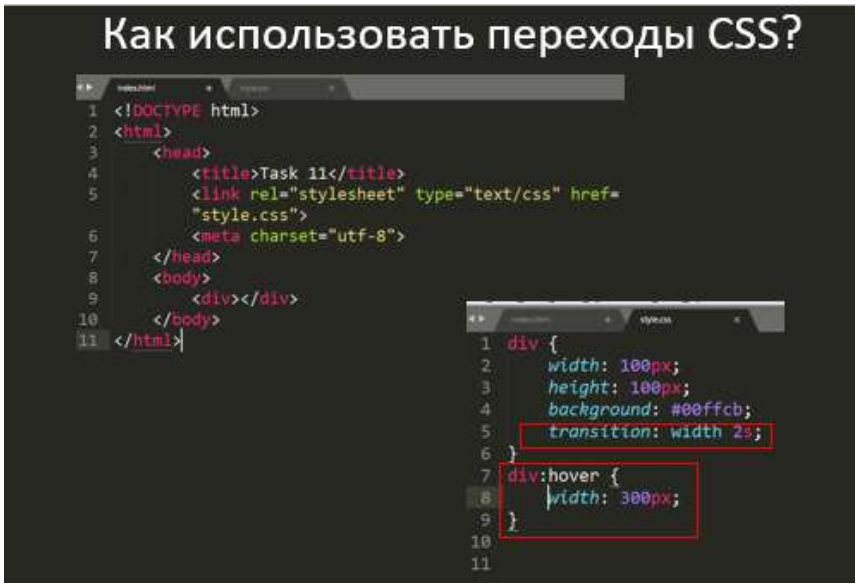


Рисунок 71 – Пример назначения анимации и псевдокласса «Hover»

Обратите внимание, что, когда курсор выходит за пределы элемента, он постепенно возвращается к своему исходному стилю.

«Hover» – это псевдокласс, означающий наведение указателя мыши на элемент. Псевдокласс используется для определения особого состояния элемента: при наведении на него указателя мыши или стиль посещенных и непосещенных ссылок.

** Самостоятельно посмотреть варианты псевдоклассов:

– https://webref.ru/css/type/pseudoclass

– https://www.w3schools.com/css/css_pseudo_classes.asp

Задание 2: Свойство перехода «transition»

1. Ознакомиться с особенностями свойства для перехода «transition» (рисунок 72).

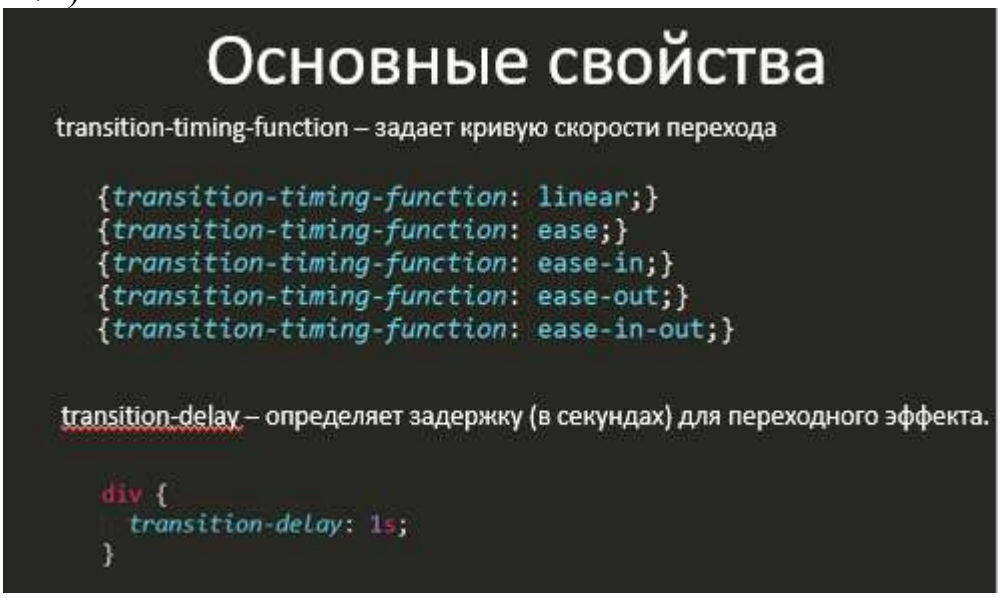


Рисунок 72 – Основные свойства перехода «transition»

2. Transition-timing-function задает кривую скорости эффекта перехода, может иметь следующие значения для назначения эффектов:

– ease – переход с медленным началом, затем быстрым, затем медленным завершением (по умолчанию);

– linear – одинаковая скоростью от начала до конца;

– ease-in – переход с медленным стартом;

– ease-out – переход с медленным окончанием;

– ease-in-out – медленное начало и окончание;

– cubic-bezier (n,n,n,n) – позволяет определять свои собственные значения.

3. Выбрать два любых значения и применить к элементу . Посмотреть результат.

4. Transition-delay –определяет задержку (в секундах) для переходного эффекта. Например – запуск перехода «отложен» на 1 секунду.

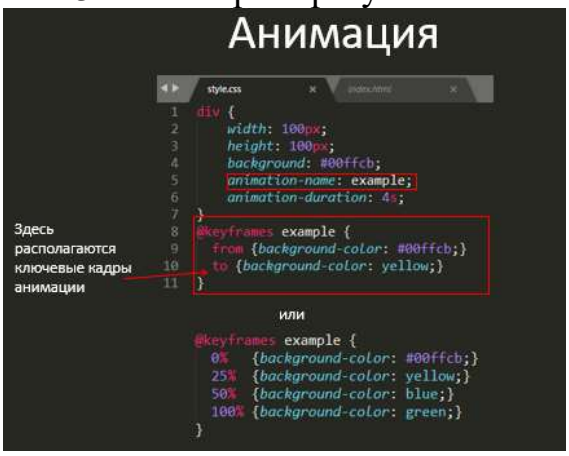
Свойства перехода CSS можно указывать одно за другим или используя сокращенное свойство transition (рисунок 73).

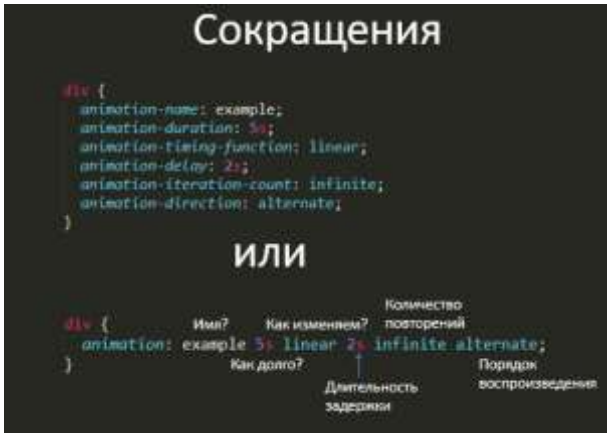
Повторить по примеру.

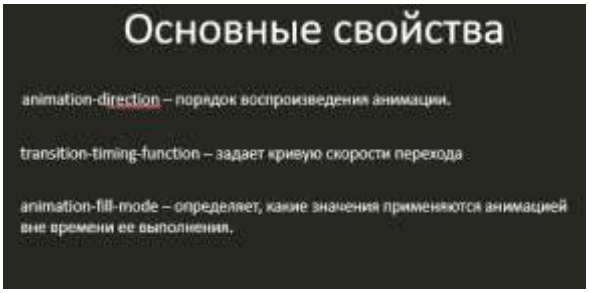
Рисунок 73 – Пример оформления стиля «transition-delay»

Задание 3: Анимация элементов (рисунок 74).

1. Самостоятельно ознакомиться с назначением анимации.
2. Повторить по примеру.
3. Посмотреть результат.







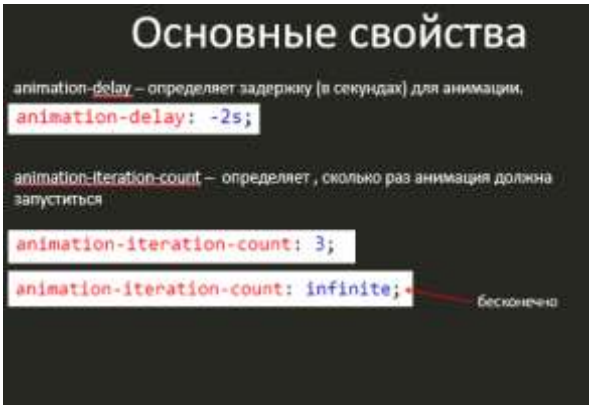


Рисунок 74 – Пример назначения анимации и основные свойства

3.19 Лабораторная работа 13. Графические и звуковые элементы

Цель: применить на практическом проекте градиентный фон и выпадающее меню.

Задание 1: Градиентный фон.

1. Познакомиться с добавлением градиентного фона.

Градиентные фоны были введены с CSS3, дизайнеры и фронтенд-разработчики вовсю радовались этому. Хотя градиентные фоны не работают в старых браузерах, они поддерживаются всеми современными браузерами.

Градиентным может быть не только фон всей страницы, но и другие отдельные элементы, например, кнопки.

В CSS градиентные фоны рассматриваются как фоновые изображения. Градиент можно добавить с помощью свойства background или background-image, как обычную фоновую картинку. Значение свойства для градиента меняется в зависимости от того, какой градиент нужен – линейный или радиальный.

Добавление линейного градиента с направлением «слева направо» (рисунок 75):

– использовать значение to right для указания направления,

– указать два цвета, от которого к какому будет совершаться переход.

* Самостоятельно рассмотреть вариации градиентов:

– https://webref.ru/layout/learn-html-css/setting-backgrounds-and-gradients – русский язык,

– https://www.w3schools.com/css/css3_gradients.asp – английский язык.

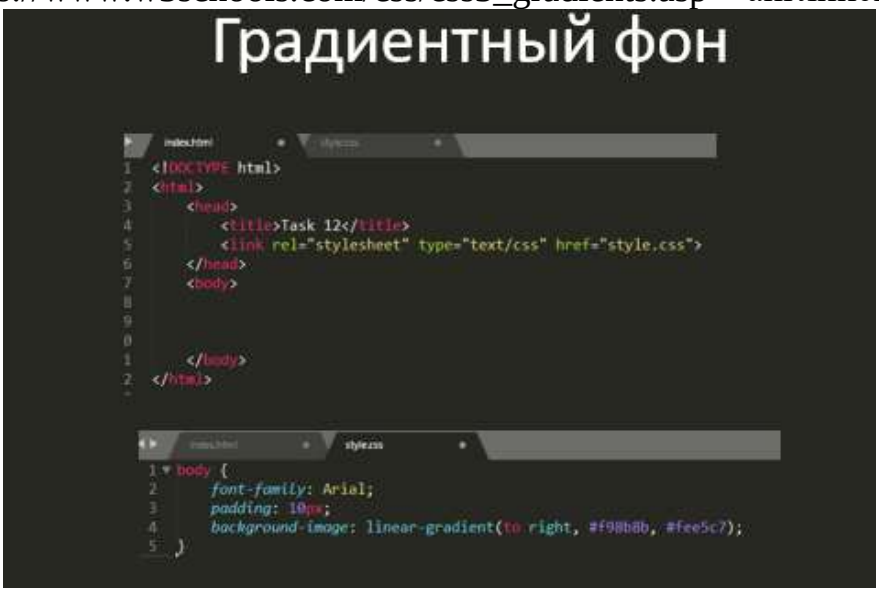


Рисунок 74 – Пример назначения градиента с направлением «слева направо»

Задание 2: Раскрывающийся список / выпадающее меню.

1. В файл html необходимо добавить (рисунок 75.1):

– один большой контейнер ,

– внутри добавить него кнопку с соответствующим классом,

– контейнер со списком (необходимое количество страниц).

2. Повторить по примеру назначение стилей (рисунок 75.2) и псевдоклассов (рисунок 75.3).

* Обратите внимание, что классы названы в соответствие с элементами, которыми они хранят, то есть прочитав класс, сразу понятно к какому элементу они относятся.

html

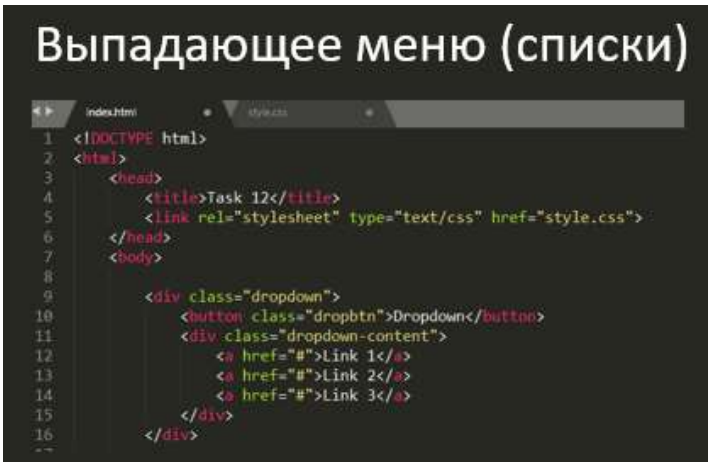


Рисунок 75.1 – Пример добавления выпадающего списка в html

css

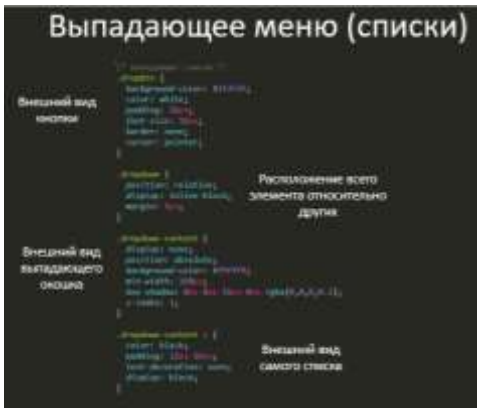


Рисунок 75.2 – Пример добавления стилей

css

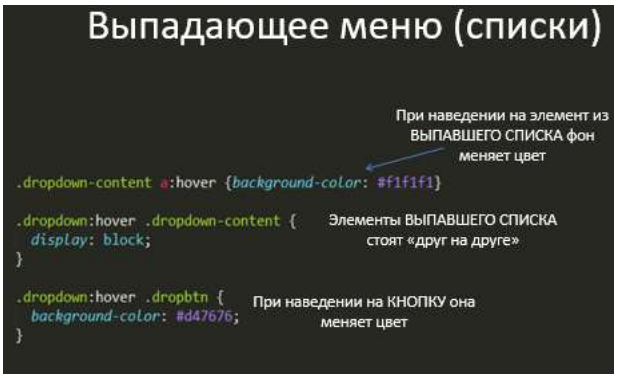


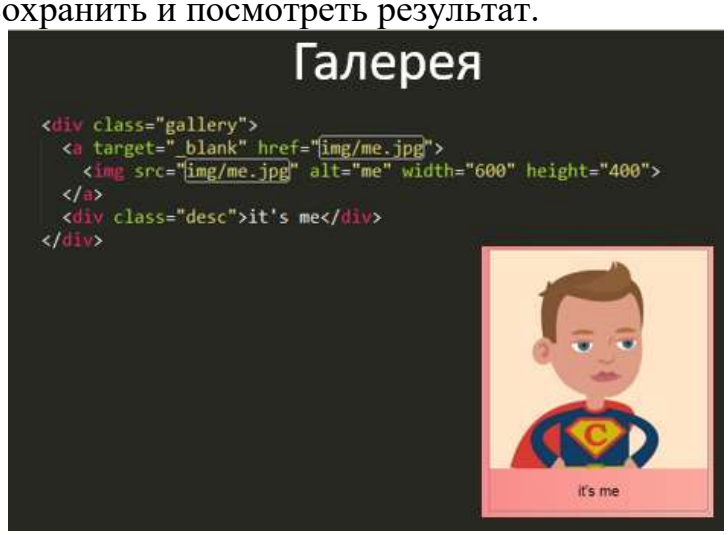
Рисунок 75.3 – Пример добавления псевдоклассов

Рисунок 75 – Раскрывающийся список / выпадающее меню

Задание 3: Самостоятельная работа – Галерея (рисунок 76).

1. Повторить по примеру.
2. Сохранить и посмотреть результат.

HTML



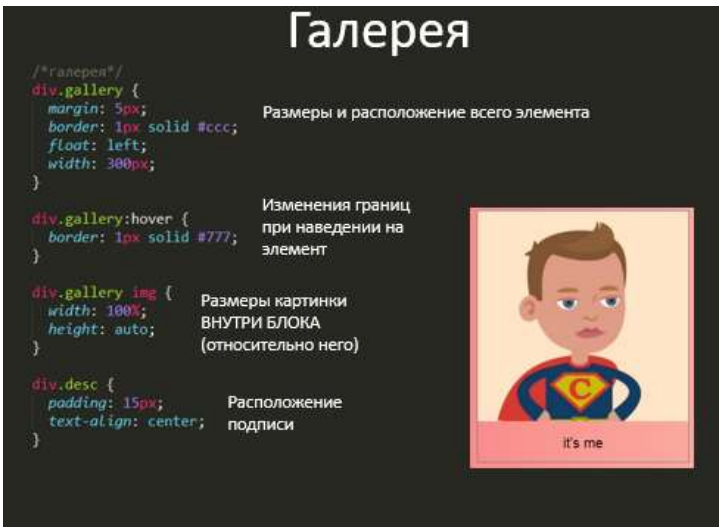


Рисунок 76 – Пример добавления галереи с назначением поведения

3.20 Лабораторная работа 14. Верстка сайта

Цель: получить практический опыт в вёрстке станиц сайта.

Задание 1: Ознакомиться с особенностями применения flexbox.

Долгое время единственными надёжными инструментами CSS верстки были такие способы как Float (обтекание) и позиционирование.

Недостатки их применения – сложно или невозможно достичь следующих требований к макету:

– вертикальное выравнивания блока внутри родителя;

– оформления всех детей контейнера так, чтобы они распределили между собой доступную ширину/высоту, независимо от того, сколько ширины/высоты доступно;

– сделать все колонки в макете одинаковой высоты, даже если наполнение в них различно.

1. Ознакомиться с основными свойствами flexbox:

– flex-direction;
– flex-wrap;
– flex-flow;
– justify-content;
– align-items;
– align-content.

Flexbox определяет набор CSS свойств для контейнера (flex-контейнер) и его дочерних элементов (flex-блоков). Первое, что нужно сделать – это указать контейнеру display:flex или display:inline-flex.

2. Повторите по примеру (рисунок 77). Посмотрите результат.

3. Примените разные варианты свойств flexbox (рисунок 78).

Рисунок 77 – Пример применения flexbox со свойствами flex-direction, justify-content, align-items

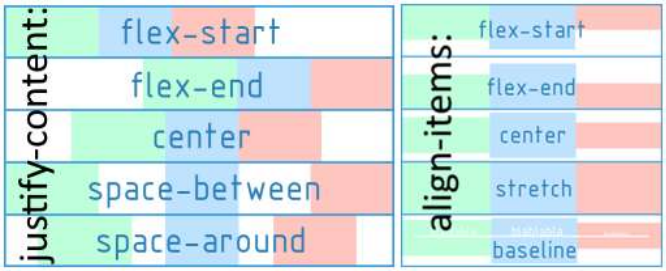


Рисунок 78 – сравнение свойств justify-content, align-items

Свойство flex-direction определяет, в каком направлении контейнера может сложить элементы flex.

* Примечание, при выборе некоторых значений необходимо регулировать ширину контейнера, иначе элементы внутри будут растягиваться. Если указана фиксированная ширина или высота макета, flexbox элементы переполнят контейнер и нарушат макет. Один из способов как это исправить – это добавить свойство *flex-wrap*.

Свойство *flex-flow* является сокращенным свойством для установки свойств flex-direction и flex-wrap.

Свойство *justify-content* используется для выравнивание элементов по горизонтали (главная ось):

– flex-start – значение стоит по умолчанию, располагает все элементы в начале главной оси;

– flex-end – располагает элементы в конце.

– center – располагает все элементы по центру главной оси;

– space-around – распределяет все элементы равномерно по главной оси, с небольшим количеством свободного места на обоих концах.

– space-between – похоже на space-around, за исключением того, что оно не оставляет места на обоих концах.

Свойство *align-items* используется для выравнивания элементов flex по вертикали (поперечная ось):

– значение stretch – стоит по умолчанию и растягивает все flex элементы, чтобы заполнить родителя вдоль поперечной (cross axis) оси. Если

у родителя нет фиксированной ширины вдоль поперечной оси, все flex элементы примут длину самого длинного flex элемента.

– значение center заставляет элементы сохранять свои собственные размеры, но центрирует их вдоль поперечной оси.

– значения flex-start и flex-end, которые выравнивают все элементы по началу и концу поперечной оси соответственно.

– значение baseline – флекс-элементы будут выравниваться по базовой линии текста в них. Эта воображаемая линия проходит по нижней части букв.

Свойство *align-content* – описывается в спецификации как «упаковка flex строк»; управляет промежутками между flex строками по перекрёстной оси. Если содержимое flex контейнера переносится на несколько строк, используется свойство align-content для управления свободным пространством между строками.

Задание 2: Самостоятельная работа.

* Свойства можно комбинировать между собой, благодаря этому получать различное всевозможное отображение элементов.

1. Выполнить все задания на ресурсе https://flexboxfroggy.com/#ru. Результат показать преподавателю.

2. Самостоятельно подобрать все значения свойств flexbox, чтобы получить результат – рисунок 79.

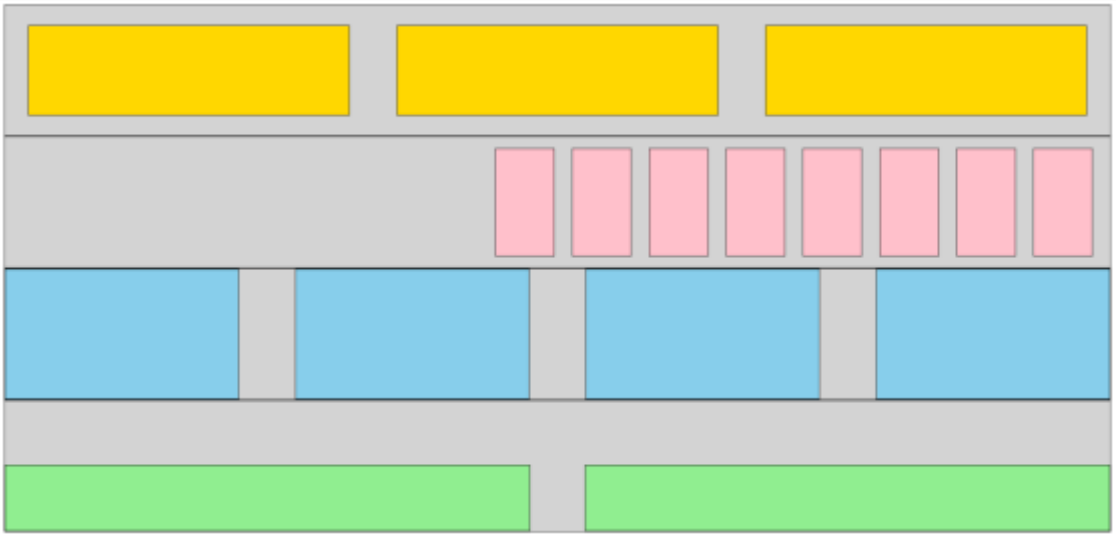


Рисунок 79 – Результат выполнения самостоятельного задания

3.21 Лабораторная работа 15-16. Верстка сайта

Цель: получить практический опыт в вёрстке адаптивных станиц сайта.

Задание 1. Ознакомиться с особенностями адаптивного веб-дизайна.

Веб-страницы можно просматривать с помощью множества различных устройств: компьютеров, планшетов и телефонов. Ввеб-страница должна хорошо выглядеть и быть простой в использовании независимо от устройства. Чтобы соответствовать устройствам меньшего размера, но не

пропускать информацию, поэтому лучше адаптировать свое содержимое к любому устройству.

Адаптивный веб-дизайн – это не программа или язык программарования, как JavaScript. Адаптивный веб-дизайн использует только HTML и CSS.

Область просмотра – это видимая область веб страницы пользователя и зависит от устройства.

Раньше веб страницы были предназначены только для экранов компьютеров, веб страницы должны были иметь статический дизайн и фиксированный размер и это было общим для планшетов и мобильных телефонов. Затем, когда серфинг в интернете в основном стал с помощью планшетов и мобильных телефонов, фиксированный размер веб страниц был слишком велик, чтобы соответствовать области просмотра. Поэтому браузеры уменьшили всю веб страницу до размера экрана.

HTML5 ввел метод, позволяющий веб дизайнерам управлять видовым экраном через тег <meta>, который необходимо установить во все веб страницы (рисунок 80).

Тег <meta> содержит инструкцию viewport браузера для управления размерами и масштабированием страницы.

Часть width=device-width задает ширину страницы следить за шириной экрана устройства (которая будет меняться в зависимости от устройства).

Часть initial-scale=1.0 задает начальный уровень масштабирования при первой загрузке страницы браузером.

```
<!DOCTYPE html>
<html>
    <head>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>
```

Рисунок 80 – Вставка тега <meta> в html страницы

Некоторые дополнительные правила:

1. Не используйте большие элементы фиксированной ширины – , например, если изображение отображается по ширине шире, чем область просмотра, видимый экран может вызвать горизонтальную прокрутку. Не забудьте подогнать содержимое под ширину окна просмотра.

2. Не позволяйте содержимому полагаться на определенную ширину окна просмотра, чтобы визуализация была хорошей - размеры и ширина экрана CSS в пикселях различаются шириной между устройствами, содержание не должно полагаться на определенную ширину видового экрана для хорошей визуализации.

3. Используйте медиа запросы CSS для применения различных стилей больших экранов - установка большой абсолютной ширины CSS для элементов страницы вызывающий элемент должен быть слишком широким для просмотра на мобильном устройстве. Вместо этого рекомендуется использовать относительные величины, например, width: 100%. Также быть

осторожным в использование больших абсолютных расположений значений. Он может причинить элементу к выводу за пределы области просмотра на небольших устройствах.

Задание 2: Повторить адаптивный веб-дизайн (рисунок 81).

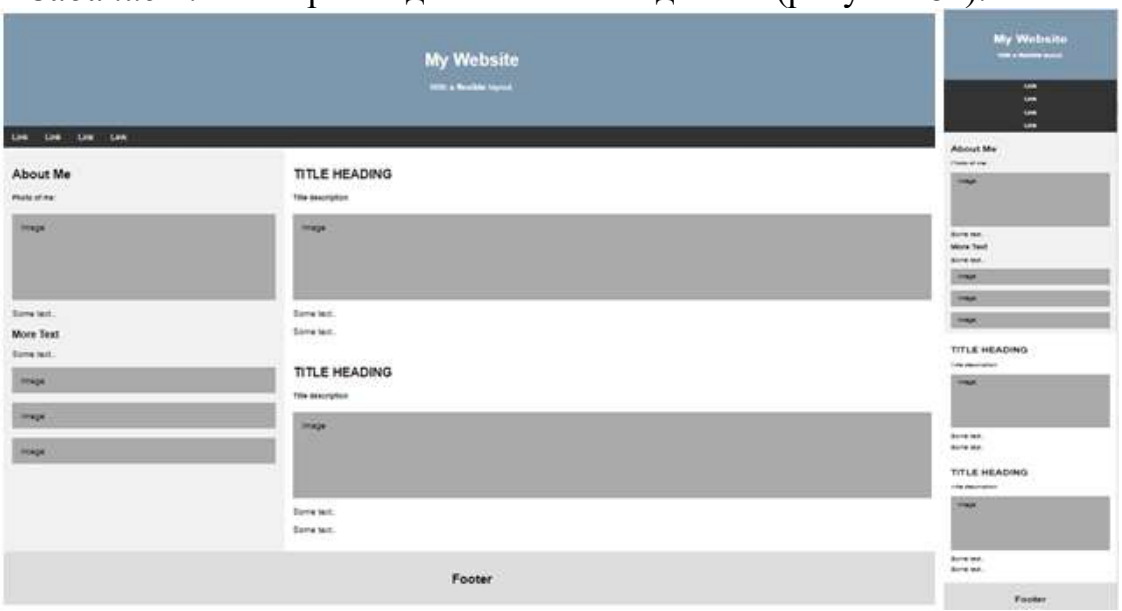


Рисунок 81 – Пример адаптивного дизайна для компьютера и мобильного телефона

1. Открыть предыдущую работу. Добавить еще один столбец слева. Повторить по примеру – рисунок 82.

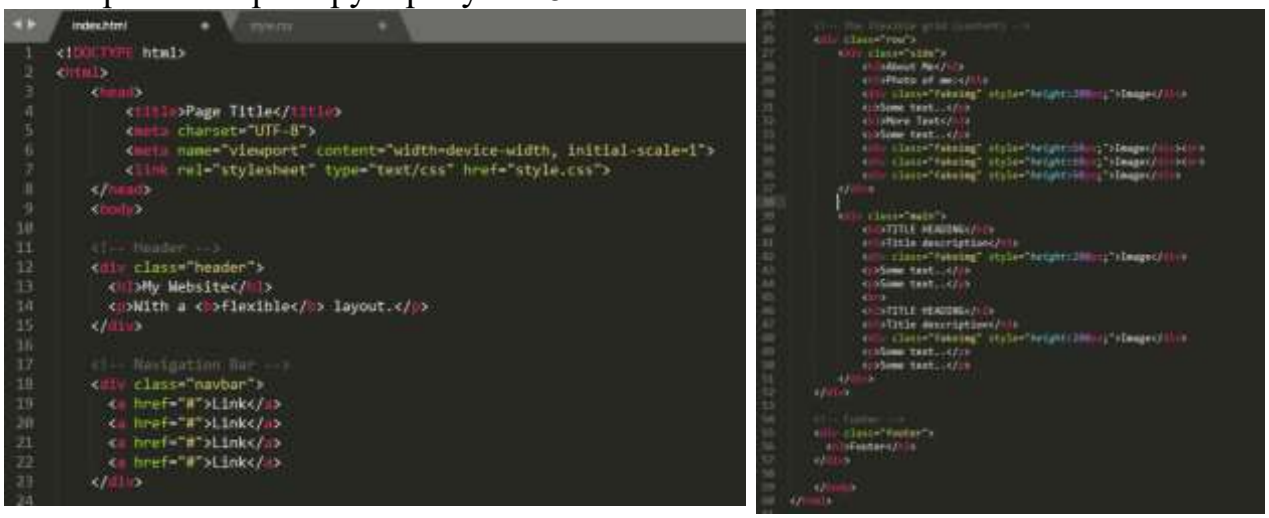
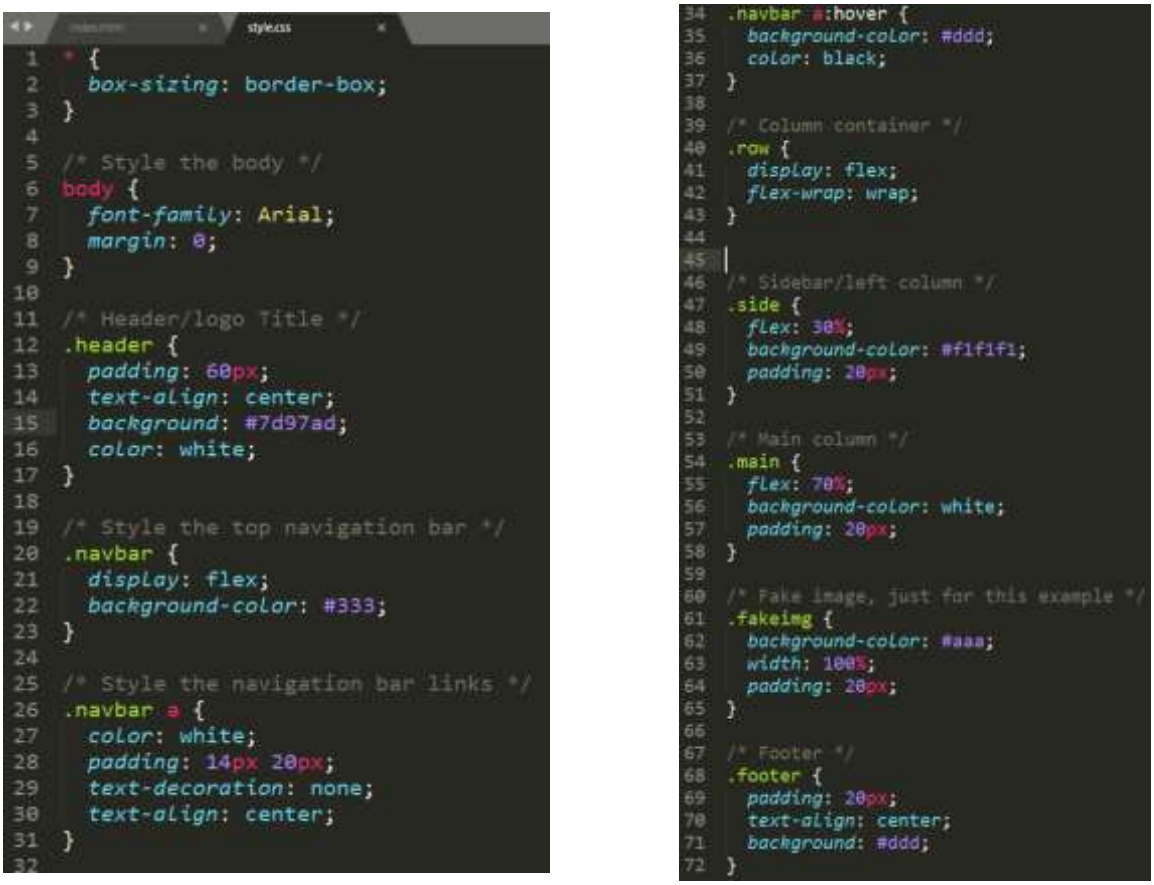


Рисунок 82 – Пример составления html документа

2. Добавить в CSS пример кода (рисунок 83).

* В CSS использовать конструкцию Flexbox. Обязательно указать какие элементы будут иметь данное свойство.



```css
* {
  box-sizing: border-box;
}

/* Style the body */
body {
  font-family: Arial;
  margin: 0;
}

/* Header/logo Title */
.header {
  padding: 60px;
  text-align: center;
  background: #7d97ad;
  color: white;
}

/* Style the top navigation bar */
.navbar {
  display: flex;
  background-color: #333;
}

/* Style the navigation bar links */
.navbar a {
  color: white;
  padding: 14px 20px;
  text-decoration: none;
  text-align: center;
}

.navbar a:hover {
  background-color: #ddd;
  color: black;
}

/* Column container */
.row {
  display: flex;
  flex-wrap: wrap;
}

/* Sidebar/left column */
.side {
  flex: 30%;
  background-color: #f1f1f1;
  padding: 20px;
}

/* Main column */
.main {
  flex: 70%;
  background-color: white;
  padding: 20px;
}

/* Fake image, just for this example */
.fakeimg {
  background-color: #aaa;
  width: 100%;
  padding: 20px;
}

/* Footer */
.footer {
  padding: 20px;
  text-align: center;
  background: #ddd;
}
```

Рисунок 83 – Пример составления CSS документа

3. Добавить медиазапросы (рисунок 84).

Медиазапросы используются в тех случаях, когда нужно применить разные CSS-стили, для разных устройств по типу отображения (например: для принтера, монитора или смартфона), а также конкретных характеристик устройства (например: ширины окна просмотра браузера), или внешней среды (например: внешнее освещение). Учитывая огромное количество подключаемых к интернету устройств, медиазапросы являются очень важным инструментом при создании веб-сайтов и приложений, которые будут правильно работать на всех доступных устройствах.

```css
/* Responsive layout - when the screen is less than 700px wide*/
@media screen and (max-width: 700px) {
      .row, .navbar {
        flex-direction: column;
      }
    }
```

Рисунок 84 – Пример добавления медиазапроса

* В данном случае создано правило для отображения на экранах, если ширина области просмотра составляет 700 пикселей или меньше, в таком случае элементы с классами .row и .navbar будут отображаться в столбик.

* Когда ширина экрана станет меньше 700px, левый столбец окажется сверху, над правым.

Задание 3. Самостоятельно доработать макет до полноценного проекта:

1. Наполнить сайт информацией.

2. Добавить красок (используй свои цвета).

3. Оформить текст, используя шрифты google fonts.

4. Добавить изображения, можно галерею. Оформить в единый стиль с подписью.

5. Оформить navbar, вставить ссылки на сторонние источники, добавить выпадающее меню.

6. Добавить анимацию для кнопок в navbar.

7. Оформить футер.

8. Добавить ссылки на социальные сети.

3.22 Лабораторная работа 17. Верстка сайта

Цель: изучить работу языка программирования javascript.

Задание 1. Ознакомиться с особенностями javascript.

Язык программирования JavaScript придумали специально для того, чтобы создавать интерактивные сайты. Такие сайты реагируют на действия: добавляют лайк, когда нажимают на «сердечко»; загружают новые посты в ленту; показывают оповещения о новом сообщении или письме. Для этого и нужен JavaScript. Сегодня это один из самых популярных и востребованных языков программирования, поэтому он необходим каждому веб-разработчику. Сейчас JavaScript – единственный язык программирования для браузеров. Он работает под Windows, macOS, Linux и на мобильных платформах, то есть везде.

Задание 2. Применить javascript на примере мини-игры «Убегающая кнопка».

* Мы создадим две кнопки, одну можно будет нажать, а другую нет, она будет убегать, при наведении на нее указателя мыши (рисунок 85).

Рисунок 85 – Пример готового проекта мини-игры «Убегающая кнопка»

1. Создать структуру HTML, CSS и файл JavaScript (рисунок 86).

– HTML состоит из вопроса и двух кнопок. В HTML для добавления JavaScript есть специальный тег, который находится в конце перед закрывающимся тегом </body>.

– Код на языке JavaScript называют скриптом. Его сохраняют в отдельный файл с расширением «.js» (например, main.js), а чтобы запустить, подключают этот файл на страницу.

* Браузер прогружает страницу сверху вниз. В случае, когда скрипты располагаются в теге <head> рендеринг останавливается для того, чтобы браузер скачал эти скрипты и затем продолжает работу, иногда это ведет к долгой загрузке контента. Поэтому, чтобы пользователь увидел контент быстро, подключение скриптов откладывают на самый последний момент.

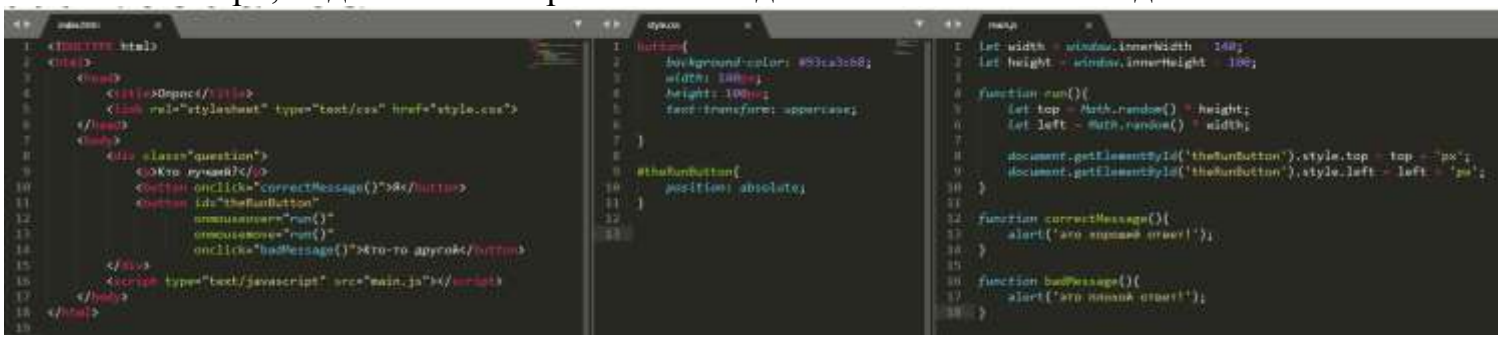


Рисунок 86 – Пример составления HTML, CSS. JS

** Программа будет выбирать рандомное положение, то есть будет задавать свойства top – отступ сверху и left – отступ слева, каждый раз, когда мы наведем указатель мыши на кнопку.

– Создать две переменные width и height с помощью ключевого слова let: для того чтобы программа знала ширину экрана. Чтобы кнопка не пряталась за пределами экрана, будет отниматься от ширины и длины экрана ширину(140) и длину(100) кнопки, таким образом создана область, в которой она будет появляться (пунктир).

2. Создать функции.

Функция – важнейший элемент структурного программирования, позволяющий группировать и обобщать программный код, который может позднее использоваться произвольное число раз:

– Функция, когда «убегающая» кнопка убегает по наведению на нее указателя мыши.

– Функция, когда нажимают на «НЕ убегающую» кнопку, программа выдаст «Это хороший ответ».

– Функция, если будет нажата «убегающая» кнопка, то программа выдаст «Это плохой ответ».

Задать функцию следующим образом:

– написать ключевое слово function,

– написать имя функции (что делает функция),

– открыть фигурные скобки и внутри написать тело функции (что она делает).

3. Создать метод.

Методы JavaScript это действия, которые можно выполнить с объектами, содержат набор вспомогательных функций и значений, доступных «через точку».

Метод Math.random() – внутри функции создаются две переменные – top и left. Это расстояния до края области, в которой будет перемещаться кнопка – это рандомные (случайные) отступы.

Метод Math.random() возвращает число с плавающей запятой из диапазона [0, 1). Поэтому если в отступе слева (left), например, нам выпадет число 0.5, то умножив его на ширину экрана (области перемещения кнопки), например, 700px, мы получим left=350px (left=0.5*700).

4. Добавить уникальный идентификатор в html.

В документе (document) находим элемент с уникальным идентификатором, который мы задали в HTML (getElementById('theRunButton')), присваиваем этому элементу для верхнего отступа наше новое рандомное значение (style.top = top+’px’), обязательно добавляем в конце ‘px’, так как мы получили ранее значение без единиц измерения, а в CSS, мы знаем, без единиц измерения работать не будет.

После того как функция готова, необходимо указать, когда ее надо «активировать». Для этого в HTML использовать события onmouseover и onmousemove.

Событие – это сигнал от браузера о том, что что-то произошло:

– mouseover – когда мышь наводится на элемент.

– mousemove – при движении мыши.

В этих случаях будет запускаться функция run().

P.S все события можно найти в интернете.

5. Добавить функцию для «правильной – не убегающей» кнопки.

Функция alert() предназначена для вывода в браузере предупреждающего модального диалогового окна с некоторым сообщением и кнопкой «ОК». При его появлении дальнейшее выполнение кода страницы прекращается до тех пор, пока пользователь не закроет это окно.

6. Добавить функцию, для «убегающей» кнопки.

Вывод сообщения при нажатии, написать самостоятельно (похоже на предыдущий пример).

Добавить событие onclick (нажатие) в HTML.

Добавить еще одно свойство в CSS для кнопки – абсолютное позиционирование, чтобы другие элементы отображались на веб-странице словно ее и нет.

3.23 Лабораторная работа 18. Верстка сайта

Цель: закрепить знания по работе языка программирования javascript.

Задание 1. Создать интерфейс интерактивной игры (рисунок 87).

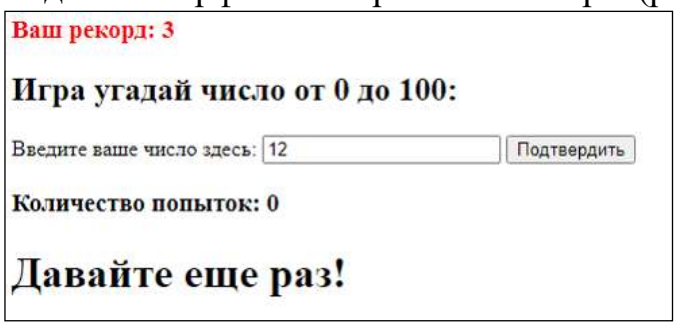


Рисунок 87 – Пример интерфейса программы

1. Повторить структуру HTML (рисунок 88).

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <h3 id="record" style="color: red;"></h3>

    <form action="">
        <h2>Игра угадай число от 0 до 100:</h2>
        <label for="">Введите ваше число здесь:</label>
        <input type="number">
        <input type="submit" value="Подтвердить">
    </form>

    <h3 id="attemts-text"></h3>
    <h1 id="output"></h1>

    <script src="main.js"></script>

</body>
</html>
```

Рисунок 88 – Пример структуры HTML

2. Повторить скрипт по примеру (рисунок 89).

main.js

```js
function newRandomNumber(){
    return Math.floor(Math.random()*100);
}

function outputAttempts(number){
    document.querySelector('#attemts-text').innerHTML = 'Количество попыток: '+number;
}

function outputMessage(message){
    document.querySelector('#output').innerHTML = message;
}

function outputRecord(number){
    document.querySelector('#record').innerHTML = 'Ваш рекорд: '+number;
}

document.addEventListener('DOMContentLoaded', function(){

    var numberOfAttempts = 0;
    var recordNumber = 0;
    var number =  newRandomNumber();
    var button = document.querySelector('form input[type=submit]');

    button.addEventListener('click',function(e){
        e.preventDefault(); // останавливает отправку формы
        var inputNumber = document.querySelector('form input[type=number]').value;
        if (inputNumber == ''){ // проверка на пустоту
            outputMessage("Вы не ввели число!"); // вывод сообщения
            return; // прерывание функции
        }
        numberOfAttempts++;
        outputAttempts(numberOfAttempts);
        if (inputNumber > number)
            outputMessage("Вы ввели число больше загаданного!");
        if (inputNumber < number)
            outputMessage("Вы ввели число меньше загаданного!");
        if (inputNumber == number){
            outputMessage("Вы УГАДАЛИ число!");
            if (recordNumber == 0 ||recordNumber >numberOfAttempts)
                recordNumber = numberOfAttempts;
            outputRecord(recordNumber);
            alert("ПОЗДРАВЛЯЕМ!!!");
            numberOfAttempts = 0;
            outputAttempts(numberOfAttempts);
            number =  newRandomNumber();
            outputMessage("Давайте еще раз!");
        }
    })
})
```

Рисунок 89 – Пример скрипта

Пояснения составления файла и скрипта:

– Заголовок H3 служит для вывода текста рекорда.

– В форме имеется описание игры в заголовке H2:

– поле для ввода числа – input type number,

– кнопка для подтверждения ввода – input type submit.

– Ниже формы в заголовке H3 – будет выводиться количество попыток.

– В H1 – важные сообщения для пользователя.

– Последней строкой подключить файл с JS.

Создать функции:

– newRandomNumber – возвращает случайное число в диапазоне от 0 до 100,

– outputAttempts – принимает число и выводит на экран количество попыток сделанных пользователем в текущей игре,

– outputMessage – принимает строку-сообщение и выводит это сообщение как важное, на экран для пользователя,

– outputRecord – принимает число и выводит в качестве рекорда на экран,

– querySelector – выбирает элемент на странице при помощи CSS селектора,

– innerHtml – управляет содержимым элемента.

3.24 Лабораторная работа 19. Верстка сайта

Цель: закрепить знания по работе языка программирования javascript.

*Задание 1. Создать ф*айловую структуру, которая состоит из 4-х элементов: index.html; magic.js; main.js; style.css.

Задание 2. Создать интерфейс интерактивного web-приложения «Предсказание на день». Повторить по примеру – рисунок 90-91.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Document</title>
</head>
<body>

    <div id="parent">
        <button id="start" class="active">узнать свою судьбу</button>
        <button id="stop">получить предсказание</button>
        <div id="fastText">?</div>
        <div id="text"></div>
    </div>

    <script src="magic.js"></script>
    <script src="main.js"></script>

</body>
</html>
```

```
#parent {
    text-align: center;
    margin-top: 100px;
}
#parent > * {
    margin-bottom: 10px;
}
#prediction {
    font-size: 18px;
}
#text {
    font-size: 17px;
    font-style: italic;
}
#prediction, #text {
    width: 800px;
    margin: 0 auto;
}
button {
    padding: 5px 10px;
    font-size: 15px;
    cursor: pointer;
}
button:not(.active) {
    display: none;
}
```

Рисунок 90 – Пример составление файла index.html и style.css

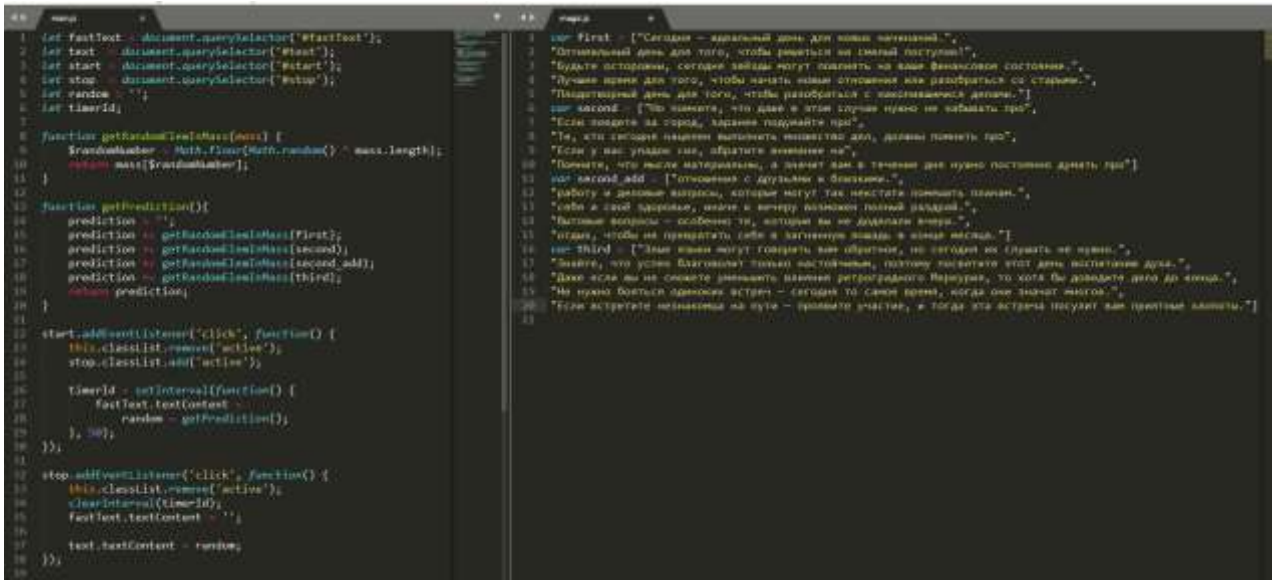


Рисунок 91 – Пример составление файла magic.js и main.js

Смысл работы данной программы в выборе случайного предсказания на день из списка готовых предложений.

На экране появится генерация всех внесённых предложений, и простой клик по кнопке выберет текущее предсказание.

Задание 3. Добавить список предложений до большего количества. Чем больше предсказаний – тем интереснее приложение. Также нужно сделать свою эксклюзивную стилизацию приложения.

3.25 Лабораторная работа 20. Верстка сайта

Цель: закрепить знания по работе языка программирования javascript. Знакомство с библиотекой Bootstrap.

Задание 1. Ознакомиться с особенностями CSS фреймворка Bootstrap.

Ссылку на подключение можно скопировать с официального сайта библиотеки (рисунок 92).

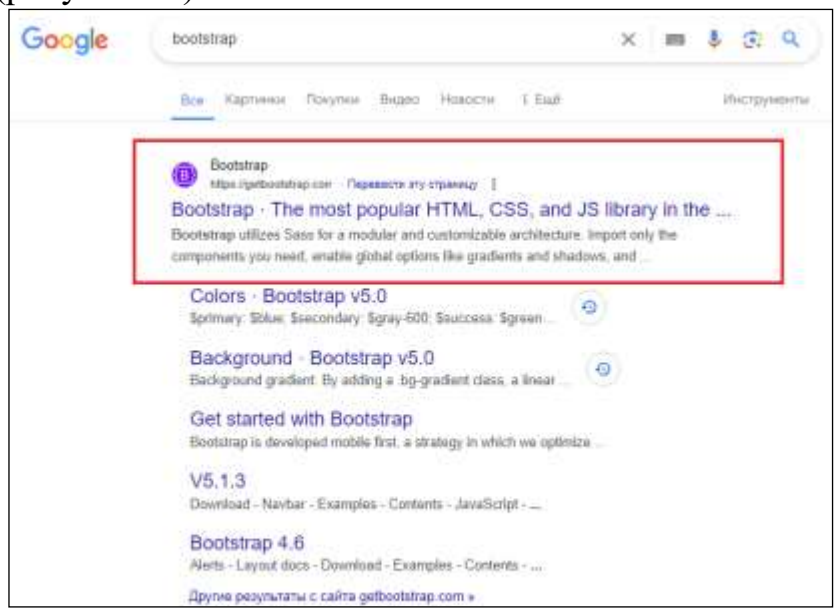


Рисунок 92 – Официальный сайт библиотеки Bootstrsp

Фреймворк (с англ. framework–«каркас, структура») – заготовка, готовая модель в программировании для быстрой разработки, на основе которой можно дописать собственный код. Он задает структуру, определяет правила и предоставляет необходимый набор инструментов для создания проекта. В основном фреймворки используются в веб-разработке.

Bootstrap –это заготовленный набор CSS классов под практически любой случай жизни! Каждая банальная задача имеет свой класс в Bootstrap. Вместо того чтобы прописывать много строчек кода в CSS файле, можно задать нужные классы элементу внутри HTML документа.

Задание 2. Повторить структуру HTML (рисунок 93).

Рисунок 93 – Пример составления HTML структуры

Задание 3. Повторить код javaScript (рисунок 94).

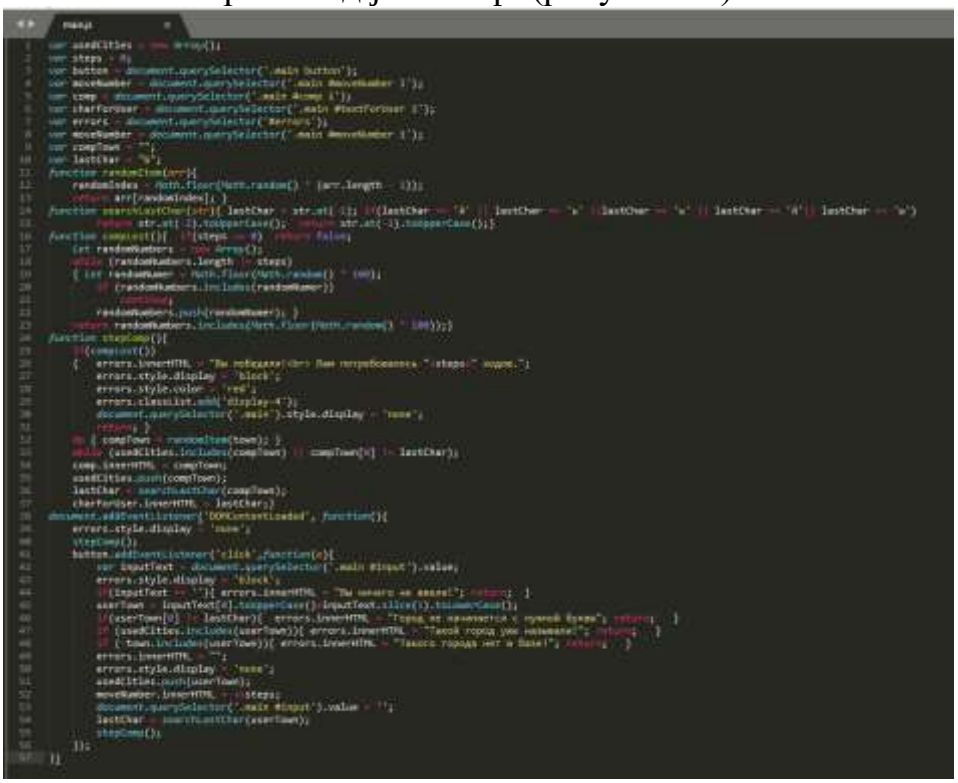
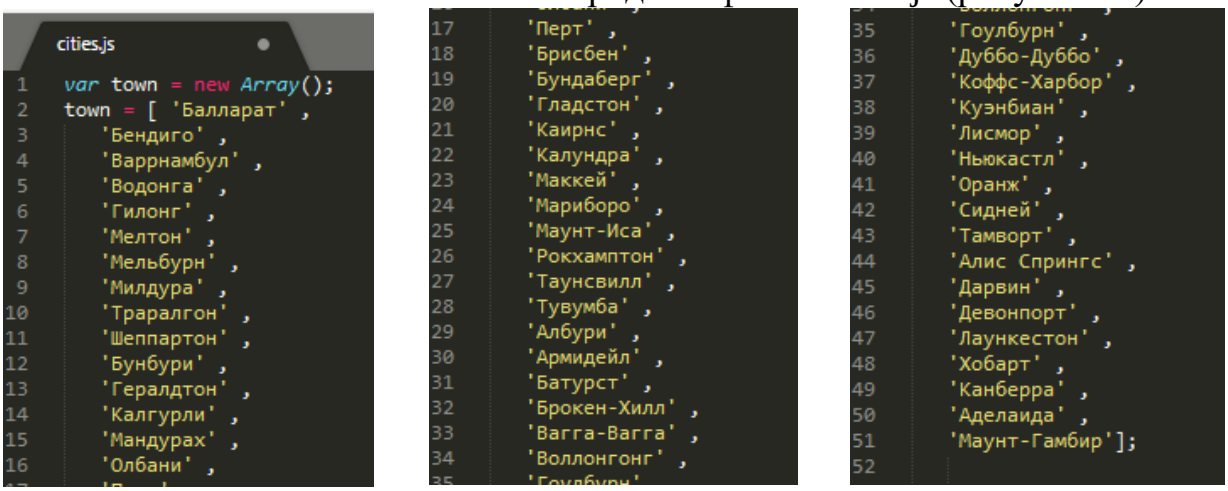


Рисунок 94 – Пример составления кода main.js

Задание 4. Составить список городов в файле cities.js (рисунок 95).



```
cities.js
var town = new Array();
town = [ 'Балларат' ,
    'Бендиго' ,
    'Варрнамбул' ,
    'Водонга' ,
    'Гилонг' ,
    'Мелтон' ,
    'Мельбурн' ,
    'Милдура' ,
    'Траралгон' ,
    'Шеппартон' ,
    'Бунбури' ,
    'Геральдтон' ,
    'Калгурли' ,
    'Мандурах' ,
    'Олбани' ,
    'Перт' ,
    'Брисбен' ,
    'Бундаберг' ,
    'Гладстон' ,
    'Каирнс' ,
    'Калундра' ,
    'Маккей' ,
    'Мариборо' ,
    'Маунт-Иса' ,
    'Рокхамптон' ,
    'Таунсвилл' ,
    'Тувумба' ,
    'Албури' ,
    'Армидейл' ,
    'Батурст' ,
    'Брокен-Хилл' ,
    'Варга-Варга' ,
    'Воллонгонг' ,
    'Гоулбурн' ,
    'Дуббо-Дуббо' ,
    'Коффс-Харбор' ,
    'Куэнбиан' ,
    'Лисмор' ,
    'Ньюкастл' ,
    'Оранж' ,
    'Сидней' ,
    'Тамворт' ,
    'Алис Спрингс' ,
    'Дарвин' ,
    'Девонпорт' ,
    'Лаункестон' ,
    'Хобарт' ,
    'Канберра' ,
    'Аделаида' ,
    'Маунт-Гамбир'];
```

Рисунок 95 – Пример составления кода cities.js

Список городов для проверки представлен отдельным файлом, где реализован массив названий. Естественно, чем больше городов в списке – тем игра будет более реалистичной. Скачать огромный по размеру список можно из сети.

Задание 5. Доработать приложение (рисунок 96).

– Набрать без ошибок код.

– Запустить работу приложения.

– Создать большой список названий городов.

– Реализовать смену фона после каждого хода.

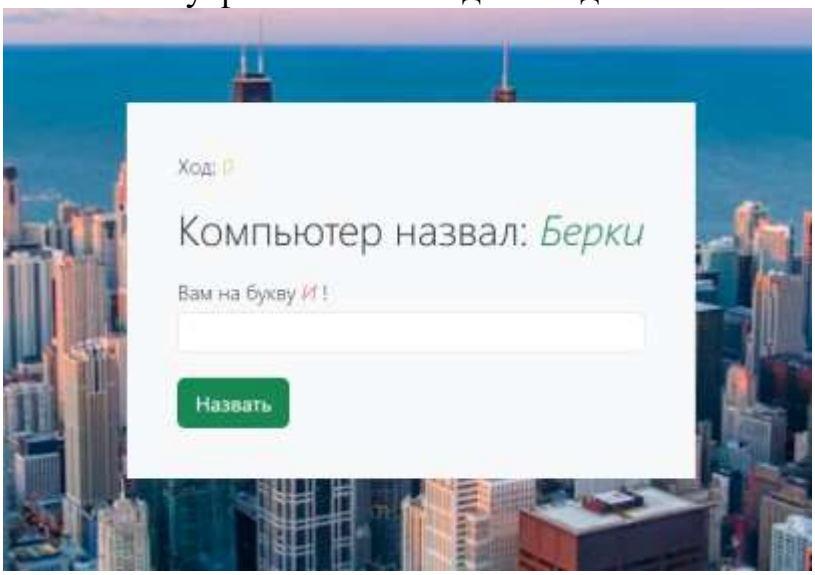


Рисунок 96 – Пример интерфейса интерактивной игры «Города»

4 ПРАКТИЧЕСКИЙ РАЗДЕЛ

4.1 Задания для управляемой самостоятельной работы студентов

Самостоятельная работа студентов направлена на совершенствование их умений и навыков по дисциплине «Основы фронтенд разработки». Цель самостоятельной работы студентов – способствование усвоению в полном объеме учебного материала дисциплины через систематизацию, планирование и контроль собственной деятельности.

Задание 1. «Тема 3. Язык HTML».

Разработать 3 страницы сайта «Рецепты», используя все полученные знания по языку HTML. Форма отчета – электронная версия (файл).

Задание 2. «Тема 4. Технология CSS»

Оформить созданный сайт «Рецепты» с помощью стилей. Форма отчета – электронная версия (файл).

Задание 3. «Тема 6. Верстка сайта»

Разработать персональный сайт по требованиям: минимум 3 страницы, оформление через стили, применение формы обратной связи, оформление галереи или портфолио, анимация, ссылки на внешние медиаресурсы. Форма отчета – электронная версия (файл).

4.2 Требования к проведению аттестации студентов

Рекомендованная форма проведения текущей аттестации студентов – тест. Примерный перечень тестовых вопросов:

1. Какая из перспектив развития Всемирной паутины связана с персонализацией и адаптацией контента под пользователя?

1) web 1.0 2) web 2.0 3) web 3.0 4) web 4.0

2. Кто считается одним из основателей современного веб-дизайна и автором книги 'Don't Make Me Think'?

1) Стив Круг 2) Тим Бернерс-Ли 3) Дональд Норман 4) Джеффри Зельдман

3. Что из перечисленного является веб-клиентом?

1) Apache HTTP Server 2) Nginx 3) Google Chrome 4) IIS

4. Какой браузер относится к движку WebKit?

1) Moxilla Firefox 2) Google Chrome 3) Safari 4) Microsoft Edge (старый)

5. Какая технология позволяет обновлять данные на веб-странице без её полной перезагрузки?

1) CSS3 2) AJAX 3) HTML5 4) PHP

6. Что из перечисленного является протоколом передачи данных в вебе?

1) HTML 2) HTTP 3) CSS 4) JavaScript

7. Какой элемент HTML5 используется для воспроизведения видео без плагинов?

1) <iframe> 2) <embed> 3) <object> 4) <video>

8. Какая особенность CSS3 позволяет создавать анимации без использования JavaScript?

1) Selector 2) Flexbox 3) Transitions 4) Grid Layout

9. Какой объект в JavaScript используется для работы с элементами веб-страницы?

1) API 2) JSON 3) DOM 4) AJAX

10. Что обозначает аббревиатура URL?

1) Unified Resource Link 3) Universal Reference Link
2) Universal Resource Locator 4) Uniform Resource Locator

11. Какой первый этап анализа веб-ресурса?

1) Программирование сайта 3) Определение целевой аудитории
2) Разработка прототипа 4) Тестирование

12. Что является ключевым элементом стратегии проектирования сайта? (Без этого невозможно построить логическую структуру сайта)

1) Создание графического дизайна 3) Наполнение сайта контентом
2) Определение целей и задач 4) Выбор доменного имени

13. Какой тип навигации предполагает использование меню и ссылок внутри одной страницы?

1) Глобальная навигация 3) Локальная навигация
2) Контекстная навигация 4) Внутренняя навигация

14. Какой инструмент чаще всего используется для создания интерактивных прототипов веб-страниц?

1) Notepad++ 2) WordPress 3) Adobe Photoshop 4) Figma

15. Что обозначает аббревиатура DNS?

1) Dynamic Name Source 3) Domain Name System
2) Digital Navigation Server 4) Data Network Service

16. Какая роль метатегов в SEO?

1) Создают навигацию по страницам 3) Управляют доступом к серверу
2) Помогают поисковым системам индексировать сайт 4) Определяют цветовую схему сайта

17. Какой орган регулирует белорусский сегмент сети интернет?

1) Министерство образования 3) Министерство связи и информации
2) Совет министров 4) Министерство внутренних дел

Для допуска к экзамену студент должен выполнить следующие требования:

– посещение лекционных занятий;
– выполнение семинарских, практических и лабораторных работ;
– выполнение заданий по управляемой самостоятельной работе;
– успешное прохождение промежуточной аттестации.

Форма проведения экзамена – устный опрос.

4.3 Перечень вопросов для проведения экзамена

1. Веб-дизайн в культуре и искусстве.
2. Сферы использования веб-технологий.
3. Принципы проектирования web-сайтов. Этапы проектирования веб-узла.
4. Место и роль веб-проектирования в системе социокультурных знаний.
5. Логическая структура сайта.
6. Этапы анализа веб-сайта. Цели и задачи сайта. Критерии качества веб-ресурса.
7. Основы дизайна (пространственные отношения, форма, текстуры, шрифты и тексты).
8. Типы сайтов: статические и динамические. Понятие систем управления содержимым (контентом) сайта – CMS.
9. Дизайн вэб-сайтов (Типы сайтов, устройство сайта, формат страницы).
10. Принципы успешной навигации.
11. Навигация десктопной и мобильной версии страницы. Основные элементы веб-страницы и способы их компоновки. Страницы с фиксированной шириной, «резиновые». Адаптивный дизайн.
12. Системы представления цвета в Интернет. Цветовые модели. Базовые цвета.
13. Основы Web-графики. Векторные и растровые изображения.
14. Форматы растровой графики JPEG, GIF, PNG. Особенности и ограничения.
15. Выбор разрешения растрового изображения и оптимизация его размеров в Adobe Photoshop.
16. Форматы векторных графических файлов в веб-пространстве: swf, svg, eps и др.
17. Типы графических изображений в Интернет (фон, визуал, логотип, навигационная графика, фовикон и др.). Изображения-распорки.
18. Интерактивная графика и анимация в веб-пространстве: новые технологии.
19. Динамические веб-страницы на основе JavaScript. Технологии Ajax.
20. Видео в веб-пространстве. Способы внедрения видео в документ. Форматы видео. Стандарты сжатия.
21. Звук в веб-пространстве. Форматы звуковых файлов. Сжатие звуковых файлов.
22. История языков разметки. HTML.
23. Синтаксис языка HTML.
24. Структура документов HTML. Строение страницы.
25. Структурные теги HTML. Ссылки. Теги размещения объектов.
26. Технология HTML 5.0.

27. Кодировка документа, определение ее вида. Кодировки кириллицы в веб-пространстве.
28. Макетирование веб-страниц с помощью CSS. Способы присоединения к документу.
29. Принципы табличной верстки. Преимущества и недостатки верстки веб-страниц с помощью таблиц.
30. Дизайн веб-страниц. Особенности и ограничения дизайна веб-сайтов.
31. Программные средства создания веб-страниц и веб-сайтов: текстовые, специализированные и визуальные редакторы.
32. Sublime Text - многофункциональный текстовый редактор программирования Web-страниц. Возможности.
33. Система доменных имен: DNS (Domain Name System). Выбор доменного имени ресурса.
34. Регистрация доменов. Получение имени своего сайта, создание структуры сайта на сервере и размещение своей страницы.
35. Подбор ключевых слов. Анализ статистики сайта.
36. Способы публикации ресурса в сети Интернет. Веб-хостинг. Белорусские провайдеры.
37. Поисковая оптимизация (SEO). Сервисы мониторинга веб-пространства.
38. Регистрация сайта (в поисковых системах, каталогах, топ-листах, в счетчиках, баннерный обмен, обмен ссылками, списки рассылки).
39. Конструкторы сайтов в интернет. Создание сайта онлайн. Обзор конструкторов сайтов.
40. Серверные платформы. Задачи веб-клиента и веб-сервера.
41. Ресурсы (услуги) Интернета. Онлайн сервисы.
42. PR в Интернет.
43. Реклама сайта.
44. Сопровождение сайта.
45. Законодательное регулирование национального сегмента сети Интернет.
46. Социальный маркетинг SMM (Social Media Marketing).
47. Ресурсы культуры и искусства в Интернет.

4.4 Критерии оценки результатов учебной деятельности студентов

В целях подготовки к текущей/промежуточной аттестации, студенту следует просмотреть все имеющиеся и рекомендуемые материалы, представленные в печатном или электронном виде. Промежуточная аттестация проводиться с целью оценки качества усвоения студентами всего объёма содержания дисциплины и определения фактически достигнутых знаний, навыков и умений, а также компетенций, сформированных за время аудиторных занятий и самостоятельной работы студента.

Критерии оценивания ответов студентов

Оценка «отлично» (10-8 баллов) / «зачтено». Ответы на поставленные вопросы излагаются логично, последовательно и не требуют дополнительных пояснений. Делаются обоснованные выводы. Демонстрируются глубокие знания в изучаемой области. Студент демонстрирует владение понятийным аппаратом и научным языком по предмету, умение его использовать в постановке и решении научных и профессиональных задач; способность самостоятельно решать сложные проблемы в рамках учебной программы; усвоение основной и дополнительной литературы, рекомендованной учебной программой; активная самостоятельная работа на лабораторных (практических) занятиях, высокий уровень культуры исполнения заданий, грамотное оформление учебной документации.

Оценка «хорошо» / «зачтено» (7-5 баллов). Ответы на поставленные вопросы излагаются систематизировано и последовательно. Материал излагается уверенно. Демонстрируется умение анализировать материал, однако не все выводы носят аргументированный и доказательный характер. Студент демонстрирует активную самостоятельную работу на практических, лабораторных занятиях, высокий уровень культуры исполнения заданий и оформления учебной документации, периодически участвует в групповых обсуждениях.

Оценка «удовлетворительно» (4 балла) / «зачтено». Допускаются нарушения в последовательности изложения. Имеются упоминания об отдельных базовых нормативно-правовых актах. Демонстрируются поверхностные знания вопроса, с трудом решаются конкретные задачи. Имеются затруднения с выводами. Студент демонстрирует достаточный объем знаний по предмету в рамках образовательного стандарта.

Оценка «неудовлетворительно» (3-1 баллов) / «не зачтено». Материал излагается непоследовательно, сбивчиво, не представляет определенной системы знаний по дисциплине. Не проводится анализ. Выводы отсутствуют. Ответы на дополнительные вопросы отсутствуют. На лабораторных (практических) занятиях студент был пассивен, демонстрировал низкий уровень культуры исполнения заданий и их оформления, отсутствие знаний по предмету в рамках образовательного стандарта или отказ от ответа.

5 ВСПОМОГАТЕЛЬНЫЙ РАЗДЕЛ

5.1 Учебная программа

Учреждение образования
«Белорусский государственный университет культуры и искусств»

УТВЕРЖДАЮ
Ректор БГУКИ
_____ Н.В.Карчевская
«31» октября 2025 г.
Регистрационный номер № УД-6/15-12 /эуч.

ОСНОВЫ ФРОНТЕНД РАЗРАБОТКИ

Учебная программа учреждения образования по учебной дисциплине модуля «Веб-дизайн и проектирование информационных ресурсов и систем» для специальности
6-05-0314-03 Социально-культурный менеджмент и коммуникации, профилизации «Мультимедийные технологии и цифровые коммуникации»

2025

Учебная программа составлена в соответствии с образовательным стандартом общего высшего образования по специальности 6-05-0314-03 Социально-культурный менеджмент и коммуникации, утвержденным постановлением Министерства образования Республики Беларусь от 21.08.2023 № 270 и учебным планом учреждения высшего образования по специальности 6-05-0314-03 Социально-культурный менеджмент и коммуникации, профилизации «Мультимедийные технологии и цифровые коммуникации», рег. № 6-05-03-70/24 уч. от 02.07.2024

СОСТАВИТЕЛИ:
Т.В. Бачурина, старший преподаватель кафедры информационных технологий в культуре учреждения образования «Белорусский государственный университет культуры и искусств»
О.М. Кунцевич, старший преподаватель кафедры информационных технологий в культуре учреждения образования «Белорусский государственный университет культуры и искусств»

РЕЦЕНЗЕНТЫ:

В.В. Казачёнок, заведующий кафедрой компьютерных технологий и систем учреждения образования «Белорусский государственный университет», доктор педагогических наук, кандидат физико-математических наук, профессор;
С.В. Вабищевич, доцент кафедры информатики и методики преподавания информатики учреждения образования «Белорусский государственный педагогический университет имени Максима Танка», кандидат педагогических наук, доцент

РЕКОМЕНДОВАНА К УТВЕРЖДЕНИЮ:
кафедрой информационных технологий в культуре учреждения образования «Белорусский государственный университет культуры и искусств» (протокол № 8 от 24.04.2025);
президиумом научно-методического совета учреждения образования «Белорусский государственный университет культуры и искусств» (протокол № 1 от 22.10.2025)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

В условиях современного общества неотъемлемым качеством квалифицированного специалиста сферы культуры и искусства становится высокий уровень владения интернет-технологиями для решения профессиональных задач по использованию, разработке, сопровождению и продвижению интернет-ресурсов в гипермедийном пространстве глобальной сети. Это предполагает знание основных технологий разработки сайтов, методов и инструментов их поискового и социального продвижения, умение применять их интегрировано для решения задач маркетинга и менеджмента.

Учебная дисциплина «Основы фронтенд разработки» направлена на овладение методами и средствами получения, хранения и обработки информации в области веб-дизайна и веб-проектирования, что позволит специалисту сферы культуры качественно выполнять профессиональные задачи.

Изучение учебной дисциплины «Основы фронтенд разработки» основывается на знаниях и умениях, полученных студентами в процессе освоения таких учебных дисциплин, как «Основы информационных технологий», «Информационная культура специалиста», «Системный анализ и моделирование информационных процессов», а также «Языки и системы программирования», «Технологии компьютерной графики», «Технологии создания баз данных сферы культуры», «Медиакультура специалиста».

Целью учебной дисциплины «Основы фронтенд разработки» является обеспечение студентов теоретическими знаниями и умениями в области веб-дизайна, навыками профессионального использования программных и технических средств веб-дизайна в будущей профессиональной деятельности менеджера при создании и сопровождении сайтов в области культуры и искусств.

Задачи учебной дисциплины:

– формирование представления об основных понятиях и концепциях гипертекстового пространства, технологий веб-дизайна; особенностей языков HTML, CSS; типов верстки сайтов и их особенностей; законодательного регулирования белорусского сегмента сети интернет.

– формирование системы базовых знаний этапов анализа и проектирования сайта;

– освоение технологий создания и публикации веб-ресурсов;

– освоение эффективных методов и особенностей поискового продвижения веб-ресурсов.;

В результате изучения учебной дисциплины студенты должны *знать*:

– понятия веб-пространства и веб-технологий;

– теоретические основы HTML, CSS, понятие о Javascript;

– тенденции развития веб-технологий;

– типы моделей верстки сайтов и их особенности;

– этапы проектирования и реализации веб-сайтов и их содержание;

– принципы и методы создания веб-ресурсов;

– методы публикации и сопровождения гипертекстовых ресурсов;

– методы поискового продвижения и поисковой оптимизации;

– специфику продвижения ресурсов сферы культуры и искусства в среде Интернет и социальных сетях.

Должны *уметь*:

– использовать при написании сайтов специализированные и визуальные редакторы веб-дизайна;

– применять различные методы верстки веб-страницы;

– применять современные инструментальные средства разработки для создания динамических и интерактивных веб-страниц;

– работать с содержимым веб-страниц, обрабатывать различные события, работать с формой и ее элементами;

– разрабатывать адаптивный дизайн сайтов;

– выполнять внутреннюю и внешнюю оптимизацию сайта.

Должны *владеть*:

– навыками использования инструментов создания веб-страниц с применением различных приемов верстки;

– основными навыками создания и продвижения веб-ресурсов.

В процессе изучения учебной дисциплины студенты должны приобрести следующую специализированную компетенцию:

– использовать технологии фронтенд разработки при создании веб-ресурсов социокультурной сферы.

В рамках образовательного процесса по учебной дисциплине студент должен не только приобрести теоретические и практические знания, умения и навыки по специальности, но и развить свой ценностно-личностный, духовный потенциал, сформировать качества патриота и гражданина, готового к активному участию в экономической, производственной, социально-культурной жизни страны.

Основными формами учебной дисциплины являются лекции, лабораторные, практические занятия и самостоятельное изучение отдельных вопросов.

Учебным планом на изучение учебной дисциплины «Основы фронтенд разработки» для студентов дневной формы получения образования всего предусмотрено 120 часов, из них 74 часа – аудиторные занятия. Примерное распределение аудиторных часов по видам занятий: лекции – 6, лабораторные занятия – 52, практические занятия – 16. Для студентов заочной формы получения образования всего предусмотрено 120 часов, из них 18 часов – аудиторные занятия. Примерное распределение аудиторных часов по видам занятий: лекции – 2 часа, лабораторные занятия – 12 часов, практические занятия – 4 часа.

Рекомендованная форма проведения текущей аттестации студентов - тест. Рекомендованная форма промежуточной аттестации студентов – экзамен.

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Тема 1. Введение. Технологии веб-дизайна

Цели и задачи дисциплины. Междисплинарные связи. Понятие гипертекстового пространство www: история создания, инструментальное, техническое, программное и организационное обеспечение. Ключевые принципы Интернета. Глобальная сеть Интернет и информационное пространство WWW: понятие и отличие. Классификация интернет-ресурсов, критерии качества и оценки.

Перспективы развития всемирной паутины. Персоналии веб-дизайна. Понятие веб-клиента и веб-сервера. Браузеры, их виды. Технологии построения веб-приложений. Технологические средства веб: URL-адресация, HTML язык, и HTTP протокол. Клиентские средства разработки веб-приложений: HTML, CSS, JAVASCRIPT. Возможности HTML5, CSS3, JavaScript и AJAX.

Тема 2. Основы проектирования сайта

Этапы анализа веб-ресурса. Стратегии проектирования. Анализ предметной области, определение назначения, задач сайта, целевой аудитории и информационной тематики. Пользователи сайта, их характеристика.

Основные этапы проектирования сайтов, дизайн интерфейсов, структурные схемы и дизайн-макеты страниц. Проектирование логической структуры сайта. Принципы навигации. Проектирование информационного наполнения. Типы контента сайта. Выбор информационных шаблонов сайта. Концепция графического дизайна сайта. Средства прототипирования. Программирование, наполнение и тестирование сайта.

Типы навигации на сайте. Разработка интерактивных прототипов веб-страниц. Важнейшие принципы информационной архитектуры и usability веб-ресурсов. Особенности подготовки текстов для веб-страниц.

Понятие онлайн-конструкторов сайтов. Создание веб-сайтов с помощью онлайн-конструкторов сайтов. Обзор конструкторов сайтов. Система доменных имен.

Метатеги как средство поискового продвижения интернет-ресурса. Законодательное регулирование белорусского сегмента сети интернет.

Тема 3. Язык HTML

Язык гипертекстовой разметки текста HTML (история, версии). Структура сайта: логическая, физическая. Требования к именам файлов. Именование главной (первой) страницы сайта.

Декларирование типа веб-документа. Синтаксис языка HTML. Тег. Контейннер (блок). Кодировки текста. Структура документов HTML. Содержимое области HEAD. Тег BODY и его параметры. Элементы, задающие шрифт. Элементы форматирования. Типы связей (ссылки). Формы.

Средства создания веб-страниц. Элементы гиперссылки: указатель ссылки, адресная часть. Абсолютный и относительный путь. Элементы веб-страницы: заголовок, текст, фон, графика, гиперссылки, таблицы.

Тема 4. Технология CSS

Технология CSS: назначение, история, основные конструкции, версии. Основные понятия и определения. Преимущества каскадных таблиц стилей. Принцип разделения содержимого и оформления веб-страницы. Стандарты CSS. Поддержка возможностей CSS со стороны браузеров. Подключение стилей. Синтаксис CSS. Цвета и фон. Шрифты. Текст. Блоки, границы и отступы. Методы реализации динамических страниц.

Тема 5. Графические и звуковые элементы

Оптимизация размеров изображения. Форматы графических файлов в Интернет: растровые: gif, png, jpg и векторные: swf , svg. Функции графики: фон, логотип, визуал, выставочная графика, навигационная и рекламная графика. Интерактивная графика и анимация в веб-пространстве.

Звук в веб-пространстве. Форматы звуковых файлов. Сжатие звуковых файлов. Видео в веб-пространстве. Форматы видео. Стандарты сжатия. Способы внедрения звука и видео в документ.

Тема 6. Верстка сайта

Модели верстки: фиксированная, резиновая, адаптивная и смешанная. Типовые макеты.

Табличная верстка сайта. Таблицы как средство компоновки веб-страниц. Принципы табличной верстки. Свойства CSS, описывающие табличную модель. Достоинства и недостатки верстки веб-страниц с помощью таблиц.

Блочная верстка сайта. Особенности поддержки блочной модели различными браузерами. Абсолютное и относительное позиционирование блоков. Особенности использования различных моделей позиционирования для верстки веб-страниц. Конструктивное использование тегов , и стилей. Свойства CSS, описывающие блочную модель. Макетирование веб-страниц с помощью CSS.

Основные методы достижения адаптивности сайта. Принципы построения адаптивного сайта. Техники адаптивной верстки. Свойства модуля flexbox и особенности верстки на его основе. Семантичность кода страницы: применение HTML-элементов, именование элементов, комбинация именованных элементов.

Разработка веб-сайта по заданной тематике. Определение цели, задач сайта, целевой аудитории сайта. Разработка структуры. Выбор макета и типа верстки. Определение общей структуры страницы. Подготовка контента и графических изображений. Верстка страниц сайта. Тестирование сайта. Анализ и оптимизация кода. Презентация готового проекта.

УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА УЧЕБНОЙ ДИСЦИПЛИНЫ

Дневная форма получения образования

Номер темы	Название темы	Количество аудиторных часов			Количество часов УСР	Форма контроля знания
		Лекции	Практические занятия	Лабораторные занятия		
1.	Введение. Технологии веб-дизайна	2	2			
2.	Основы проектирования сайта	2	2	2	2	
3.	Язык HTML	1	2	10	4	Индивидуальное задание
4.	Технология CSS	1	2	10	4	Индивидуальное задание
5.	Графические и звуковые элементы		2	4		
6.	Верстка сайта		4	14	6	Индивидуальное задание
Всего:		6	14	40	14	

УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА УЧЕБНОЙ ДИСЦИПЛИНЫ

Заочная форма получения образования

Номер темы	Название темы	Количество часов для дневной формы получения образования	Количество аудиторных часов			Количество часов для самостоятельного изучения учебного
			Лекции	Практические занятия	Лабораторные занятия	
1.	Введение. Технологии веб-дизайна	4	1			3
2.	Основы проектирования сайта	6	1			5
3.	Язык HTML	17		2	2	13
4.	Технология CSS	17		2	4	11
5.	Графические и звуковые элементы	6			2	4
6.	Верстка сайта	24			4	20
Всего:		74	2	4	12	56

ИНФОРМАЦИОННО-МЕТОДИЧЕСКАЯ ЧАСТЬ

Литература

Основная

1. Беликова, С. А. Основы HTML и CSS: проектирование и дизайн веб-сайтов: учебное пособие по курсу «Web-разработка» : [16+] / С. А. Беликова, А. Н. Беликов ; Южный федеральный университет. – Ростов-на-Дону ; Таганрог : Южный федеральный университет, 2020. – 176 с. : ил. – URL: https://biblioclub.ru/index.php?page=book&id=598663.

2. Никулова, Г. А. Проектирование и реализация Web-интерфейса : учебно-методическое пособие / Г. А. Никулова. – Липецк : Липецкий ГПУ, 2020. – 66 с. – URL: https://e.lanbook.com/book/156075.

3. Петракова, Н. В. Основы HTML : учебно-методическое пособие / Н. В. Петракова. – Брянск : Брянский ГАУ, 2022 – Часть 1 – 2022. – 50 с. – URL: https://e.lanbook.com/book/304958.

4. Побединский, Е. В. Проектирование веб-сайтов с использованием технологий PHP, HTML, CSS и WordPress : учебное пособие / Е. В. Побединский, В. В. Побединский. – Екатеринбург : УГЛТУ, 2018. – С. 5-27; 49-87. – URL: https://e.lanbook.com/book/142518

5. Смоленцева, Т. Е. Базовые и прикладные информационные технологии. Разработка Web-приложений : учебно-методическое пособие / Т. Е. Смоленцева. – Москва : РТУ МИРЭА, 2021. – 78 с.– URL: https://e.lanbook.com/book/218702.

6. Хромушин, В. А. Сборник примеров HTML страниц : учебное пособие / В. А. Хромушин, Р. В. Грачев, Н. Д. Юдакова. – Тула : ТулГУ, 2022. – 192 с. – URL: https://e.lanbook.com/book/264062.

7. JavaScript в HTML-документах : методические указания / составители А. А. Логачев, Н. Б. Смелова. – Санкт-Петербург : СПбГЛТУ, 2018. – 28 с. – URL: https://e.lanbook.com/book/107779.

Дополнительная

1. Аграновский, А. В. Тестирование веб-приложений : учебное пособие / А. В. Аграновский. – Санкт-Петербург : ГУАП, 2020. – 155 с. – URL: https://e.lanbook.com/book/216533.

2. Батенькина, О.В. Юзабилити информационных систем: учебное пособие / О.В. Батенькина, О.Н. Ткаченко. – Омск: ОмГТУ, 2015. – 144 с. – URL:https://e.lanbook.com/book/149059.

3. Журавлева, И.А. Технология разработки интернет ресурсов: курс лекций : учебное пособие : [16+] / авт.-сост. И. А. Журавлёва. – Ставрополь : Северо-Кавказский Федеральный университет (СКФУ), 2018. – 171 с. : ил. – URL: по подписке. – URL: https://biblioclub.ru/index.php?page= book&id=562579.

4. Можаров М. С. Проектирование и разработка информационных систем с web-интерфейсом: Учебное пособие / М.С.Можаров, А.Э.Можарова; М-во науки и высшего образования Рос.Федерации. - Новокузнецк: НФИ КемГУ, 2019. - 135 с.// Лань : электронно-библиотечная система. – URL: https://reader.lanbook.com/book/169625#2.

5. Роббинс, Дженнифер Нидерст. Веб-дизайн для начинающих. HTML, CSS, JavaScript и веб-графика / Дженнифер Нидерст Роббинс. - 5-е изд. - Санкт-Петербург : БХВ-Петербург, 2021. - 912 с., [22] л. фот. : рис., табл.

6. Саблина, Н.А. Основы Web-дизайна : учебно-методическое пособие / составитель Н.А. Саблина. – Липецк : Липецкий ГПУ, 2018. – 50 с. – URL: https://e.lanbook.com/book/115017.

7. Тузовский, А. Ф. Проектирование и разработка web-приложений : учеб. пособие для академического бакалавриата / А. Ф. Тузовский. – М. : Издательство Юрайт, 2016. – 218 с. – Серия : Университеты России.

8. Фрэйн, Бен. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств = Responsive Web Design with HTML5 and CSS3 / Бен Фрэйн ; [пер. с англ. Н. Вильчинский]. - 2-е изд. - Санкт-Петербург : Питер, 2017. - 272 с. : ил.

Рекомендуемые методы преподавания

Материал излагается на основе современных методических требований с учетом уровня знаний студентов. При чтении лекций особое внимание уделяется рассмотрению основ фронтенд разработки и практического применения полученных знаний в различных направлениях сферы культуры и искусства. Практические занятия направлены на формирование умений и навыков, использование полученных теоретических знаний при выполнении конкретных заданий по тематике учебной дисциплины. Методика проведения указанных занятий должна содействовать развитию творческих способностей каждого студента и приобретению навыков самостоятельной работы. Следует применять новые формы организации процесса обучения: визуализированные лекции, индивидуальная практическая работа и т. п.

Перечень рекомендованных средств диагностики

Для измерения степени соответствия учебных достижений студента требованиям образовательного стандарта рекомендуется использовать проектную деятельность, включающую проблемные, творческие задачи, предполагающие эвристическую деятельность и неформализованный ответ.

Для выявления и исключения пробелов в знаниях студентов рекомендуется использовать следующие средства:

– фронтальный опрос на лекциях, лабораторных и семинарских занятиях;

– защита выполненных на лабораторных занятиях работ;

– выполнение тестовых заданий для контроля умения анализировать и грамотно выбирать метод моделирования;

– выполнение творческих заданий и их оформление, которые предполагают самостоятельный выбор метода решения задачи.

– консультации и собеседование.

Рекомендации по организации самостоятельной работы студентов

Самостоятельная работа студентов направлена на обогащение их умений и навыков по учебной дисциплине, усвоение в полном объеме содержания через систематизацию, планирование и контроль собственной деятельности. Рекомендованная форма заданий для управляемой самостоятельной работы (индивидуальное задание) может быть: выполнение практических заданий по написанию HTML верстке, создание информационного ресурса по заданной тематике и другое. Такая организация работы способствует развитию как информационной, так и профессиональной компетенции.

5.2 Основная литература

1. Беликова, С. А. Основы HTML и CSS: проектирование и дизайн веб-сайтов: учебное пособие по курсу «Web-разработка» : [16+] / С. А. Беликова, А. Н. Беликов ; Южный федеральный университет. – Ростов-на-Дону ; Таганрог : Южный федеральный университет, 2020. – 176 с. : ил. – URL: https://biblioclub.ru/index.php?page=book&id=598663.

2. Никулова, Г. А. Проектирование и реализация Web-интерфейса : учебно-методическое пособие / Г. А. Никулова. – Липецк : Липецкий ГПУ, 2020. – 66 с. – URL: https://e.lanbook.com/book/156075.

3. Петракова, Н. В. Основы HTML : учебно-методическое пособие / Н. В. Петракова. – Брянск : Брянский ГАУ, 2022 – Часть 1 – 2022. – 50 с. – URL: https://e.lanbook.com/book/304958.

4. Побединский, Е. В. Проектирование веб-сайтов с использованием технологий PHP, HTML, CSS и WordPress : учебное пособие / Е. В. Побединский, В. В. Побединский. – Екатеринбург : УГЛТУ, 2018. – С. 5-27; 49-87. – URL: https://e.lanbook.com/book/142518

5. Смоленцева, Т. Е. Базовые и прикладные информационные технологии. Разработка Web-приложений : учебно-методическое пособие / Т. Е. Смоленцева. – Москва : РТУ МИРЭА, 2021. – 78 с. – URL: https://e.lanbook.com/book/218702.

6. Хромушин, В. А. Сборник примеров HTML страниц : учебное пособие / В. А. Хромушин, Р. В. Грачев, Н. Д. Юдакова. – Тула : ТулГУ, 2022. – 192 с. – URL: https://e.lanbook.com/book/264062.

7. JavaScript в HTML-документах : методические указания / составители А. А. Логачев, Н. Б. Смелова. – Санкт-Петербург : СПбГЛТУ, 2018. – 28 с. – URL: https://e.lanbook.com/book/107779.

5.3 Дополнительная литература

1. Аграновский, А. В. Тестирование веб-приложений : учебное пособие / А. В. Аграновский. – Санкт-Петербург : ГУАП, 2020. – 155 с. – URL: https://e.lanbook.com/book/216533.

2. Батенькина, О.В. Юзабилити информационных систем: учебное пособие / О.В. Батенькина, О.Н. Ткаченко. – Омск: ОмГТУ, 2015. – 144 с. – URL:https://e.lanbook.com/book/149059.

3. Журавлева, И.А. Технология разработки интернет ресурсов: курс лекций : учебное пособие : [16+] / авт.-сост. И. А. Журавлёва. – Ставрополь : Северо-Кавказский Федеральный университет (СКФУ), 2018. – 171 с. : ил. – URL: по подписке. – URL: https://biblioclub.ru/index.php?page= book&id=562579.

4. Можаров М. С. Проектирование и разработка информационных систем с web-интерфейсом: Учебное пособие / М.С.Можаров, А.Э.Можарова; М-во науки и высшего образования Рос.Федерации. - Новокузнецк: НФИ КемГУ, 2019. - 135 с.// Лань : электронно-библиотечная система. – URL: https://reader.lanbook.com/book/169625#2.

5. Роббинс, Дженнифер Нидерст. Веб-дизайн для начинающих. HTML, CSS, JavaScript и веб-графика / Дженнифер Нидерст Роббинс. - 5-е изд. - Санкт-Петербург : БХВ-Петербург, 2021. - 912 с., [22] л. фот. : рис., табл.

6. Саблина, Н.А. Основы Web-дизайна : учебно-методическое пособие / составитель Н.А. Саблина. – Липецк : Липецкий ГПУ, 2018. – 50 с. – URL: https://e.lanbook.com/book/115017.

7. Тузовский, А. Ф. Проектирование и разработка web-приложений : учеб. пособие для академического бакалавриата / А. Ф. Тузовский. – М. : Издательство Юрайт, 2016. – 218 с. – Серия : Университеты России.

8. Фрэйн, Бен. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств = Responsive Web Design with HTML5 and CSS3 / Бен Фрэйн ; [пер. с англ. Н. Вильчинский]. - 2-е изд. - Санкт-Петербург : Питер, 2017. - 272 с. : ил.

9. 10 лучших стратегий создания веб-сайтов [Электронный ресурс]. Режим доступа: https://justwebworld.com/wp-content/plugins/gtranslate/url_addon/gtranslate.php?glang=ru&gurl=best-10-strategies-designing-websites/&utm_source=copilot.com

10. 19 примеров веб-дизайна, который все портит [Электронный ресурс]. Режим доступа: https://ziex.by/blog/19-primerov-veb-dizajna-kotoryj-vse-portit?utm_source=copilot.com

11. UX / UI дизайн веб-сайтов [Электронный ресурс]. Режим доступа: https://silverweb.by/dizajn/?utm_source=copilot.com

12. Веб-студия [Электронный ресурс]. Режим доступа: https://b-r.by/?utm_source=copilot.com

13. Топ-10 самых популярных сайтов новостей в Беларуси [Электронный ресурс]. Режим доступа: https://justarrived.by/ru/news/top-10-most-popular-news-websites-in-belarus?utm_source=copilot.com

14. Этапы и алгоритм создания сайта с нуля – пошаговая система, которая помогает создать продающий целевой проект [Электронный ресурс]. Режим доступа: https://web-valley.ru/o-studii/etapy-sozdaniya-sajta?utm_source=copilot.com