|  |  |
| --- | --- |
| ДИСЦИПЛИНА | Создание программного обеспечения |
| ИНСТИТУТ | Институт перспективных технологий и индустриального программирования |
| КАФЕДРА | Кафедра индустриального программирования |
| ВИД УЧЕБНОГО МАТЕРИАЛА | Лекция |
| ПРЕПОДАВАТЕЛЬ | Макиевский Станислав Евгеньевич |
| СЕМЕСТР | 3 семестр, 2024-2025 гг. |

# 

## Лекция №6

**Технологии frontend разработки. Профессия frontend-разработчика.**

**Кто такой Frontend-разработчик?**

Frontend-разработчик - это специалист, который занимается созданием клиентской части веб-приложений и сайтов. Он отвечает за визуальное представление и взаимодействие с пользователем. Frontend-разработчики работают с языками программирования, технологиями и инструментами, которые позволяют создавать интерактивные и удобные интерфейсы.

Навыки Frontend-разработчика

* HTML и CSS: Frontend-разработчик должен быть знаком с языками разметки HTML, который определяет структуру веб-страницы, и CSS, который отвечает за её стилевое оформление.
* JavaScript: Это основной язык программирования для Frontend-разработки. JavaScript позволяет создавать интерактивные элементы, управлять поведением страницы и взаимодействовать с пользователем.
* Фреймворки и библиотеки: Frontend-разработчики используют различные фреймворки и библиотеки, такие как React, Angular или Vue.js, чтобы упростить процесс разработки и повысить производительность.
* Работа с API: Frontend-разработчики часто взаимодействуют с сервером через API, чтобы получать и отправлять данные без перезагрузки страницы.

**Задачи Frontend-разработчика**

Верстка: Frontend-разработчик отвечает за создание веб-страниц с помощью HTML и CSS, чтобы обеспечить правильное отображение контента на различных устройствах и в разных браузерах.

Интерактивность: Frontend-разработчик добавляет интерактивные элементы, такие как кнопки, формы, анимации и динамические элементы, используя JavaScript и соответствующие фреймворки.

Оптимизация производительности: Frontend-разработчик оптимизирует код и ресурсы для достижения быстрой загрузки и отзывчивости веб-приложений.

Frontend-разработчик играет ключевую роль в процессе создания веб-приложений и сайтов. Он обеспечивает привлекательный и удобный пользовательский интерфейс, который делает взаимодействие с приложением более комфортным и эффективным. Frontend-разработчик сотрудничает с дизайнерами, бэкенд-разработчиками и другими специалистами, чтобы обеспечить интеграцию всех компонентов в единое и гармоничное решение.

**Использование современных языков и фреймворков**

Frontend-разработчики активно используют современные языки программирования и фреймворки, которые облегчают и ускоряют разработку. Например, JavaScript с его последними версиями (ES6/ES7 и выше) предоставляет множество новых возможностей, таких как стрелочные функции, деструктуризация и async/await. Фреймворки, такие как React, Angular и Vue.js, позволяют создавать масштабируемые и переиспользуемые компоненты, упрощая процесс разработки. Это означает, что веб-приложения и сайты должны хорошо отображаться и работать на различных устройствах и экранах, включая компьютеры, планшеты и мобильные телефоны. Использование медиа-запросов и гибких макетов помогает создавать адаптивные интерфейсы.

Использование современных языков программирования и фреймворков является ключевым элементом современной Frontend-разработки. Они позволяют Frontend-разработчикам создавать более эффективные, масштабируемые и поддерживаемые веб-приложения и сайты. Ниже приведены некоторые из популярных современных языков и фреймворков, которые широко используются в современной Frontend-разработке:

* JavaScript является основным языком программирования для Frontend-разработки. С его помощью Frontend-разработчики могут создавать интерактивные элементы и обеспечивать динамическое взаимодействие с пользователем. Современные версии JavaScript (ES6/ES7 и выше) предоставляют множество новых возможностей, таких как стрелочные функции, деструктуризация, классы, модули и async/await.
* HTML (HyperText Markup Language) и CSS (Cascading Style Sheets) - это основы Frontend-разработки. HTML используется для определения структуры и содержимого веб-страницы, а CSS - для стилевого оформления и внешнего вида элементов. Современные спецификации HTML5 и CSS3 предоставляют богатые возможности для создания красивых и интерактивных пользовательских интерфейсов.
* React - это один из самых популярных JavaScript-фреймворков, разработанный компанией Facebook. Он позволяет создавать пользовательские компоненты, которые легко переиспользовать и поддерживать. React используется для создания масштабируемых и быстрых веб-приложений, обладает большой активной общественностью и множеством сторонних пакетов.
* Angular - это фреймворк от компании Google для разработки одностраничных веб-приложений (SPA). Он предоставляет набор инструментов и модулей для создания сложных и масштабируемых приложений. Angular использует шаблонный движок и предоставляет свою систему модулей и инъекцию зависимостей.
* Vue.js - это легкий и простой в использовании JavaScript-фреймворк для создания пользовательских интерфейсов. Он обладает маленьким размером, что делает его привлекательным для небольших проектов и начинающих разработчиков. Vue.js предоставляет удобные инструменты для создания интерактивных веб-приложений.
* TypeScript - это современный язык программирования, который является надмножеством JavaScript и предоставляет дополнительные возможности статической типизации и проверки ошибок на этапе разработки. Он позволяет создавать более безопасный и стабильный код, а также улучшает среду разработки и автодополнение.
* SASS (Syntactically Awesome Style Sheets) и Less - это препроцессоры CSS, которые позволяют использовать переменные, вложенные стили, миксины и другие расширенные функции для упрощения разработки и организации стилей.
* Next.js и Nuxt.js - это фреймворки для создания универсальных (universal) или серверно-рендеринговых (server-side rendered) веб-приложений. Они позволяют легко настроить SSR (серверный рендеринг) и предоставляют множество инструментов для оптимизации производительности и SEO.
* Webpack - это мощный инструмент для сборки и упаковки JavaScript-файлов и других ресурсов, таких как CSS, изображения и шрифты. Он позволяет оптимизировать размер и производительность веб-приложений.
* ESLint и Prettier - это инструменты для статического анализа и форматирования кода. Они помогают поддерживать стандарты кодирования, обнаруживать потенциальные ошибки и улучшать читаемость кода.

**Использование систем контроля версий**

Разработчики активно используют системы контроля версий, такие как Git, для отслеживания изменений в коде, совместной работы с другими разработчиками и управления версиями проекта. Это помогает предотвратить потерю данных, улучшить коллаборацию в команде и обеспечить безопасность проекта.

**Тестирование и безопасность кода**

Большое значение тестированию кода. Использование тестов позволяет обнаруживать ошибки и проблемы в коде на ранних этапах разработки, что позволяет избежать проблем в продакшн-среде. Основные виды тестов - это модульное тестирование (Unit тесты) и тестирование пользовательского интерфейса (UI тесты).

Не малое внимание уделяют безопасности своих приложений. Злоумышленники постоянно ищут уязвимости в коде, поэтому важно использовать безопасные практики разработки, такие как защита от инъекций, управление сессиями и защита от межсайтовых сценариев (XSS).

**Виды MVP-подходов. Что такое MVP?**

MVP расшифровывается как "Minimum Viable Product" или "Минимально жизнеспособный продукт". Это концепция, которая входит в разработку программного обеспечения, продуктов или услуг. MVP - это минимальный набор функций и возможностей, которые позволяют выпустить продукт на рынок, чтобы протестировать его гипотезы и получить обратную связь от пользователей с минимальными затратами времени и ресурсов.

Основные принципы MVP:

* Минимальность: MVP содержит только основные функции и возможности продукта. Это не полноценный продукт, а скорее его прототип с самым необходимым функционалом.
* Жизнеспособность: MVP должен быть способен работать и выполнять свою основную функцию. Он должен быть пригодным для использования и тестирования гипотез.
* Обратная связь: Основная цель MVP - получить обратную связь от пользователей. Это помогает выявить сильные и слабые стороны продукта и определить, какие изменения необходимы для его улучшения.
* Быстрый запуск: MVP должен разрабатываться и запускаться как можно быстрее. Главное - проверить гипотезы и начать сбор обратной связи.
* Итеративность: После запуска MVP можно вносить изменения, учитывая обратную связь, чтобы улучшить продукт. Затем можно выпустить новые версии с улучшенным функционалом.

Виды MVP-подходов:

* Прототипный MVP: В этом подходе создается быстрый прототип с минимальным функционалом. Прототип может быть интерактивным, чтобы пользователи могли взаимодействовать с ним и оценить его основные возможности.
* Приложение с ограниченным функционалом: В данном подходе разрабатывается полноценное приложение, но только с базовыми функциями. Это позволяет проверить, заинтересованы ли пользователи в идее продукта.
* Тестовый рынок: Этот подход связан с предварительным определением спроса на продукт. Можно использовать лендинг или рекламные материалы для привлечения клиентов, а затем анализировать их реакцию.
* "Демо" продукт: Создается продукт, который полностью демонстрирует возможности будущего продукта. Он может не содержать реального функционала, но призван показать его потенциал и привлечь инвесторов или партнеров.
* "Картинка" продукта: В этом случае используются визуальные средства, такие как макеты и дизайны, чтобы показать пользователю будущий продукт. Это может быть полезно для сбора обратной связи до полной разработки.

## ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1. "Приемы объектно-ориентированного проектирования. Паттерны проектирования" - Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес - Питер, 2002 - 366 страниц.
2. "Защищенный код" - Р. Мартин - Питер, 2011 - 464 страницы.
3. "Методология Scrum" - Кен Швабер, Джефф Сазерланд - Диалектика, 2011 - 192 страницы.
4. "Создание успешных MVP проектов" - Александр Антоненко, Андрей Соснович - Питер, 2018 - 272 страницы.
5. "Lean-стартап. Как осуществлять инновации быстро и дешево" - Эрик Рис - Манн, Иванов и Фербер, 2011 - 384 страницы.
6. "Искусство программирования" - Дональд Кнут - Вильямс, 2007 - Более 7000 страниц (суммарно в 4-х томах).