Лекция 1

Основы React

React — это одна из самых популярных библиотек для разработки интерфейсов. Она используется такими компаниями, как Facebook, Netflix, Airbnb. На курсе мы шаг за шагом изучим основы, перейдём к продвинутым темам и создадим полноценные проекты.

Экосистема, установка окружения, JSX

Лектор: Жумагазиева Багдат

Экосистема веб-разработки

Frontend - интерфейс

- Всё, что видит и использует пользователь: кнопки, формы, страницы
- Технологии: HTML, CSS, JavaScript
- Фреймворки: React, Vue, Angular

Backend - логика и данные

- Обработка запросов и выполнение бизнес-правил
- Работа с базами данных и API
- Технологии: Node.js, Python (Django/FastAPI), Java (Spring), PostgreSQL, MongoDB

React и SPA

- **React —** библиотека для динамичных интерфейсов
- Сайты превращаются в **SPA (Single Page Application)**:
    - Обновление данных без перезагрузки страницы
    - Быстрый отклик и плавный UX
    - Похоже на работу мобильного приложения

Роль React

Что такое React

- **Библиотека**, а не фреймворк
- Для работы с **UI** (пользовательским интерфейсом)

Философия React

- Интерфейс = набор **компонентов**
- Компоненты легко переиспользовать

Технология

- Использует **Virtual DOM**
- Быстрее классического обновления страниц

Установка окружения

Node.js

- Исполняет **JavaScript вне браузера**
- Основа для запуска React-приложений

Менеджер пакетов

- **npm, pnpm или yarn**
- Установка библиотек и зависимостей

Редактор кода

- **VS Code**
- Поддержка плагинов, автодополнения
- Интеграция с **Git**

Инструменты сборки

Раньше

- **Create React App (CRA)**
  Удобный, но стал **медленным**

Сегодня

- **Vite** — лёгкий и быстрый сборщик
  Запуск проекта за **секунды**
- Поддержка **TypeScript** «из коробки»

Создание проекта с Vite

Шаги установки

1. Установить **Node.js** (если не установлен)
2. Выполнить в терминале: npm create vite@latest my-app
3. Выбрать **React** или **React + TypeScript**
4. **Зайти в папку проекта и установить зависимости:**
   - cd my-app
   - npm install
5. Запустить проект: npm run dev

ESLint и Prettier

ESLint

- Проверяет код на ошибки
- Следит за стилем написания
- Подсказывает лучшие практики

Prettier

- **Автоматически форматирует** код
- Делает стиль кода единообразным

Зачем это нужно

- Код у всех выглядит одинаково
- Легче читать проекты друг друга
- Меньше ошибок и споров о «стиле»

ESLint и Prettier Установка

Установка
1. npm install eslint prettier --save-dev
2. npx eslint --init

Дальше он задаст вопросы:

- Какой проект? → *JavaScript modules (import/export)*
- Где работает код? → *Browser*
- Используете React? → *Yes*
  Используете TypeScript? → *Yes/No* (по ситуации)
- Формат файла конфигурации → *JSON* или *JS*

3. Создай файл **.prettierrc** в корне проекта, например:

```
{
  "singleQuote": true,
  "semi": true,
  "trailingComma": "es5"
}
```

4. Интеграция с VS Code

- Установи расширения:
  - **ESLint**
  - **Prettier - Code Formatter**

Git и контроль версий

Зачем нужен Git

- Сохраняет **историю проекта**
- Позволяет работать **в команде**
- Легко откатиться к старой версии

Где хранят проекты

- **GitHub**
- **GitLab**

Основные команды

```
git init              # инициализация проекта
git add .             # добавить изменения
git commit -m "msg"   # сохранить версию
git push              # загрузить на сервер
```

Что такое JSX

JSX — это способ писать **HTML-подобный код** внутри JavaScript.

HTML + JS = JSX

```
function Welcome() {
  return (
    <div>
      <h1>Hello, React!</h1>
      <p>Это мой первый компонент</p>
    </div>
  );
}
```

Особенности JSX

1. Атрибуты пишутся как в HTML, но есть отличия:

 // вместо class

2. Внутри JSX можно вставлять **JavaScript-выражения**: <h1>{2 + 2}</h1> // 4
3. Компоненты пишутся с заглавной буквы: <App />

Другие особенности:

- Все теги должны быть **закрыты** (`<img />`, `<br />`).
- В JSX может быть только **один корневой элемент**.
- Можно использовать **условия и циклы** внутри `{}`.

Первый компонент

```
function Welcome(props) {
  return <h1>Привет, {props.name}</h1>;
}

<Welcome name="React" />
```

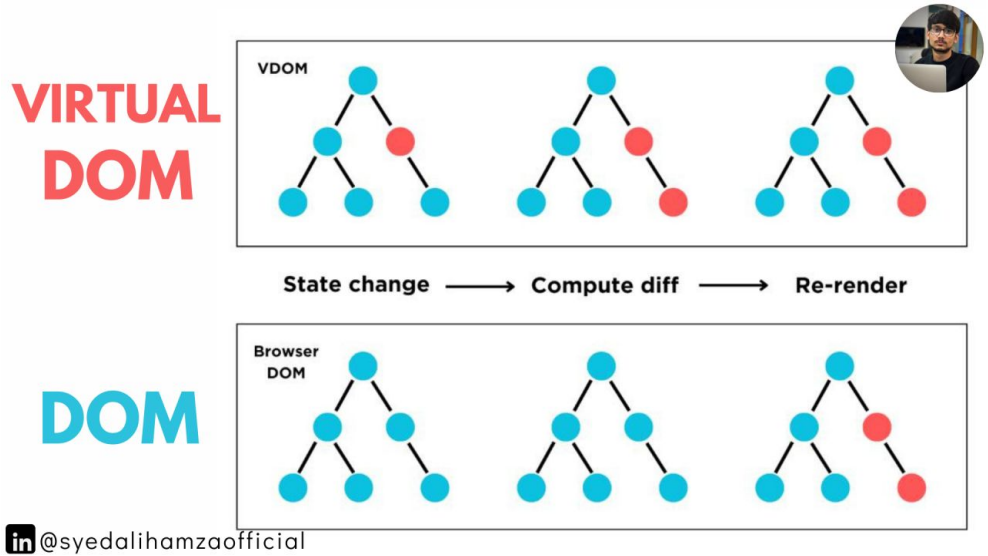
Virtual DOM

Virtual DOM — это лёгкая копия DOM в памяти.

State change (изменение состояния)
– мы меняем что-то в приложении (например, текст кнопки).

Compute diff (сравнение версий)
– React сравнивает старый Virtual DOM с новым.
– Красным отмечено то, что изменилось.

Re-render (перерисовка)
– в настоящий DOM попадают только изменённые элементы, а не вся страница.



SPA vs MPA

MPA — Multi Page Application (многостраничное приложение)

- Это «классический сайт».
- Каждая страница загружается отдельно.
- При клике на ссылку браузер делает полный переход → страница мигает и перезагружается.
- Пример: интернет-магазин 2000-х, новостные сайты.

SPA — Single Page Application (одностраничное приложение)

- Загружается **одна страница**.
- При переходе React сам обновляет только нужную часть экрана.
- Страница не перезагружается полностью, всё выглядит плавно и быстро.
- Пример: Gmail, YouTube, Facebook.

Инструменты разработчика

React Developer Tools

- Расширение для браузера (Chrome/Firefox).
- Показывает дерево компонентов и их состояние.

Hot Reloading

- Код обновляется сразу, без перезагрузки страницы.
- Удобно при разработке: изменили текст → сразу видно результат.

DevTools браузера

- Вкладки Elements, Console, Network.
- Помогают искать ошибки и смотреть, как работает приложение.