

REACT, VITE



План лекции

Основные моменты:





1. React
2. Vite
3. Создание нового проекта
4. Функциональные компоненты
5. Props
6. Структура проекта
7. Virtual dom





Что такое React?



- **React — JavaScript-библиотека для создания пользовательских интерфейсов (UI).**
 - **Разработан Facebook, используется для построения компонентов.**
 - **Основная идея — компонентный подход: UI разбивается на независимые, переиспользуемые части.**
- 
- 

Что делает React особенным?



- Virtual DOM — производительное обновление интерфейса.
- Декларативный подход — описываешь что должно быть, а не как.
- Упрощает работу с состоянием и взаимодействием между частями UI.



Vite

Vite — это современный инструмент сборки (build tool) и разработки фронтенд-приложений, который заменяет более старые решения вроде Webpack.



Что делает Vite в React-проекте?

1. Dev-сервер (для разработки):
 - Запускает приложение локально.
 - Использует ES-модули и native браузерную загрузку.
 - Позволяет редактировать код и мгновенно видеть результат (hot module replacement, HMR).



Что делает Vite в React-проекте?

2. Сборка (build):

- Использует esbuild и Rollup для быстрой сборки в production.
- Оптимизирует код, разбивает его на чанки, минифицирует и т.д.



Создание нового проекта

npm create vite@latest

- ◇ Project name:
| project-01
- ◇ Select a framework:
| React
- ◇ Select a variant:
| TypeScript



Что делает Vite в React-проекте?

npm i - для установки зависимостей

npm run dev - для запуска проекта

Ctrl + c - для остановки проекта



Что делает Vite в React-проекте?

npm i - для установки зависимостей

npm run dev - для запуска проекта

Ctrl + c - для остановки проекта





JSX



JSX — синтаксис, похожий на HTML в JavaScript-файлах.

```
const element = <h1>Hello, world!</h1>;
```

- Под капотом — это вызов `React.createElement()`.
- Можно вставлять выражения через `{}`:

```
const name = "Anna";  
const greeting = <p>Hello, {name}!</p>;
```





Функциональные компоненты ✕

- Компонент — это функция, возвращающая JSX.

```
function Welcome() {  
  return <h1>Welcome to React!</h1>;  
}
```

- Именуются с большой буквы (Welcome, а не welcome).



Рендеринг компонента



React-элемент вставляется в DOM через
ReactDOM.createRoot():

```
import React from 'react';  
import ReactDOM from 'react-dom/client';  
import App from './App';
```

```
const root =
```

- ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />);



Props (свойства компонентов)

Объявление компонента с пропсами:

// название компонента Greeting.tsx

```
function Greeting({ name }) {  
  return <p>Hello, {name}!</p>;  
}
```

Использование:

```
<Greeting name="Alice" />
```

```
<Greeting name="Bob" />
```



Props (типизация)

```
type GreetingProps = {  
  name: string;  
};
```

```
function Greeting({ name }: GreetingProps) {  
  return <p>Hello, {name}!</p>;  
}
```



Структура проекта

my-app/	
├── public/	# Публичные файлы (favicon, robots.txt и т.п.)
│ └── vite.svg	# Пример SVG-иконки
├── src/	# Исходный код приложения
│ ├── assets/	# Изображения, иконки, шрифты
│ ├── components/	# Переиспользуемые компоненты
│ │ └── Button.tsx	
│ ├── pages/	# Страницы (если используется маршрутизация)
│ │ └── Home.tsx	
│ ├── layouts/	# Общие шаблоны (например, с хедером/футером)
│ ├── hooks/	# Кастомные хуки React
│ ├── store/	# Pinia/Zustand или другие состояния
│ ├── types/	# Глобальные типы и интерфейсы TypeScript
│ ├── utils/	# Утилиты и вспомогательные функции
│ ├── App.tsx	# Корневой компонент
│ ├── main.tsx	# Точка входа (рендеринг React)
│ └── index.css	# Глобальные стили (может быть Tailwind)
├── .eslintrc.cjs	# Настройка ESLint
├── .prettierrc	# Настройка Prettier
├── tsconfig.json	# Конфигурация TypeScript
├── vite.config.ts	# Конфигурация Vite
├── package.json	# Зависимости проекта
└── README.md	



Что такое Virtual DOM?



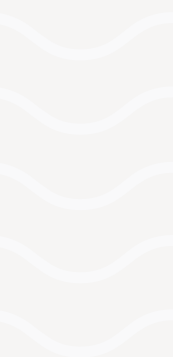
Virtual DOM — это легковесная копия реального DOM, которую React создает и использует для сравнения изменений, не трогая реальный DOM напрямую сразу.



Как это работает



- React рендерит Virtual DOM из компонентов (например, `<App />` → структура DOM-элементов в памяти).
- Когда состояние или пропсы меняются — React создает новую копию Virtual DOM.
- React сравнивает старый и новый Virtual DOM (это называется "diffing").
- Он находит минимальные различия (что изменилось).
- Реальный DOM обновляется только там, где есть изменения.



Если вы нашли опечатку или ошибку в презентации сообщите
Alisher Khamidov

