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 CSS/CSS3, SASS

2.3 Язык программирования JavaScript

2.4 Библиотека React

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

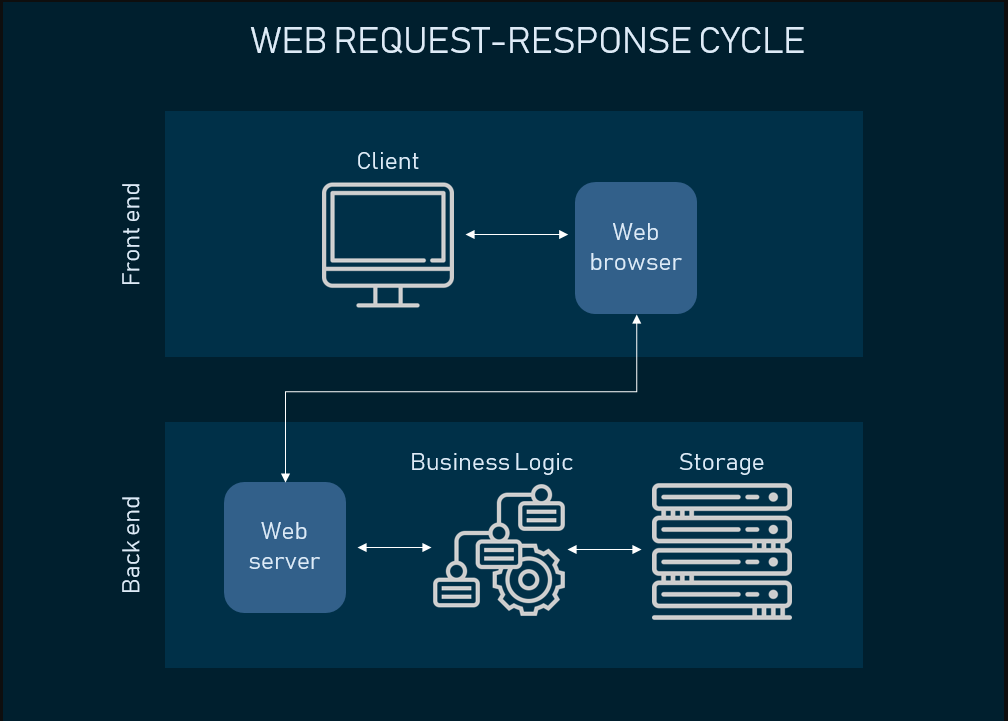
Любой сайт состоит из пользовательской и серверной частей. На странице в интернете вы видите текст, кнопки, панели, изображения и видео. Можете перемещаться по сайту, свободно изучать содержимое веб-сайта. Перед вами — пользовательская часть сайта и результат труда frontend-разработчика: визуализация, 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.

**2.2 Каскадные таблицы стилей CSS/CSS3, SASS**

CSS — это язык описания внешнего вида документа, то есть он отвечает за то, как выглядят веб-страницы: цвет фона и декоративных элементов, размер и стиль шрифтов. Термин расшифровывается как Cascading Style Sheets (каскадные таблицы стилей). CSS взаимодействует с другим языком разметки — HTML, который отвечает за размещение элементов на странице.

Можно представить, что HTML — это такая общая, условная схема, как грубый чертеж планировки квартиры, из которого можно узнать, где находятся двери и окна, а с какой стороны кухня или коридор. В таком случае CSS для сайта — это дизайн-проект, который помогает увидеть более детальную планировку с цветом стен, мебелью и декором.

Для чего нужен CSS.

Допустим, с помощью HTML текст уже расположен в верхней части документа. Задать для него цвет, тип шрифта и его кегль, сделать текст жирным или выделить курсивом можно и без использования CSS. Для этого в исходный код нужно добавить тег, который определяет начертание текста. Например, в этом тексте с помощью тега <b> фрагмент выделен жирным шрифтом:

Первая часть текста набрана обычным шрифтом. А вот эта часть будет выделена жирным.

<!DOCTYPE html>  
<html>  
<head>  
<title>**Жирный текст с помощью тега b**</title>  
</head>  
<body>  
<p>Первая часть текста набрана обычным шрифтом. <b>**А вот эта часть будет выделена жирным.**</b></p>  
</body>  
</html>

Использование тегов для форматирования текста в HTML загромождает исходный код, усложняет его, а значит, вероятность допустить в нем ошибку становится выше. Для того чтобы этого избежать, создали отдельный язык для стилевой разметки — CSS. Кроме размера и цвета шрифтов, этот язык разметки регулирует разделение заголовков, подзаголовков и основного текста, размер полей и отступов, отдельные цветовые фреймы для выделения текста, цвет основного фона, шапки и подвала.

Как устроены таблицы стилей.

CSS, как и любой язык, имеет свой синтаксис. В нем есть правила — значения, которые определяют внешний вид элементов. CSS-правило состоит из селектора, CSS-свойств и их значений:

Селекторы — это метки, которые помогают браузеру понять, к какой части HTML-кода нужно применить заданные параметры.

CSS-свойства — это определенные параметры оформления, например цвет элемента или текста (color) или цвет фона (background).

Значение — это просто значение, оно выражается текстом или числом, например черный (black).

селектор {  
свойство: значение;  
}  
CSS-правила в коде заключаются в фигурные скобки {…}. Перед открытием скобки обязательно нужно указать селектор, к которому относится это правило.

В примере селектором является <p>, и он выбирает все теги с именем <p>, color — это CSS-свойство а black — значение CSS-свойства. Связка «свойство: значение» называется блоком объявления стилей. Внутри него свойство отделяется от значения двоеточием, а один блок от другого отделяет точка с запятой.

Таблицы называются каскадными, потому что работают по принципу каскада — то есть правило, прописанное ниже, считается приоритетным. Например, если в нашем примере под значением фонового цвета мы пропишем еще одно значение color: red, то цвет текста будет красным, а не черным.

p {  
color: black  
background: #fff  
color: red  
}

Подключение стилей CSS.

Чтобы использовать CSS совместно с HTML, можно выбрать один из способов:

1. Весь код, написанный на CSS, прописывается в отдельном внешнем файле с расширением .css. Его подключают к HTML-странице при помощи тега <link href> — это служебный тег, который на странице не будет видно:

<head>  
<link href=”style.css” rel=”stylesheet”>  
</head>  
Атрибут rel со значением stylesheet указывает, что в документе применяются именно стили текста. Это важно, так как, кроме применения стилей, тег <link> используется еще во множестве разных значений.

2. Прописать стили CSS внутри конкретного тега с помощью атрибута style:

<p style= «color: black; background: #fff»> Добавление стиля с помощью атрибута style</p>

В этом случае можно не использовать фигурные скобки, потому что и без них браузеру будет понятно, к какому элементу относится правило.

Float — это свойство, которое определяет, по какой стороне будет выравниваться элемент: по правой или по левой. Например, чтобы выровнять элемент по левому краю, подписываем CSS-свойство: float: left;

Этот способ удобно использовать, например, для обтекания картинок текстом, но некоторое время назад с помощью float верстали практически все. Например, это был распространенный метод верстки меню. Чтобы пункты меню образовывали линию, каждому элементу присваивался float: left или float: right, чтобы они выстраивались друг за другом:

Flexbox.

Flexbox пришел на смену float в 2011 году. Его преимущество заключалось в том, что блоки стали более гибкими. Элементы сжимались и растягивались, заполняя пространство, их без проблем можно было выровнять по вертикали и горизонтали. Например, чтобы выстроить элементы меню по центру, достаточно прописать в коде: justify-content: center;

Расположение элементов по центру с помощью CSS flexbox.

Один элемент можно растянуть так, чтобы блоки заняли всю ширину страницы. Для этого для блока child (селектор класса) прописывается CSS-свойство: flex-grow: 1;

Растягивание элементов на ширину страницы с помощью CSS flexbox

Посмотреть все возможности flexbox можно в [интерактивной шпаргалке](https://tpverstak.ru/flex-cheatsheet/), с помощью которой можно выбрать необходимое значение и скопировать фрагмент кода.

Grid.

Grid помогал решать многие задачи, с которыми сталкивались разработчики на протяжении долгого времени. Задачи решались методами, которые изначально для этого не предназначались. Главным нововведением стало расположение элементов относительно нескольких осей — по вертикали и по горизонтали. Растянуть элементы не несколько строк или несколько столбцов можно, присвоив каждому значения a, b, c, d и так далее. Затем прописываем в CSS-коде:

grid-template-areas:

“a b c d d”  
“f f g d d”

В разметке значение «a» — это элемент 1, «b» — элемент 2, «c» — это элемент 3, а «d» повторяется дважды, так как занимает две колонки.

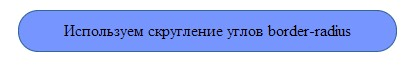
CSS3.

CSS3 — это новое поколение CSS, которое до сих пор находится в доработке. С его помощью можно анимировать элементы без использования JavaScript, создавать тени и градиенты, скруглять углы блоков.

Например, для скругления элемента header используется свойство border-radius:

header {  
background-color:#7695FE;  
border: thin #336699 solid;  
padding: 10px;  
margin: 10px;  
text-align: center;  
border-radius: 25px;  
}

Проблема заключается в том, что не все браузеры одновременно начинают поддерживать нововведения CSS3, поэтому в браузере, который поддерживает свойство border-radius, кнопка будет выглядеть так:



А другой может проигнорировать это свойство:



Из-за этих нестыковок верстальщики, которые работают с CSS3, установили правило: «Веб-сайты не должны выглядеть одинаково на всех браузерах, и это нормально».

SASS.

Sass — это метаязык (язык для описания другого языка), который упрощает и ускоряет написание CSS-кода. Его часто называют препроцессором CSS — это означает, что Sass имеет свой синтаксис, из которого генерирует CSS-код, понятный любому браузеру.

Препроцессор Sass помогает:

сделать CSS-код понятнее и проще. Его легче масштабировать, обновлять и поддерживать;

расширить функциональность. С помощью Sass можно использовать CSS-константы, встроенные функции, вложенные правила, примеси (смешанные стили), наследование и так далее;

избежать многократного повторения одинаковых фрагментов кода. Это экономит время разработчика, уменьшает объем файлов стилей и ускоряет обработку страниц.

Как работает Sass

Отличительная особенность Sass — наличие двух синтаксисов.

1. SASS не использует фигурные скобки, характерные для стилей CSS. Вместо них применяются отступы. Расширение файла — .sass.

Пример кода:

&:disabled

color: $mdc-button-disabled-ink-color

cursor: default

pointer-events: none

2. SCSS использует фигурные скобки. Расширение файла — .scss.

Пример кода:

&:disabled {

color: $mdc-button-disabled-ink-color;

cursor: default;

pointer-events: none;

}

}

При этом оба диалекта поддерживают не только стандартные CSS-комментарии, но и построчные примечания, отделяемые от кода двойным слэшем //:

/\* это пример обычного CSS-комментария \*/

.main-header {

background-color: $primary\_1; // это пример построчного комментария

}

Браузеры не понимают код Sass — они распознают и отображают только стили CSS. Для преобразования используется специальное приложение — препроцессор, который переводит конструкции в понятный браузеру формат. В готовый CSS-код не включаются построчные комментарии. В ходе компиляции в CSS передаются только стандартные примечания, отделяемые от кода символами /\* \*/.

Переменные в Sass

Переменные применяются для хранения данных и значений для повторного использования. Sass поддерживает несколько типов переменных:

строки;

списки;

числовые значения;

пустые значения null;

цвета;

логические (булевы) значения.

Проиллюстрируем использование переменных на примере. На скриншотах видно, как содержимое файла .scss определяет код CSS и HTML. Используются переменные:

myFont — задает шрифт Helvetica;

myColor — устанавливает красный цвет;

myFontSize — определяет размер шрифта, 18 кегль;

myWidth — задает ширину контейнера, 680 пикселей.

Слева — переменные в файле .scss, справа — представление в браузере.

**2.3 Язык программирования JavaScript**

JavaScript (JS) — это интерпретируемый язык программирования, который используют для написания frontend- и backend-частей сайтов, а также мобильных приложений. Часто в текстах и обучающих материалах название языка сокращают до JS. Это язык программирования высокого уровня, то есть код на нем понятный и хорошо читается.

JS поддерживают все популярные браузеры. Во frontend-части сайтов язык используют для создания интерактива (анимаций, всплывающих форм, автозаполнения), так как он связан с HTML и CSS и может ими манипулировать. В backend-части с языком JavaScript работают на платформе Node.js. С ее помощью, например, разрабатывают серверные веб-приложения и подключают библиотеки. В поисковике Google на JavaScript работает строка автозаполнения, а Netflix, Uber, eBay используют его в своем backend. Уже 6 лет JS — самый популярный язык среди разработчиков по версии GitHub.

Особенности JavaScript

JavaScript простыми словами называют языком скриптов или сценариев. Скрипты — это набор инструкций, которые выполняются при загрузке страницы. Браузер самостоятельно интерпретирует код на JavaScript, для этого даже не требуется компиляция (перевод языка программирования в машинный код).

Скрипты можно прописать внутри кода страницы или подключить к HTML отдельным файлом. Например в стандартной разметке index.html JS-код прописывают внутри тега script, помещенного в тег body:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Документ</title>

</head>

<body>

<h1>Привет!</h1>

<script>

console.log('Привет, мир!');

</script>

</body>

</html>

Тогда в браузере благодаря инструкции console.log появится слово «Привет!».

вывод текста с помощью JavaScript

Но обычно для этого создается отдельный файл с расширением .js. Его называют script.js и прописывают код в нем:

console.log(‘Hello world!’)

вывод в консоль JS

А в основном коде уже внутри тега script прописывают путь к этому файлу:

<body>

<h1>Привет!</h1>

<script src="js/script.js"></script>

</body>

</html>

JS — мультипарадигменный язык, так как он поддерживает разные парадигмы (стили) программирования и особенности:

Объектно-ориентированный — представляет программу как систему отношений между объектами — наборами данных и функций.

Императивный — в этом стиле программирования используют инструкции, которые выполняются последовательно, а для доступа к данным используются переменные, значение которых может быть переписано в ходе вычисления.

Функциональный — в нем вычисление основано на математических функциях. Так как функциональное программирование не опирается на переменные, которые могут менять значение, результаты запросов будут всегда одинаковыми.

Динамическая типизация: Переменные в JavaScript не требуют объявления типа данных. Они могут изменять свой тип в процессе выполнения, что делает код более гибким, но также требует внимательного управления типами.

Обработка событий: JavaScript является мощным инструментом для обработки событий на веб-страницах. Он может реагировать на действия пользователей, такие как клики мышью, нажатия клавиш и другие.

Манипуляция DOM: JavaScript может изменять структуру и содержимое веб-страницы, обеспечивая динамическое обновление контента без перезагрузки страницы.

Асинхронные операции: JavaScript позволяет выполнять асинхронные операции, такие как загрузка данных с сервера, без блокирования выполнения других задач.

Серверный JavaScript: С помощью платформы Node.js JavaScript может быть использован на серверной стороне для создания высокопроизводительных и масштабируемых веб-приложений.

Множество фреймворков и библиотек: JavaScript имеет обширное сообщество разработчиков и большое количество библиотек и фреймворков, таких как React, Angular, Vue.js, и другие, что упрощает создание сложных веб-приложений.

Кроссплатформенность: JavaScript может быть использован для разработки веб-приложений на различных платформах, включая веб, мобильные устройства и даже десктоп.

Открытый стандарт: JavaScript базируется на открытых стандартах ECMAScript, что обеспечивает его интероперабельность и расширяемость.

Веб-страницы частично обрабатываются с помощью JavaScript на компьютере пользователя. Это снижает нагрузку на сервер: часть операций выполняется без запросов к нему, что экономит время и трафик.

Где применяется JavaScript

Клиентская часть веб—приложений (frontend). Это интерфейс страницы, то есть всё, что видит пользователь: контент, кнопки, формы обратной связи, меню. С помощью JS интерфейс реагирует на действия пользователя (клики мыши, нажатия клавиш), также язык отвечает за запоминание данных и автозаполнение форм.

Серверная часть веб—приложений (backend). Серверный код пишут на платформе Node.js. На JS работают, например, запросы AJAX (asynchronous javascript and XML), которые отправляются на сервер в фоновом режиме, без перезагрузки веб-страницы, и push-уведомления — всплывающие сообщения в браузере, которые реализуются с помощью технологии Comet. Такие уведомления приходят со специального Comet-сервера, который постоянно поддерживает соединение с браузером. Как раз с помощью JavaScript устанавливается это соединение.

Мобильные приложения на Android, iOS, Windows Mobile — когда нужно кросс-платформенное приложение или адаптация веб-приложения, а языков Kotlin (для Android) и Swift (для iOS) недостаточно, то используется JavaScript.

Для чего нужен JS

В веб-приложениях HTML отвечает за разметку страницы. Расположение контента, внешний вид интерфейса описывается с помощью CSS, а JavaScript отвечает за интерактивность и функционал этого интерфейса. Вот примеры задач, которые решает JS:

Взаимодействие с пользователем. Всплывающие окна с полем для ввода почты или других данных, уведомления об использовании cookie.

Обработка данных HTML. JS помогает заполнить необходимые поля в формах, проверять буквенные и числовые значения, а также управлять HTML-элементами на странице. Например, в конструкторе лендингов Tilda можно перетаскивать блоки и по клику менять шрифты, цвета текста, фонов и другие значения.

Анимация. Это не только движущиеся объекты на странице и кнопки, но и карусель в галерее или плавная прокрутка страницы. Например, Pinterest с помощью JavaScript создал эффект бесконечной ленты.

Математические вычисления. С помощью JS-скриптов на сайтах реализованы калькуляторы, например подсчет суммы заказа в корзине интернет-магазина.

Мобильные приложения: С использованием фреймворков, таких как React Native и NativeScript, JavaScript может быть использован для создания мобильных приложений для разных платформ.

Серверный JavaScript: используется на серверной стороне с использованием платформы Node.js для создания высокопроизводительных и масштабируемых веб-приложений.

Обработка событий: JavaScript используется для обработки событий, таких как клики мышью, нажатия клавиш, отправка форм и другие действия пользователя. Это позволяет создавать реактивные и отзывчивые интерфейсы.

Как работает JavaScript в браузере

Действия пользователя на странице вызывают события: клик на кнопке запускает анимацию, захват мышкой и перемещение курсора двигают объект по странице. Каждое из этих действий выполняется с помощью скриптов, написанных на JavaScript. У них есть определенный алгоритм работы:

Пользователь совершает действие — нажимает на кнопку «Зарегистрироваться».

Браузер фиксирует выполнение определенных условий — клик мыши в конкретной области на странице. За реакцию на действие отвечает команда onclick, которая запускает прописанный JS-код.

Затем запускается JS-код, который прописан для этих условий, — анимация нажатия кнопки и открытие формы регистрации.

Страница полностью или частично обновляется — в зависимости от настроек форма может открыться в этой же или в соседней вкладке.

Недостатки JavaScript

Несмотря на множество преимуществ, JavaScript также имеет некоторые недостатки:

Браузерные несовместимости: Разные браузеры могут по-разному интерпретировать и поддерживать JavaScript. Это может вызвать проблемы с кросс-браузерной совместимостью и потребовать дополнительного кода для обработки различий.

Безопасность: JavaScript выполняется на стороне клиента, что делает его уязвимым для атак, таких как внедрение вредоносного кода (XSS). Разработчики должны быть осторожными и следовать лучшим практикам безопасности.

Производительность: Несмотря на улучшения в исполнении JavaScript-кода, он может быть менее эффективным по сравнению с некоторыми другими языками, особенно для вычислительно сложных задач.

Глобальная область видимости: JavaScript имеет одну глобальную область видимости, что может вызвать конфликты имен переменных между разными частями кода.

Callback Hell: Вложенные колбэки в асинхронном коде могут создавать так называемое «Callback Hell» — сложные и труднопонимаемые структуры кода.

Долгая история развития: JavaScript имеет долгую историю развития, что означает наличие устаревших конструкций и несовместимостей между разными версиями языка.

Отсутствие строгой типизации: Для некоторых разработчиков динамическая типизация JavaScript может быть источником ошибок и затруднить отладку.

Ограниченный доступ к ресурсам компьютера: Из соображений безопасности JavaScript имеет ограниченный доступ к ресурсам компьютера пользователя, что может затруднить выполнение некоторых задач.

Зависимость от интернета: Большинство веб-приложений на JavaScript требуют постоянного соединения с интернетом, и они могут быть менее функциональными в оффлайн-режиме.

Сложность асинхронного кода: Работа с асинхронными операциями и обещаниями (Promises) может быть сложной и вызвать проблемы, такие как утечки памяти.

2.4 Библиотека React

React.js — это библиотека для языка программирования JavaScript с открытым исходным кодом для разработки пользовательских интерфейсов. Она помогает быстро и легко реализовать реактивность — явление, когда в ответ на изменение одного элемента меняется все остальное.

У React открытый исходный код и мощное сообщество. Это одна из самых популярных библиотек для веб-разработки.

Кто пользуется React

React применяют фронтенд-разработчики. В модели MVC, название которой расшифровывается как Model-View-Controller, интерфейс — это View, представление, внешнее отображение, с которым взаимодействует пользователь, та часть сайта или приложения, которая видна человеку. Специалисты, которые с ней работают, в частности, используют React.js. Также с React могут работать верстальщики, тестировщики и другие специалисты, задействованные в создании веб-интерфейсов.

Для чего нужен React

React используют для создания одностраничных и многостраничных приложений, разработки крупных сайтов. Например, с его помощью написан стриминговый сервис Netflix и реализованы новостные ленты крупнейших социальных сетей. Библиотека предназначена:

для создания функциональных интерактивных веб-интерфейсов, работая с которыми, не нужно постоянно обновлять страницу;

быстрой и удобной реализации отдельных компонентов и страниц целиком — элементы в React легко использовать повторно;

легкой разработки сложных программных структур — их просто описывать, если использовать реализованный в React подход;

доработки новой функциональности с любым изначальным стеком технологий: библиотека не зависит от остального инструментария и будет хорошо работать, на чем бы ни был написан код;

разработки одностраничных и многостраничных приложений (SPA и PWA). Это приложения, которые функционируют как программы и веб-сервисы и имеют соответствующий интерфейс;

работы с серверной частью сайта или разработки мобильных приложений. В таких случаях React используют совместно с инструментами, адаптирующими веб-технологии под другие цели.

Особенности библиотеки React

React имеет ряд особенностей, которые делают его гибким и мощным инструментом. Мы разберем некоторые из них.

React не является полноценным фреймворком, являясь библиотекой для создания пользовательских интерфейсов. Важно понимать разницу между фреймворками и библиотеками:

Фреймворк — это комплексное решение, которое предоставляет набор инструментов, библиотек и правил для разработки приложений. Фреймворк часто определяет структуру приложения, а разработчику предоставляется меньше свободы в выборе инструментов и архитектурных решений. Примеры фреймворков включают Angular и Ember.

Библиотека — это набор функций и компонентов, которые помогают в определенных задачах, но не навязывают общую архитектуру приложения. Разработчик имеет большую свободу выбора остальных инструментов и архитектурных решений. React — это пример библиотеки, которая сосредотачивается на создании пользовательских интерфейсов.

Таким образом, React предоставляет инструменты для создания компонентов пользовательского интерфейса, управления состоянием и взаимодействия с DOM, но не навязывает конкретную архитектуру приложения. Разработчики могут использовать React в комбинации с другими библиотеками и инструментами по своему усмотрению, что делает его гибким и позволяет создавать разнообразные типы веб-приложений.

Декларативность

Декларативный стиль означает, что достаточно один раз описать, как будут выглядеть результаты работы кода — элементы в разных состояниях. Ему не нужно фокусироваться на способах достижения результатов: большую часть задач выполнит библиотека. React.js будет автоматически обновлять элементы в зависимости от условий, главная задача — грамотно описать их. Удобный и понятный подход облегчает написание и отладку кода.

Виртуальное DOM-дерево

Любой веб-интерфейс основан на HTML-документе и CSS-стилях, к которым подключен код на JavaScript. Структура HTML-документа, точнее его модель, называется DOM-деревом (DOM расшифровывается как Document Object Mode, объектная модель документа). Это древовидная модель, в которой в иерархическом виде собраны все используемые на странице элементы.

React включает в себя механизм, который управляет процессом рендеринга компонентов. Рендеринг — это процесс создания визуального представления компонента, которое отображается в браузере или мобильном приложении.

Особенность React в том, что он создает и хранит в кэше виртуальное DOM-дерево — копию DOM, которая изменяется быстрее, чем реальная структура. Это нужно для того, чтобы быстро обновлять страницы. Если пользователь выполнит действие или наступит какое-либо событие, DOM должна измениться, так как изменятся объекты на странице. Но реальная объектная модель может быть огромной, ее обновление — медленный процесс. Поэтому React работает не с ней, а с виртуальной копией в кэше, которая весит меньше.

Когда происходит событие, из-за которого код должен обновить объект, изменение быстро отображается в виртуальном DOM. После этого обновляется реальная объектная модель. Для пользователя это значит, что изменения на странице отобразятся мгновенно, а не после долгой загрузки.

Обновление DOM по частям

Чтобы улучшить быстродействие, React обновляет DOM не полностью. Он хранит в памяти две облегченных копии: актуальную и предыдущую. Когда что-то обновляется, библиотека сравнивает версии между собой и изменяет только ту часть дерева, которая действительно поменялась. Это нужно, чтобы не перезагружать DOM целиком и не замедлять страницу. Подход кажется сложным, но он важен для оптимизации загрузки.

Возможность повторно использовать компоненты

React основан на компонентах — отдельных элементах веб-интерфейса. Компоненты инкапсулированы, то есть самостоятельны: в каждом из них размещены все необходимые методы и данные. Подробнее про инкапсуляцию можно почитать в нашей статье про ООП: это один из ключевых критериев объектно-ориентированного подхода. В случае с компонентами React инкапсуляция также означает, что состояние элемента хранится в нем самом, а не в отдельном объекте.

Инкапсулированные самостоятельные компоненты можно использовать повторно, размещать в другом месте кода, в ином разделе или на другой странице. Данные можно переносить по всему приложению, использовать вне DOM конкретной страницы. Это ускоряет разработку и сокращает количество действий для создания функционирующего интерфейса. Благодаря отсутствию сложных зависимостей инкапсуляция также облегчает отладку.

Нисходящий поток данных

Компоненты могут передавать свойства и данные друг другу, но только в одном направлении — от «родительских» к дочерним. Это помогает реализовать четкую иерархию, облегчает отладку. Однонаправленный поток данных означает, что программист всегда может понять, откуда именно к элементу поступили данные.

Еще одна особенность потока данных — отсутствие возможности изменять свойства напрямую. Для этого существуют специальные callback-функции. Так называются части исполняемого кода, которые передаются от одного компонента другому в виде параметра функции. Это позволяет поддерживать стабильность: свойства оказываются неизменяемыми после создания, а callback-функция как бы пересоздает их. В результате использования коллбэков поток данных выглядит так: свойства и информация передаются сверху вниз, а события происходят снизу вверх.

Синтаксис JSX

JSX расшифровывается как JavaScript XML. Это расширение языка JavaScript, которое помогает описывать HTML-подобные элементы с помощью кода на React. С помощью синтаксиса на React создают компоненты страницы и гибко управляют ими.

Несмотря на то что элементы похожи на HTML, это по-прежнему язык JavaScript с возможностью быстро и легко изменять DOM с помощью кода. И все же JSX воспроизводится как HTML: по сути разработчик описывает нужный компонент на языке разметки, а тот остается JavaScript-объектом с широкой функциональностью. Это удобно, упрощает программирование, но может запутать начинающих.

React Hooks

В старых версиях управлять состояниями можно было с помощью классов — специальных конструкций, о которых можно подробнее прочесть в статье про ООП. Сейчас в React.js есть поддержка хуков — так называются специальные функции-«крючки», которые «цепляются» за состояние элемента или за метод. Изменение состояния или вызов метода «тащит» за собой эти функции, и они автоматически выполняются — это помогает избежать использования классов, облегчает и упрощает написание кода.

В React есть встроенные хуки, но разработчик может создавать и собственные — это помогает повторно использовать код и уменьшить его количество.

React Developer Tools

Так как React.js — очень популярная технология, его создатели разработали бесплатные расширения для браузера с инструментами для проверки и отладки. Фронтендеры часто пользуются консолью и панелью разработчика в браузере, чтобы проверить, как работает их код. React Developer Tools облегчают задачу и расширяют возможности. Например, можно просматривать прямо в браузере компоненты с большим уровнем вложенности и не искать их в коде долгое время. React Developer Tools существуют для новых версий популярных браузеров Firefox и Google Chrome.

Преимущества React

Популярность

Это один из трех самых распространенных инструментов для фронтенд-разработки. Кроме него, популярны Vue.js и Angular, но первый пока не так распространен, а второй намного сложнее в изучении. Еще одна популярная технология — jQuery, но она постепенно уходит в прошлое. На React написано множество сайтов и приложений. Ее используют такие крупные компании, как Яндекс, Uber, Сбербанк, Авито, BBC, Airbnb, Netflix и другие. Вакансий много, и они высоко оплачиваются даже на начальных уровнях.

Огромное сообщество

Это еще одно следствие популярности: React-разработчиков много, в том числе опытных, многие из них пишут туториалы и статьи, помогают новичкам влиться в процесс разработки. В ходе обучения начинающий всегда может обратиться на тематические ресурсы, где ему постараются помочь. Людей в сообществе хватает, можно получить ответы на большинство вопросов. Документация поддерживается и обновляется, существует в том числе на русском языке.

Развитая экосистема

Создатели React активно побуждали пользователей вкладываться в развитие проекта, и энтузиасты писали свои технологии для совместного использования с библиотекой. В результате сейчас сформировалась целая экосистема: десятки библиотек и других инструментов, которые можно использовать в разработке. Это делает процесс проще и легче, позволяет решать огромное количество задач.

Читайте также

Что нужно знать, чтобы начать работать frontend-разработчиком?

Простота создания

С помощью компонентов проекта можно быстро и легко собрать интерактивный, чутко реагирующий на любые изменения интерфейс сайта или приложения любой сложности. Это намного проще, чем прописывать реакции на всевозможные события вручную, и снижает количество ошибок. Так как компоненты можно использовать повторно в любом месте кода, задача становится еще легче. Например, для создания сложного многостраничного сайта или приложения не нужно многократно писать один и тот же компонент.

Реактивность

Это одно из ключевых преимуществ проекта, вынесенное в название. Библиотека реагирует на обновление компонента и автоматически отображает его изменения в дереве документа. Изменения могут происходить в ответ на действия пользователя, какие-то внешние перемены или другие события. В результате сайты и приложения становятся более привлекательными для пользователя.

Эффективность

Благодаря виртуальному DOM библиотека экономит ресурсы. Чтобы изменить состояние элементов, не нужно полностью перезагружать все DOM-дерево, лишний раз тратить трафик пользователя и нагружать браузер. Изменяются только конкретные элементы, это происходит через виртуальное DOM-дерево — сайты и приложения становятся «легче» и удобнее.

Высокая скорость

Сайты и приложения, написанные на React, быстрые и отзывчивые благодаря виртуальному DOM. Дерево занимает меньше места, быстрее обновляется.

Также ускоряется работа программистов. Компоненты легко создавать и переиспользовать, между ними уже продумана логика сообщения. Главная задача — грамотно описать состояния. Не нужно прописывать всю логику и принципы с нуля, это за разработчика уже сделали создатели React.

Удобная отладка

Ошибок с React бывает меньше, чем при написании кода на «чистых» языках без библиотек. Это происходит, потому что многое реализуется автоматически, соответственно, уменьшается влияние «человеческого фактора». Но даже если при разработке программист допустил ошибку, ее довольно легко отследить и исправить благодаря четкой логике потока данных, дополнительным инструментам, понятному синтаксису. Код на React легко читать, понимать и отлаживать.

Недостатки React

Запутанность синтаксиса

Технология JSX удобная и широко используется, но сначала она может вызвать сложности в изучении. Людям непривычно «смешивать» друг с другом HTML и JavaScript, поэтому в синтаксисе легко запутаться и ошибиться. Со временем путаница и ошибки проходят — к этой особенности нужно просто привыкнуть.

Неполное соответствие MVC

React отвечает только за часть View. Поэтому для реализации модели потребуется подбирать стек из разных технологий. React можно использовать практически с любым стеком.

Это одна из причин, по которой React иногда называют библиотекой, а не фреймворком: по одному из определений фреймворк должен позволять полностью реализовать ту или иную модель.

Трудности с SEO-оптимизацией

React-компоненты намного сложнее оптимизировать в рамках SEO. SEO — это поисковая оптимизация, проработка сайта таким образом, чтобы он соответствовал требованиям поисковых систем. Если сайт находится на первой странице поиска по тематическим запросам, его посетят больше пользователей.

Чтобы поисковая система «заметила» сайт, страницы должен проиндексировать поисковый робот. Роботу проще индексировать HTML. React-компоненты, написанные на JavaScript, индексируются тяжелее и медленнее, поэтому страницы сложнее оптимизировать.

Как начать пользоваться React

Для начала новичку нужно на достаточном уровне изучить JS. Кроме «чистого» JavaScript, React поддерживает TypeScript. Но для работы с React знать TypeScript необязательно.

После изучения JavaScript новичок может начинать работу с проектом. У официального сайта есть русская версия, на ней подробно описаны способы, как это можно сделать. Можно воспользоваться онлайн-песочницей или подключить документ с библиотекой к страницам сайта — для тренировки и обучения подойдет любой вариант.

На ранних этапах начинающему разработчику может хватить новых версий документации на русском языке. Чтобы получить полноценную практику и глубоко изучить библиотеку, потребуется помощь наставника.

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/>
* <https://blog.skillfactory.ru/glossary/css/>
* <https://blog.skillfactory.ru/glossary/sass/>
* <https://blog.skillfactory.ru/glossary/javascript/>
* https://blog.skillfactory.ru/glossary/react/