**Frontend Developer**

Roadmap для понятного изучения

# Изучите основы вёрстки — HTML, CSS.

Это основа любой веб-страницы. HTML используется для создания структуры страницы, включая заголовки, параграфы, ссылки, изображения и многое другое. CSS отвечает за стилизацию HTML-элементов.

Ссылки:

1. Курс <https://www.freecodecamp.org/learn>
2. <https://code.mu/ru/markup/manual/html/>
3. <https://code.mu/ru/markup/manual/css/>

# Изучение Java script

Язык программирования, который добавляет интерактивность и логику на веб-страницы. С помощью JS можно обрабатывать события, изменять DOM-дерево, взаимодействовать с сервером через API и многое другое.

Ссылки:

1. Курс <https://www.freecodecamp.org/learn>
2. <https://code.mu/ru/>

# Изучение основ GIT

Git необходим для работы в команде и управления версионностью кода. Нужно освоить базовые команды (commit, push, pull, merge) и работу с удалёнными репозиториями, такими как GitHub или GitLab.

Ссылки:

1. <https://git-scm.com/book/ru/v2>

# Dom

**DOM** — это интерфейс, который браузеры используют для работы с HTML и XML документами. Когда страница загружается в браузер, HTML-код превращается в DOM-дерево, которое представляет структуру документа в виде узлов. Эти узлы можно программно изменять с помощью JavaScript.

Ссылки:

1. <https://learn.javascript.ru/dom-nodes>

# JavaScript-фреймворки: React, Vue, Angular — важно знать хотя бы один из них. CSS-фреймворки

Фреймворки облегчают создание сложных и интерактивных веб-приложений, предоставляя готовую структуру и инструменты для работы с компонентами, управлением состоянием и роутингом. Они ускоряют разработку и упрощают поддержку кода.

**Почему это важно:**

* **Организация кода**: Фреймворки предлагают определённую архитектуру, которая помогает организовать код, особенно в больших проектах.
* **Компонентный подход**: Вместо создания веб-страниц как набора отдельных частей, фреймворки позволяют вам работать с повторно используемыми компонентами, что делает код более модульным и управляемым.

Ссылки:

1. Курс <https://www.freecodecamp.org/learn>
2. https://bootstrap-5.ru/

# Управление состоянием: Redux, MobX, Vuex для более сложных приложений

Когда приложение становится сложным, с большим количеством взаимодействующих между собой компонентов и данных, управление состоянием становится критической задачей. Без правильного подхода можно легко потеряться в передаче данных и обработке изменений.

**Почему это важно:**

* **Централизованное управление состоянием**: Redux, MobX и Vuex предоставляют одно место для хранения всех данных приложения, что делает код более предсказуемым и облегчающим отладку.
* **Отслеживание изменений**: Эти инструменты позволяют чётко контролировать, когда и как данные изменяются в приложении, обеспечивая более чистую и управляемую архитектуру.
* **Совместная работа компонентов**: Использование менеджеров состояния решает проблемы передачи данных между множеством компонентов, что упрощает разработку сложных приложений.

Ссылки:

1. <https://redux.js.org/>
2. <https://mobx.js.org/>
3. [Vue Mastery](https://www.vuemastery.com/vuex/)

# Инструменты сборки: Webpack, Parcel, Vite для сборки и оптимизации кода

Инструменты сборки используются для оптимизации кода и управления зависимостями. Они собирают ваш JavaScript, CSS и другие ресурсы в оптимизированный пакет, который можно легко развернуть на сервере.

**Почему это важно:**

* **Оптимизация кода**: Инструменты сборки уменьшают размер ваших файлов (минификация, устранение дубликатов) и делают ваш код быстрее для загрузки и выполнения в браузере.
* **Модульная разработка**: Webpack, Parcel и Vite позволяют использовать современные JavaScript-модули (import/export), что делает разработку более организованной.
* **Поддержка современных возможностей**: Эти инструменты позволяют вам использовать новейшие функции JavaScript, даже если браузеры ещё их не поддерживают напрямую, благодаря таким инструментам, как Babel (транспиляция).
* **Горячая перезагрузка**: Webpack и другие сборщики часто поддерживают «горячую замену модулей» (hot module replacement), что делает процесс разработки быстрее и удобнее, так как изменения моментально применяются без необходимости перезагрузки всей страницы.

Ссылки:

1. Курс <https://www.freecodecamp.org/learn>

# Адаптивный и кроссбраузерный дизайн

Важно уметь создавать сайты, которые корректно отображаются на разных устройствах (мобильных, планшетах, компьютерах) и работают во всех современных браузерах.

Ссылки:

1. [Отзывчивый веб-дизайн | Coursera](https://www.coursera.org/learn/responsive-web-design)

# Инструменты разработки

1. **DevTools** (в Chrome, Firefox) — панели разработчика для отладки и анализа производительности веб-страниц.
2.  **VS Code** — один из самых популярных текстовых редакторов для разработчиков, с множеством расширений для работы с веб-технологиями

Ссылки:

1. Курс <https://www.freecodecamp.org/learn>

# Дополнительно:

1. Базовые знания UX/UI

Ссылки:

* 1. [Udemy Free Resource Center: Find Free Online Courses | Udemy](https://www.udemy.com/ru/courses/free/)

Понимание принципов пользовательского опыта (UX) и пользовательского интерфейса (UI) поможет создавать удобные и привлекательные интерфейсы. Нужно знать о принципах визуальной иерархии, доступности, навигации и взаимодействии пользователя с интерфейсом.

1. **Тестирование** — основное внимание уделяется написанию тестов (Jest, Mocha, Cypress).
2. **Typescript** — расширение JavaScript, которое добавляет строгую типизацию.

расширение JavaScript, которое добавляет строгую типизацию

Ссылки:

1. Курс <https://www.freecodecamp.org/learn>