Geekbrains

**Разработка Frontend части сайта на примере лэндинга сайта о путешествиях.**

Программа: Разработчик

Специализация: Frontend разработка (React)

Ткачев Денис Вадимович

Г. Нижний Новгород

2024 год

Введение (2–3 стр.)

Глава 1. Основы Frontend разработки (~15 стр.)

1.1 Структура сайта, что такое Frontend веб-сайта

1.2 Кто такой Frontend разработчик. Основные этапы разработки веб-сайта или приложения

1.3 Этапы Frontend разработки

1.4 Что такое макет сайта и какую роль он играет во Frontend разработке

Глава 2. Основные инструменты Frontend разработки веб-сайта (~15 стр)

2.1 Язык разметки HTML5

2.2 Каскадные таблицы стилей CSS3, SASS

2.3 Язык программирования JavaScript

2.4 Фреймворки и библиотеки: ускорение разработки

2.5 Адаптивный дизайн: создание адаптивных интерфейсов

Глава 3. Разработка Frontend части сайта (~20 стр.)

3.1 Макет сайта

3.2 Разметка сайта HTML

3.3 Стили сайта, использовался препроцессор SASS

3.4 Функционирование сайта, язык программирования JavaScript

3.5 Использование библиотек React

3.7 Подведение итогов.

Заключение (~ 4 стр.)

Список используемых источников

Введение

Глава 1. Основы Frontend разработки

* 1. Структура сайта, что такое Frontend веб-сайта

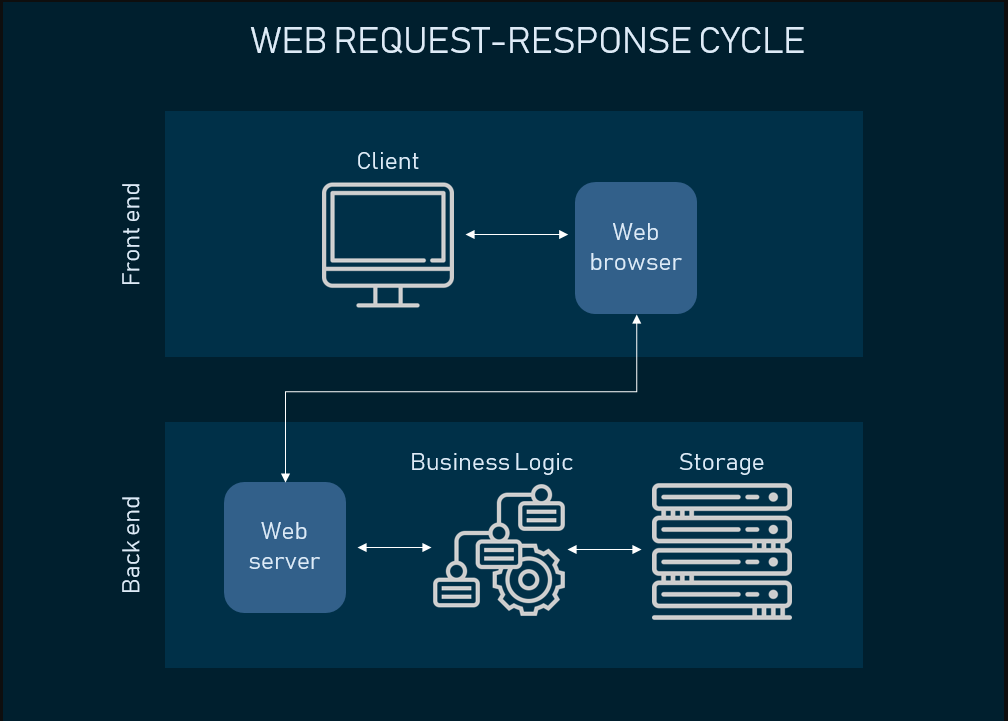
Любой сайт состоит из пользовательской и серверной частей. На странице в интернете вы видите текст, кнопки, панели, изображения и видео. Можете перемещаться по сайту, свободно изучать содержимое веб-сайта. Перед вами — пользовательская часть сайта и результат труда frontend-разработчика: визуализация, интерактивность и понятность интерфейса. Вы видите красивый дизайн, подсвеченные кнопки и интересную типографику, сайтом удобно пользоваться. Но как всё это выглядит изнутри?

За логику, работоспособность и правильное функционирование сайта отвечает серверная часть, которая скрыта от пользователя. Её созданием занимается backend-разработчик, а управлять может только администратор сайта через специальный интерфейс.

Любой запрос, который делает пользователь, передаётся на сервер. Здесь всё и происходит: запрос обрабатывается, фильтруется, а ответ отправляется обратно. Backend-разработка отвечает за правильное выполнение этого процесса.

Frontend веб-сайта — это все, что пользователь видит и с чем может взаимодействовать при помощи браузера. Для разработки frontend в качестве базовых инструментов используются: HTML (для создания базовой структуры страниц и контента), CSS (для стилизации внешнего вида) и JavaScript (для добавления интерактивности). Такой же набор инструментов используется в процессе создания прогрессивных веб-приложений — мобильных приложений, которые выглядят, как нативные, но при этом создаются с участием frontend -технологий.

На рисунке ниже представлено разделение frontend и backend частей веб-сайта.



Frontend и Backend части веб-сайта.

* 1. **Кто такой Frontend разработчик. Основные этапы разработки сайта или приложения**

**Frontend** -**разработчик** (от англ. Frontend Developer) создаёт интерфейсы веб-сайтов и приложений, с которыми взаимодействуют пользователи. Дизайнер определяет внешний вид сайта, а разработчик пишет код, чтобы этот дизайн отображался в интернете. И отвечает за то, чтобы пользователю было удобно взаимодействовать с сайтом.

Frontend-разработчик создаёт структуру страницы: добавляет контент, интерактивные кнопки, настраивает расположение блоков на странице, «оживляет» сайт с помощью анимации. Другими словами — создаёт элементы веб-сайта, которые видит пользователь. Они должны в точности соответствовать тому, как их отобразил в макете дизайнер, и работать так, как указал заказчик или менеджер.

Большая часть работы специалиста заключается в том, чтобы веб-сайт или приложение были простым в навигации и понятным для пользователя. Например, чтобы эта статья одинаково хорошо отображалась и на телефоне, и на экране компьютера, разработчик адаптировал блог под разные типы устройств.

Над сайтом обычно трудится команда разработчиков и дизайнеров. backend -разработчик создает инфраструктуру, которая позволяет веб-сайту обрабатывать запросы данных. Дизайнер рисует макет, продумывая расположение кнопок, картинок и текста. А frontend-разработчик с помощью кода визуализирует этот макет и делает сайт удобным для пользователей.

Для этого он выполняет следующие задачи:

● Верстает страницы, добавляет текст, изображения, кнопки, иконки, всплывающие окна и чаты.

● Делает сайт интерактивным: проверяет, чтобы все кнопки работали, ссылки вели на правильные страницы, всплывающие окошки отображались в нужной части страницы.

● Настраивает кроссбраузерность, чтобы сервис выглядел одинаково хорошо на разных устройствах.

Специалисты востребованы во всех сферах, где создают сайты или приложения.

Frontend-разработчик может найти работу:

● IT-компании, где программисты занимаются коммерческой разработкой сайтов, приложений, ПО, корпоративных сервисов и CRM.

● IT-отделы компаний. Маркетплейсы, EdTech-платформы, операторы сотовой связи и банки, онлайн-магазины и соцсети нуждаются в постоянном совершенствовании своих сервисов. Разработчик может работать как внутри компании, так и в агентстве на аутсорс-разработке.

● Разработчики корпоративного софта. Специалисты работают над созданием различных CRM-систем — «Битрикс24», «Мегаплан», сервисов управления проектами — Trello, Asana, веб-приложений для бизнеса — Google Документы (Google Docs), Tilda и пр.

Основные этапы разработки сайта или приложения

На первом этапе работы над будущим сайтом важно выявить конечного пользователя. После этого мы определяем, что и как пользователи будут делать на сайте, — разрабатываем пользовательские сценарии. На этом же этапе проектируется архитектура и логика работы продукта.

Дальше проект передается команде разработчиков. Она начинает работу по web-проекту на основе ТЗ от заказчика: аналитики, прототипов и дизайн-концепции.

* Разработка дизайн-макетов

Дизайн-макет — это набор всех страниц сайта или экранов приложения под необходимые разрешения и устройства. В дизайн-макете также описаны правила использования всех элементов.

На этом этапе работают дизайнеры и менеджер. В своей работе они опираются на аналитику, прототипы, дизайн-концепцию и SEO-рекомендации.

Задача менеджера — подобрать на проект дизайнеров с необходимым опытом. Он также контролирует чистоту макетов и единство стилистики.

Если речь идет о сайте, то для каждого экрана разрабатываются адаптивные макеты. Так с ним будет удобно работать на экранах разных устройств.

Если в проекте подразумевается сложная анимация и визуальные эффекты, то на этапе разработки дизайн-макета подключаются frontend-разработчики. Они заранее продумывают техническую реализацию дизайнерских идей. Разработчик сразу будет знать, с чем ему нужно будет работать. А дизайнеры адаптируют свою работу под возможности реализации.

В финале макеты готовятся к верстке, формируется UI-кит. Это PSD- или Sketch-файлы с полезными элементами для дизайна интерфейсов и сценарии их взаимодействия с пользователем. Тут же описываются все типы заголовков и их поведение при изменении верстки. UI-кит помогает frontend-разработчикам во время верстки сохранить стилистическое единство всех разделов.

* Программирование

Теперь, когда дизайн-макет готов, за работу берутся frontend-разработчики. На основе дизайн-макета, UI-кита, архитектуры продукта они программируют визуальные решения, пользовательские сценарии и функциональность. Их цель — сделать продукт удобным для пользователей и эффективным с точки зрения бизнеса.

**Frontend-разработчики** пишут скрипты, при помощи которых подгружается контент.Делают элементы интерактивными.Реализуют сложные анимации и визуальные эффекты.Пишут API для интеграции с бэкендом или CMS-системой. API — это описание способов взаимодействия программ между собой. Она позволяет связать действия пользователя с бизнес-логикой продукта.Пишут юнит-тесты для проверки, насколько корректно работает та или иная функциональность.

**Backend-разработчики** реализуют процессы, происходящие на серверной части сайта в ответ на действия пользователя. Они делают так, чтобы продукт функционировал. Программируют серверную часть. Проектируют базы данных. Настраивают пользовательские роли и управление в админпанели.

Разрабатывают API для интеграции со сторонними сервисами.

* Тестирование

Тестирование помогает оценить проделанную работу и исправить выявленные ошибки. Его проводят после каждого этапа разработки. Чем больше функций у веб-сервиса, тем больше тестов. Основные виды тестирования:

Контроль качества Frontend. Дизайнеры проверяют, насколько готовая верстка соответствует макетам.

Функциональные автотесты. Специальные боты проходят по заданным сценариям и выдают отчеты о технических ошибках.

Unit-тесты. Их пишут разработчики в процессе создания продукта. Такие тесты проводятся после доработки каждой функции. Они отслеживают стабильность работы, находят и исправляют конфликты.

Нагрузочное тестирование. Помогает оценить производительность сервиса при возрастании нагрузки. Таким образом можно определить мощности, необходимые для правильной работы сервиса.

* Релиз сайта или приложения

После завершения всех этапов:

Переносят продукт на сервер заказчика.

Передают инструкцию по использованию.

При необходимости проводят обучение пользователей.

Как правило, компания-разработчик не бросает работу над проектом после релиза.

* Развитие и поддержка

После запуска проекта, чтобы он эффективно работал и приносил прибыль, необходимо его поддерживать. В техническую поддержку входят:

Гарантийное обслуживание. Студия разработки предоставляет гарантию на разработанный продукт на фиксированный срок. Например, вскоре после релиза приложения в нем обнаружили ошибки, связанные с кодом разработчиков. Если это произошло в срок действия гарантийного обслуживания, то исправления вносятся бесплатно.

Абонентское обслуживание. Часто используется для ecommerce-проектов, которым важно, чтобы сайт, приложение или маркетплейс всегда был доступен. Специалисты техподдержки должны при любом форс-мажоре, например, хакерской атаке, быстро устранить проблему.

Развитие проекта. В этом случае команда специалистов дорабатывает и улучшает функциональность и интерфейс продукта: добавляет новые фичи, меняет дизайн, занимается контекстной и таргетированной рекламой, SEO-оптимизацией, контент-наполнением.

Заказчику выгоднее пользоваться услугами технической поддержки того же подрядчика, что и разработал веб-сервис. Дальнейшими работами по проекту занимается команда специалистов (менеджеры, дизайнеры, разработчики, аналитики), которая разрабатывала сайт или приложение. Им не надо долго вникать в проект или чужой код.

Для сравнения, если предприятие обращается за разовыми работами по технической поддержке к подрядчику, то на его проект отправят любого освободившегося специалиста.

Еще одно преимущество для заказчика — готовая команда высококвалифицированных специалистов. Техподдержка на аутсорсе освобождает время и средства компании. Не надо тратить ресурсы на открытие технического отдела, искать и нанимать специалистов.

В рамках техподдержки проводится постоянный мониторинг веб-ресурса. Например, если сайт "упал", то специалисты техподдержки сразу же об этом узнают. Произойти это может по разным причинам: по ошибке заказчика, из-за проблем с хостингом.

Для техподдержки важно, как можно быстрее исправить ситуацию. Даже если проблему устранили до того, как о ней узнал заказчик, техподдержка проинформирует его об инциденте.

* 1. **Этапы Frontend разработки**

Frontend-разработка является одним из самых динамично развивающихся направлений в сфере информационных технологий. С каждым годом появляются новые инструменты, технологии и подходы, которые делают пользовательские интерфейсы более красивыми, функциональными и удобными для пользователей. Тут мы рассмотрим основные этапы frontend-разработки, начиная от создания статических страниц на HTML и CSS и заканчивая разработкой динамических приложений на React.

**HTML и CSS: основы веб-разработки**

HTML (HyperText Markup Language) и CSS (Cascading Style Sheets) являются основными языками для создания статических веб-страниц. HTML используется для структурирования контента, а CSS – для оформления и стилизации элементов. Важно понимать основные принципы работы этих языков, чтобы создавать красивые и удобные интерфейсы.

**JavaScript: язык программирования для веб-сайта**

JavaScript – это клиентский язык программирования, который позволяет делать веб-страницы интерактивными и динамическими. С помощью JavaScript можно обрабатывать события, изменять содержимое страницы, отправлять запросы на сервер и многое другое. Освоение JavaScript – важный шаг для любого frontend-разработчика.

**Фреймворки и библиотеки:**

Фреймворки и библиотеки, такие как React, Vue.js, Angular и другие, помогают ускорить разработку веб-приложений, предоставляя готовые компоненты, инструменты для управления состоянием приложения и другие полезные функции. Изучение и использование таких инструментов позволяет создавать сложные приложения более эффективно.

**Адаптивный дизайн: создание адаптивных интерфейсов**

С развитием мобильных устройств становится все важнее создавать интерфейсы, которые хорошо выглядят и работают на различных устройствах и разрешениях экрана. Адаптивный дизайн позволяет создавать интерфейсы, которые автоматически адаптируются под разные условия использования.

* 1. **Что такое макет сайта и какую роль он играет во Frontend разработке**

Макет - это концепция будущего сайта, и от того, насколько глубоко он проработан, зависит восприятие блоков, механики взаимодействий и информации в целом.

Создание сайтов - длительный и трудоемкий процесс, поэтому к дизайну макета сайта особые требования. Они делаются для того, чтобы спорных ситуаций в процессе дальнейшей разработки было меньше, потому что после этапа дизайна сайт нужно сверстать и запрограммировать.

Какие требования предъявляются к макетам дизайна

В первую очередь у каждого артборда (страницы дизайна) должна быть единая сетка. Сетки очень разные, выбор зависит от того, для чего создается веб-сайт. Например, для интернет-магазина подойдет более стандартизированная сетка, где все очевидно, логично и понятно. Не нужно, чтобы пользователь лишний раз думал, куда ему нажать, чтобы произошло какое-то действие. Также бывают и креативные веб-сайты, например, какой-нибудь арт-галереи, где на первый взгляд может показаться, что элементы расставлены хаотично, но в этом тоже есть своя задумка.

Удаление ненужных слоев, скрытых или неактуальных вариантов. Это требование больше касается макетов, созданных в Adobe Photoshop, потому что слоев может быть больше тысячи, и при дальнейшей разработке верстальщику будет сложно понять “а что же по итогу мы верстаем?”. В Figma есть система компонентов и вариантов внутри компонента, у тому же все стили можно сразу увидеть, если дизайнер позаботился заранее, и собрал их все в одном месте.

Очень важно группировать слои по смыслу и называть соответствующе. В группе может быть, как весь первый экран, так и набор из заголовка-текста описания-кнопки. Это делается для того, чтобы не искать какой-то элемент по всему артборду в Figma или Adobe Photoshop. Артборды называть по названию страницы.

Одинаковые отступы в блоках. Например, в Figma есть инстурумент auto layout, с помощью которого можно задать блокам одинаковые отступы и быстро варьировать с настройками. В Adobe Photoshop за этим моментом нужно более тщательно следить, чтобы не ошибиться.

Цветовая модель макета должна быть RGB. Это стандарт для мониторов и экранов.

Масштабировать картинки необходимо пропорционально, чтобы не происходило искажений.

Не использовать системные псевдостили шрифта в формате жирности или курсива. Использовать только начертания из семейства шрифта Light/Bold/Regular/Light Italic/Bold Italic/Regular Italic.

Необходимо избавляться от дробных расстояний в отступах и в размерах иконок. Размеры должны быть целым числом.

Все важные элементы собрать где-то в одном месте. Либо это Style Guide с четким описанием всего, либо ui kit - особой разницы нет, главное, чтобы в макете этот момент был предусмотрен.

Желательно прикреплять архив со шрифтами в дополнение к макету. Если шрифт есть в Google Fonts, то давать ссылку на него.

Для сложных взаимодействий лучше всего прилагать референсы, создавать анимацию и описывать её. Таким образом с большей вероятностью будет сделано как задумывалось. Анимировать макет можно с помощью Photoshop или других специальных программ: Adobe Animate, Adobe Edge, Adobe After Effects или Principle.

При соблюдении этих несложных правил, разработка на дальнейших этапах сокращается и тем самым упрощается работа всей команды.

Макет веб-сайта играет важную роль в работе frontend-разработчика, потому что на его основе специалист программирует визуальные решения, пользовательские сценарии и функциональность.

Макет помогает frontend-разработчику:

* заранее продумывать техническую реализацию дизайнерских идей;
* сохранять стилистическое единство всех разделов во время вёрстки.
* понять, как элементы ведут себя при изменении размера окна браузера. Это позволяет специалисту спланировать свою работу и ничего не нужно будет переделывать или дополнять.
* понять, как сайт будет выглядеть на разных устройствах. Например, с его помощью можно показать, что на одном экране логотип смотрится хорошо, на другом — размыто.
* предусмотреть возможные изменения. Проект — живой организм, который постоянно меняется. Поэтому в программном коде предусматриваются возможности его быстрого и безболезненного изменения или даже удаления.
* выявить слабые места. Это избавит от проблем в дальнейшем, сделает систему более устойчивой к внешним воздействиям.
* реализовать сложные анимации и визуальные эффекты. Они делают сайт живее, привлекают пользователя, помогают взаимодействовать с сайтом.

**Глава 2. Основные инструменты Frontend разработки веб-сайта**

**2.1 Язык разметки HTML / HTML5**

HTML это сокращение от Hypertext Markup Language (язык гипертекстовой разметки). Из расшифровки названия понятно, что инструмент применяется для управления отображением контента на интернет-странице, его структуризации.

Файлы HTML обычно имеют расширение «.html» или «.htm». Это стандартное расширение для веб-страниц, написанных на языке разметки.

Технология гипертекстовой разметки веб-страниц была предложена в 1989 году британским специалистом Тимом Бернерсом-Ли. Сначала язык применялся для обмена научной рабочей документацией между инженерами института CERN, сотрудником которого был Бернерс-Ли. Немного позднее применение языка HTML было расширено настолько, что он, наряду с такими базовыми элементами, как HTTP и URL лег в основу Всемирной паутины и веб-программирования.

Когда пользователь посещает сайт, браузер «подтягивает» файл HTML с данными о структуре и содержании веб-страницы. Функция HTML состоит в выстраивании внешней базы, фундамента, но сам запуск сайта в функционал не входит. HTML только указывает, где должны располагаться элементы, каков их базовый визуал, где брать стили для элементов и скрипты.

HTML-документ можно составлять в любом редакторе, который есть в операционной системе: Notepad на MS Windows, TextEdit в Mac, Pico на Linux. Браузер для работы HTML–документа желателен, но необязателен. Он нужен для того, чтобы показать отформатированный документ.

Просматривать HTML-страницы можно и без выхода в интернет. Для этого нужно создать несколько HTML-файлов в одной папке, расположить в них гиперссылки и переходить по ним от одного документа к другому.

Что можно и нельзя сделать на HTML.

HTML представляет собой основу внутренней структуры сайта, его базовый каркас. Необходимо учитывать, что этот код является не языком программирования, как, например, Python или C#, а инструментом для разметки гипертекста. С его помощью браузер выстраивает интернет-страницу в виде, который понятен для людей, вырисовывает ее с помощью CSS и добавляя логику через JavaScript. HTML оптимален для начинающих программистов, он прост в освоении, а приобретенные навыки помогут уже в изучении языков программирования.

В HTML-файле можно задавать:

Гиперссылки, списки, формы, разметку страницы, таблицы, абзацы, картинки, видео, заголовки.

Также с помощью HTML можно создать базовый дизайн. Например, установить цвет и шрифт текста на странице или фон для блоков. Использовать только код HTML для оформления веб-страниц не рекомендуется, дизайн будет примитивным и не современным. С CSS же творческий процесс ничем не ограничивается. Тем не менее, ряд функций в настоящий момент приходит в HTML из других, более серьезных инструментов. Например, Drag&Drop (перемещение элементов мышкой) ранее было исключительно в JavaScript, теперь это можно делать и на HTML.

Теги HTML.

HTML-документ в браузере преобразуется в веб-страницу и состоит из набора тегов. Они помогают представлять текст на экране: благодаря им браузер понимает, что он читает не просто текст, а структурированную информацию, разбитую на блоки.

Тег выглядит как набор символов, заключенный в угловые скобки. Символы в скобках обозначают имя тега, которое описывает его функции. Вот несколько примеров:

<h1> Заголовок </h1>, <p>параграф </p>, <i>*курсивный текст* </i>.

Структура тегов в HTML.

Тег – это составной элемент, определяющий разметку структурных блоков. Он открывается, и этим начинает свое действие; и закрывается, обозначая завершение команды. Закрытые и открытые теги различаются только слешем перед именем тега. Теги создают оболочку, в которую помещается содержимое.

Внутри тега могут быть атрибуты – дополнительная информация, которую нужно скрыть из основного текста. Они ставятся только в открывающий тег, между ним и именем тега должен быть пробел, а после него идет знак равенства. Значение атрибута заключается в кавычки. С их помощью можно расширить возможности тегов и обратиться к ним, чтобы узнать подробную информацию.

Есть теги, которые нет необходимости закрывать. Пример: тег переноса строки <br> — он одиночный и закрывать его не нужно. Раньше одиночные теги писались с закрывающим слэшем перед закрывающей скобкой. Например: <br />. В стандарте HTML5 использование закрывающего слеша в одиночных тегах необязательно. Примеры одиночных тегов: <br>, <hr>, <img>.

Помимо атрибутов в тег можно добавлять вложения, эти элементы могут менять стиль текста. Например, можно выделить какое-то слово <strong>**жирным**</strong> шрифтом.

<!DOCTYPE html> –предназначается для указания типа документа, так как браузер может интерпретировать разные версии HTML (например, EXtensible HyperText Markup Language, расширенный язык разметки гипертекста). По умолчанию его всегда включают в начало страницы.

<html> </html> – сообщает браузеру, что это за HTML-документ. Этот тег хранит в себе остальные теги.

<head> </head> – нужен для хранения других элементов, которые помогают браузеру в работе с данными. Внутри него есть метатеги, которые применяются, чтобы сохранять информацию для браузеров и поисковых систем.

<body> </body> – тело документа, в котором находятся все видимые пользователю элементы.

<title> </title> – заголовок веб-страницы. Именно его браузер загрузит как название, а при сохранении страницы в избранное он использует эту фразу как описание закладки.

<img> – помещает изображение в нужное место. Обычно к нему добавляют атрибут src, в котором содержится путь к этому изображению. Атрибуты width, height определяют ширину и высоту изображения в пикселях.

Основная разметка HTML-страницы – это заголовки, параграфы и списки. Они структурируют информацию на странице.

Преимуществом HTML можно считать широкое распространение, код можно написать в любом текстовом редакторе, совместимость с подавляющим числом браузеров, очень легкий синтаксис, можно изучить основы за пару дней, поддержка стандарта консорциумом Всемирной паутины (WWW Consortium), простая интеграция с базовыми языками программирования, такими как PHP.

Недостатками можно назвать, то что HTML не подходит для создания динамических страниц. Для этого может понадобиться JavaScript или PHP, некоторые браузеры медленно осваивают поддержку новых функций, иногда бывает сложно предугадать реакцию старых браузеров на новые теги.

HTML не обрабатывает данные, а только их отображает. То есть с помощью него нельзя выполнить сложение или умножение, можно только показать текст, в котором будет содержаться нужная формула с ответом. Он отвечает за разметку – ограниченный набор действий, который помогает браузеру отображать страницы.

Однако HTML обладает синтаксисом, семантикой и лексикой, поэтому он попадает в категорию декларативных языков веб-программирования.

Особенности HTML5:

Новые элементы, а также возможность использования SVG и математических формул. Эти новшества разработаны для упрощения создания и управления графическими и мультимедийными объектами в сети без необходимости использования сторонних API и плагинов.

Новые семантические элементы, разработанные для того, чтобы обогащать содержимое документа (страницы).

Новые атрибуты, хотя ряд элементов и атрибутов был удалён.

API и DOM стали основными частями спецификации HTML5.

Особенности обработки ошибок вёрстки, поэтому синтаксические ошибки должны рассматриваться одинаково всеми совместимыми браузерами.

Область использования HTML5 вышла далеко за пределы веб-среды интернет: HTML5 применяется также для создания мобильных приложений под Android, iOS, Windows Mobile и даже для создания десктопных приложений для обычных компьютеров.

Список используемых источников

* <https://habr.com/ru/companies/otus/articles/674748/>
* <https://skillbox.ru/media/code/frontend_i_backend/>
* <https://practicum.yandex.ru/blog/professiya-frontend-razrabotchik/>
* <https://atwinta.ru/material/blog/etap-web-razrabotka/>
* <https://optimalgroup.ru/blog/etapy-razrabotki-veb-produkta/>
* <https://dzen.ru/a/Zh6zIrmPUiH7NcG->
* <https://activica.ru/dictionary/chto-takoe-maket-saita.html>
* <https://kcid.ru/life/blog/poleznye-stati/risuem-sayt-sovmestnaya-rabota-veb-dizaynera-i-frontend-razrabotchika/>
* <https://optimalgroup.ru/blog/etapy-razrabotki-veb-produkta/>
* <https://blog.skillfactory.ru/glossary/verstka/>
* <https://practicum.yandex.ru/blog/kak-sozdat-maket-sayta/>
* <https://htmlacademy.ru/blog/middle/frontend-in-design>
* <https://ru.wikipedia.org/wiki/HTML5>
* <https://habr.com/ru/articles/31320/>
* <https://metanit.com/web/html5/1.1.php>
* <https://skillbox.ru/media/code/sozdaem_sayt_na_html/>
* <https://blog.skillfactory.ru/glossary/html5/>
* https://blog.skillfactory.ru/glossary/html/