React - финал

NEXT JS
React Native
Astro JS
Electron JS
Паттерны проектирования
linux
Дальнейшее развитие

Что прошли за курс

HTML - язык разметки

CSS - красивая верстка

SCSS (SASS) - удобная верстка

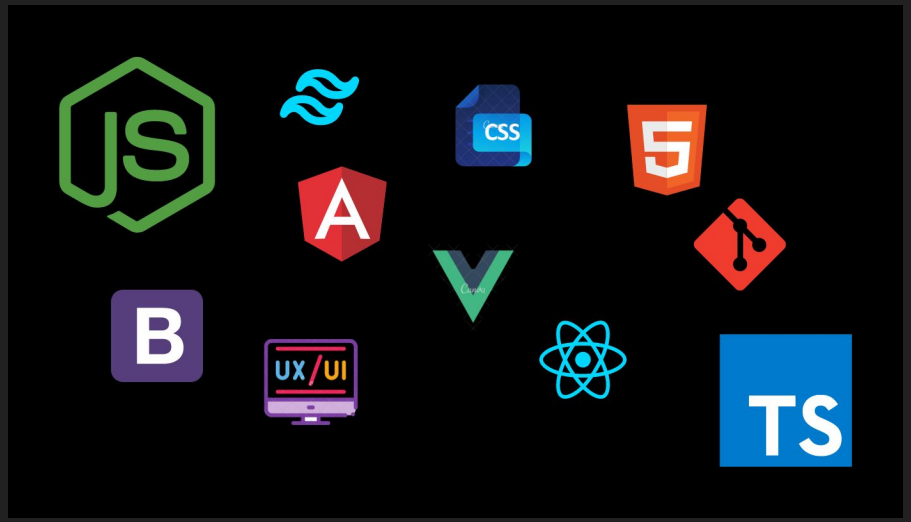
JavaScript

- Базовый (синтаксис языка)
- Браузерный (WebAPI, event loop)
- Продвинутый (ООП, асинхронность, паттерны)

GIT - работа с удаленными репозиториями (github)

React

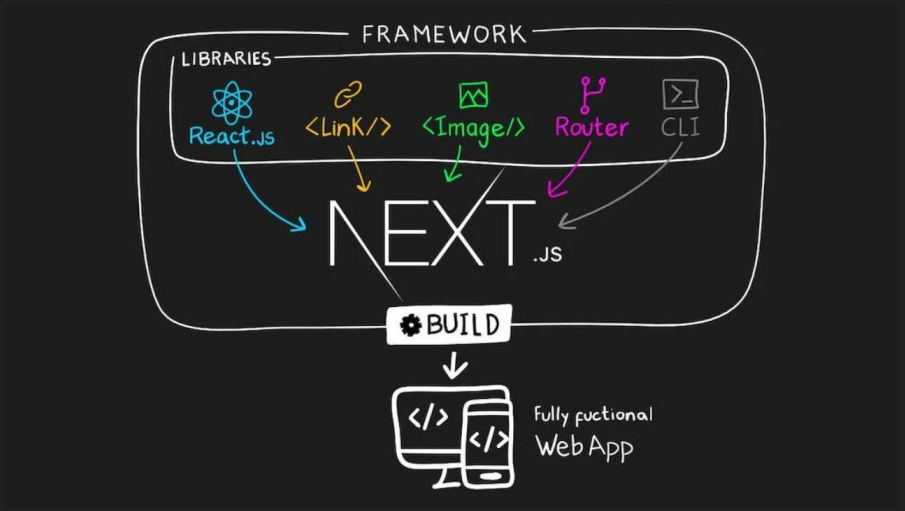
- Базовый (синтаксис фреймворка)
- Продвинутый (библиотеки поверх фреймворка)



HTML
CSS (SASS/SCSS)
JavaScript (JS) | TypeScript (TS)
React

Что такое Next js

Next.js — это популярный фреймворк для React, который позволяет создавать современные веб-приложения с серверным рендерингом (SSR) и статической генерацией страниц (SSG). Он добавляет к React ряд мощных возможностей и инструментов, упрощая разработку производительных и SEO-оптимизированных сайтов.



Установка
npx create-next-app

Next js SSR (Server Side Rendering)

В случае с SSR (Server-Side Rendering) в Next.js процесс рендеринга отличается от стандартных SPA (Single Page Application) приложений. В SSR HTML генерируется на сервере каждый раз при запросе страницы, а затем отправляется пользователю, что улучшает время отображения контента и позволяет индексировать страницу поисковыми системами.

SSR vs SPA:

В SPA (Single Page Application) весь рендеринг происходит на клиенте. Пользователь сначала получает минимальный HTML (обычно с одним корневым тегом), а затем вся страница рендерится на клиенте с помощью JavaScript.

В SSR же HTML рендерится на сервере и сразу передается пользователю, что улучшает первоначальное время отображения и SEO, так как поисковые боты видят уже готовую страницу.

Пример процесса SSR в Next.js:

Запрос: Пользователь открывает страницу /posts/1.

Серверный рендеринг: Next.js сервер получает запрос и вызывает getServerSideProps, чтобы получить данные для страницы, например, пост с id = 1.

HTML-ответ: Сервер генерирует HTML с постом и отправляет его в браузер.

Гидратация: Браузер получает этот HTML и начинает загружать JavaScript для того, чтобы страница стала интерактивной.

Интерактивность: Когда JS загружен, React берет на себя управление и делает страницу интерактивной (например, добавляет события на кнопки, формы и т.д.).

Преимущества и Недостатки SSR:

Преимущества SSR:

- **Ускоренный рендеринг первого экрана:** Пользователи быстрее видят контент, потому что HTML отправляется сразу.
- **SEO:** Поисковые системы могут легко индексировать страницы, так как они получают полный HTML.
- **Динамическое содержимое:** Сервер может передавать разные данные при каждом запросе, в зависимости от пользователя или других факторов.

Недостатки SSR:

- **Большая нагрузка на сервер:** Поскольку каждый запрос требует серверного рендеринга, это может увеличить нагрузку на сервер, особенно при большом числе пользователей.
- **Более сложная архитектура:** Нужно учитывать как серверные, так и клиентские части приложения, что требует более сложного управления состоянием и рендерингом.

Что такое React Native

React Native — это фреймворк с открытым исходным кодом, созданный Facebook, который позволяет разрабатывать мобильные приложения с использованием JavaScript и React. В отличие от React для веба, который работает в браузере, React Native предназначен для создания нативных мобильных приложений для iOS и Android, используя одну кодовую базу.

Установка

npx create-expo-app

Пошаговая инструкция по запуску React Native

Установка Expo CLI: Expo — это простой способ начать работу с React Native. Оно помогает быстро разворачивать и тестировать приложения.
npm install -g expo-cli

Создание нового React Native проекта - ***expo init MyReactNativeApp***

Запуск проекта - ***expo start***, после этого откроется веб-интерфейс Expo Developer Tools, и в терминале будет отображен QR-код.

Тестирование приложения:

На мобильном устройстве:

- Скачайте приложение Expo Go из App Store (для iOS) или Google Play (для Android).
- Откройте Expo Go и просканируйте QR-код, который отображается в терминале или веб-интерфейсе. Приложение запустится на вашем устройстве.

На эмуляторе:

- Для Android: Установите Android Studio и запустите эмулятор.
- Для iOS: Если у вас macOS, используйте Xcode и его iOS-симулятор.
- В веб-интерфейсе Expo Developer Tools выберите нужный эмулятор для запуска.

Построение приложения (опционально)

- ***expo build:android*** - Для Android
- ***expo build:ios*** - Для iOS (требуется macOS)

Плюсы и минусы React Native

Плюсы React Native:

- Кроссплатформенность: Одна кодовая база для iOS и Android.
- Нативный интерфейс: Приложения выглядят как нативные.
- Горячая перезагрузка: Быстрое внесение изменений без перезапуска приложения.
- Большое сообщество: Множество готовых библиотек и решений.
- JavaScript: Широко используемый язык программирования.
- Нативные модули: Возможность использовать нативный код, если нужно.

Минусы React Native:

- Сложности с производительностью: Не всегда достигает уровня нативных приложений.
- Ограниченные нативные модули: Иногда нужно писать нативный код для доступа к функциональности платформы.
- Различия платформ: Часто требуется учитывать специфические особенности iOS и Android.
- Меньше инструментов для отладки: Отладка сложнее, чем в нативной разработке.
- Размер приложения: Приложения могут быть больше, чем нативные аналоги.

Что такое Astro JS

Astro JS — это современный фреймворк для создания быстрых статических сайтов и гибридных приложений. Его основная цель — минимизировать количество JavaScript на стороне клиента, позволяя разработчикам отправлять только необходимый контент в браузер. Astro сочетает в себе лучшие практики для построения производительных веб-сайтов, таких как server-side rendering (SSR) и static site generation (SSG).

Установка
npm create astro

Особенности Astro JS

Island Architecture: Astro использует архитектуру «островов», где JavaScript отправляется в браузер только для интерактивных частей страницы (островов). Это значительно уменьшает объем клиентского кода и улучшает производительность.

Фреймворк-агностичность: В Astro можно использовать компоненты из разных фреймворков, таких как React, Vue, Svelte и Solid. Таким образом, можно интегрировать разные подходы к разработке в одном проекте.

Без клиентского JavaScript по умолчанию: По умолчанию Astro отправляет в браузер только HTML и CSS. Это позволяет страницам загружаться быстрее, так как не нужно обрабатывать лишний JavaScript.

SSR и SSG: Astro поддерживает как рендеринг на стороне сервера (SSR), так и генерацию статических сайтов (SSG), что делает его гибким для разных типов приложений.

Простота интеграции: Astro поддерживает плагины и можно легко добавить дополнительные функции, такие как обработка Markdown файлов, оптимизация изображений, стилизация и многое другое.

Островная архитектура

Островная архитектура (Island Architecture) — это подход к разработке веб-приложений, который фокусируется на оптимизации загрузки и производительности, минимизируя количество JavaScript, который загружается на стороне клиента. Этот подход особенно активно используется в фреймворках, таких как Astro JS.

Electron JS

Electron.js — это фреймворк для создания кросс-платформенных десктопных приложений с использованием веб-технологий, таких как HTML, CSS и JavaScript. Он позволяет разработчикам создавать приложения, которые работают на Windows, macOS и Linux, используя единый код.

Примеры приложений на Electron.js:

- Visual Studio Code
- Slack
- Discord

npm install electron

Основные особенности Electron.js

Кроссплатформенность: Приложения, созданные с помощью Electron, могут работать на различных операционных системах без необходимости изменения кода.

Использование веб-технологий: Electron позволяет использовать привычные веб-технологии, такие как React, Vue.js и Angular, для создания интерфейсов приложений.

Доступ к системным API: Electron предоставляет доступ к системным функциям, таким как файловая система, уведомления и работа с буфером обмена, что позволяет создавать полноценные десктопные приложения.

Простота разработки: Разработчики могут использовать один и тот же код как для клиентской, так и для серверной части, что упрощает процесс разработки.

Сообщество и поддержка: Electron имеет большое сообщество и множество доступных ресурсов, что облегчает обучение и решение проблем.

Что такое Паттерн

Паттерн проектирования — это часто встречающееся решение определённой проблемы при проектировании архитектуры программ.

В отличие от готовых функций или библиотек, паттерн нельзя просто взять и скопировать в программу. Паттерн представляет собой не какой-то конкретный код, а общую концепцию решения той или иной проблемы, которую нужно будет ещё подстроить под нужды вашей программы.

Паттерны часто путают с алгоритмами, ведь оба понятия описывают типовые решения каких-то известных проблем. Но если алгоритм — это чёткий набор действий, то паттерн — это высокоуровневое описание решения, реализация которого может отличаться в двух разных программах.

Если привести аналогии, то алгоритм — это кулинарный рецепт с чёткими шагами, а паттерн — инженерный чертёж, на котором нарисовано решение, но не конкретные шаги его реализации.

Зачем нужны паттерны

Проверенные решения. Вы тратите меньше времени, используя готовые решения, вместо повторного изобретения велосипеда. До некоторых решений вы смогли бы додуматься и сами, но многие могут быть для вас открытием.

Стандартизация кода. Вы делаете меньше просчетов при проектировании, используя типовые унифицированные решения, так как все скрытые проблемы в них уже давно найдены.

Общий программистский словарь. Вы произносите название паттерна, вместо того, чтобы час объяснять другим программистам, какой крутой дизайн вы придумали и какие классы для этого нужны.

Критика паттернов

Нужда в паттернах появляется тогда, когда люди выбирают для своего проекта язык программирования с недостаточным уровнем абстракции. В этом случае, паттерны — это костыль, который придает этому языку суперспособности.

Паттерны пытаются стандартизировать подходы, которые и так уже широко используются. Эта стандартизация кажется некоторым людям догмой и они реализуют паттерны «как в книжке», не приспосабливая паттерны к реалиям проекта.

Вникнув в паттерны, человек пытается применить свои знания везде. Даже там, где можно было бы обойтись кодом попроще.

Если у тебя в руках молоток, то все предметы вокруг начинают напоминать гвозди.

Классификация паттернов

Паттерны отличаются по уровню сложности, детализации и охвата проектируемой системы. Проводя аналогию со строительством, вы можете повысить безопасность перекрёстка, поставив светофор, а можете заменить перекрёсток целой автомобильной развязкой с подземными переходами.

Самые низкоуровневые и простые паттерны — идиомы. Они не универсальны, поскольку применимы только в рамках одного языка программирования.

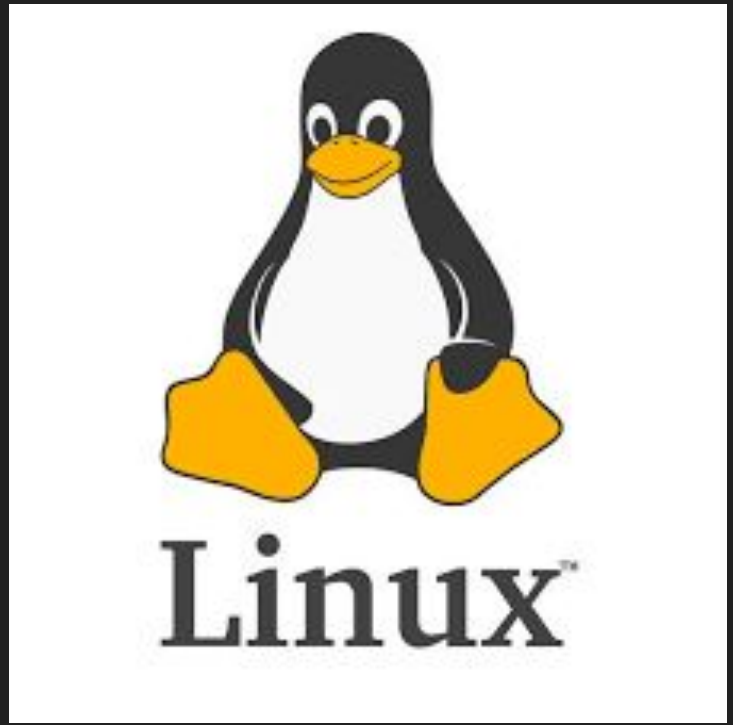
Самые универсальные — архитектурные паттерны, которые можно реализовать практически на любом языке. Они нужны для проектирования всей программы, а не отдельных её элементов.

Кроме того, паттерны отличаются и предназначением. В этой книге будут рассмотрены три основные группы паттернов:

- **Порождающие паттерны** беспокоятся о гибком создании объектов без внесения в программу лишних зависимостей.
- **Структурные паттерны** показывают различные способы построения связей между объектами.
- **Поведенческие паттерны** заботятся об эффективной коммуникации между объектами.

Что такое Linux

Linux — это семейство операционных систем на основе ядра Linux, разработанного Линусом Торвальдсом в 1991 году. Linux является свободным и открытым программным обеспечением, что означает, что любой желающий может скачать, использовать, изменять и распространять его. В отличие от коммерческих операционных систем, таких как Windows или macOS, Linux предлагает большую гибкость и контроль над системой.



Зачем нужен Linux

Терминал Linux (или командная строка) — это текстовый интерфейс, который позволяет пользователю взаимодействовать с операционной системой, вводя команды. Это мощный инструмент, который предоставляет множество возможностей для управления системой, автоматизации задач и работы с программным обеспечением.

Знание терминала Linux предоставляет фронт-энд разработчикам мощный инструмент для повышения эффективности, упрощения рабочих процессов и расширения возможностей в разработке и деплое приложений. Умение работать с командной строкой является важным навыком, который может существенно улучшить опыт работы и повысить конкурентоспособность на рынке труда.

Профессии в Web разработке

Верстальщик (умирающая профессия)

Тестировщик / QA инженер (мануальный / авто-тестировщик)

Front-end разработчик (React / Vue / Angular / next.js / nuxt.js / astro.js)

Back-end разработчик (node.js + express js / nest.js)

Mobile (мобильный) разработчик (React-native)

Desktop разработчик (electron js)

Проектный / Продуктовый менеджер (чистые soft скилы)

Dev-ops (настройка серверной части, взаимодействие с серверами, CI-CD, docker и тд)

Data аналитик (аналитика и планирование, оцифровка бизнеса, работа с данными)

Развитие в IT сфере

Составляющие описания квалификации разработчика (работника)

- Опыт (какие результаты вы уже достигли)
- Hard скилы (какие у вас есть знания)
- Soft скилы (как вы умеете коммуницировать в социуме)

Категории роста как специалиста:

- Горизонтальный рост
- Вертикальный рост

Горизонтальный рост

Горизонтальный рост - увеличение знаний, то есть повышение своей квалификации как разработчика в своей сфере. **Улучшение HARD скилов (важный знания и опыт)**

Углубленное изучение языка, изучение фреймворков, изучение библиотек, изучение паттернов проектирования, новые знаний в целом.

Более того, всегда можно открыть новый “горизонт”, начать изучать другой язык, другое направление (front-end / back-end / mobile)

- **Junior (начинающий без опыта)** - есть знания, но нет опыта;
- **Strong Junior (начинающий с опытом)** - есть и знания и опыт;
- **Middle (разработчик)** - умеет решать задачи самостоятельно (опытный);
- **Middle+ (сильный разработчик)** - эксперт в своей области без внушительного опыта;
- **Senior (сильный и опытный разработчик)** - эксперт в своей области с большим опытом.

Вертикальный рост

Вертикальный рост - должностной рост, то есть повышение своей зоны ответственности. **Улучшение SOFT скилов (важна ответственность и социализация)**

Рост в компании, умение управлять людьми, умение брать на себя ответственность, более глобальное видение развития проекта / продукта.

- **Intern (стажер)** - нет ни знаний, ни опыта, требует обучение и проверку работы;
- **Разработчик** - основная единица в IT сфере, выполняет поставленные задачи самостоятельно;
- **Team-leader (Лидер команды по специальности)** - умеет управлять людьми и эксперт в своей области;
- **Tech-leader (Лидер команды со стороны кода)** - эксперт в программировании;
- **CTO / Chief Technical Officer (Технический директор)** - эксперт в программировании, управлении командой и умеет глобально планировать, имеет колоссальный опыт.

Ресурсы

NEXT JS - ТЫК

React Native - ТЫК

RoadMap по изучению React Native - ТЫК

Astro JS - ТЫК

Electron JS - ТЫК

паттерны проектирования - ТЫК

RoadMap по изучению linux - ТЫК