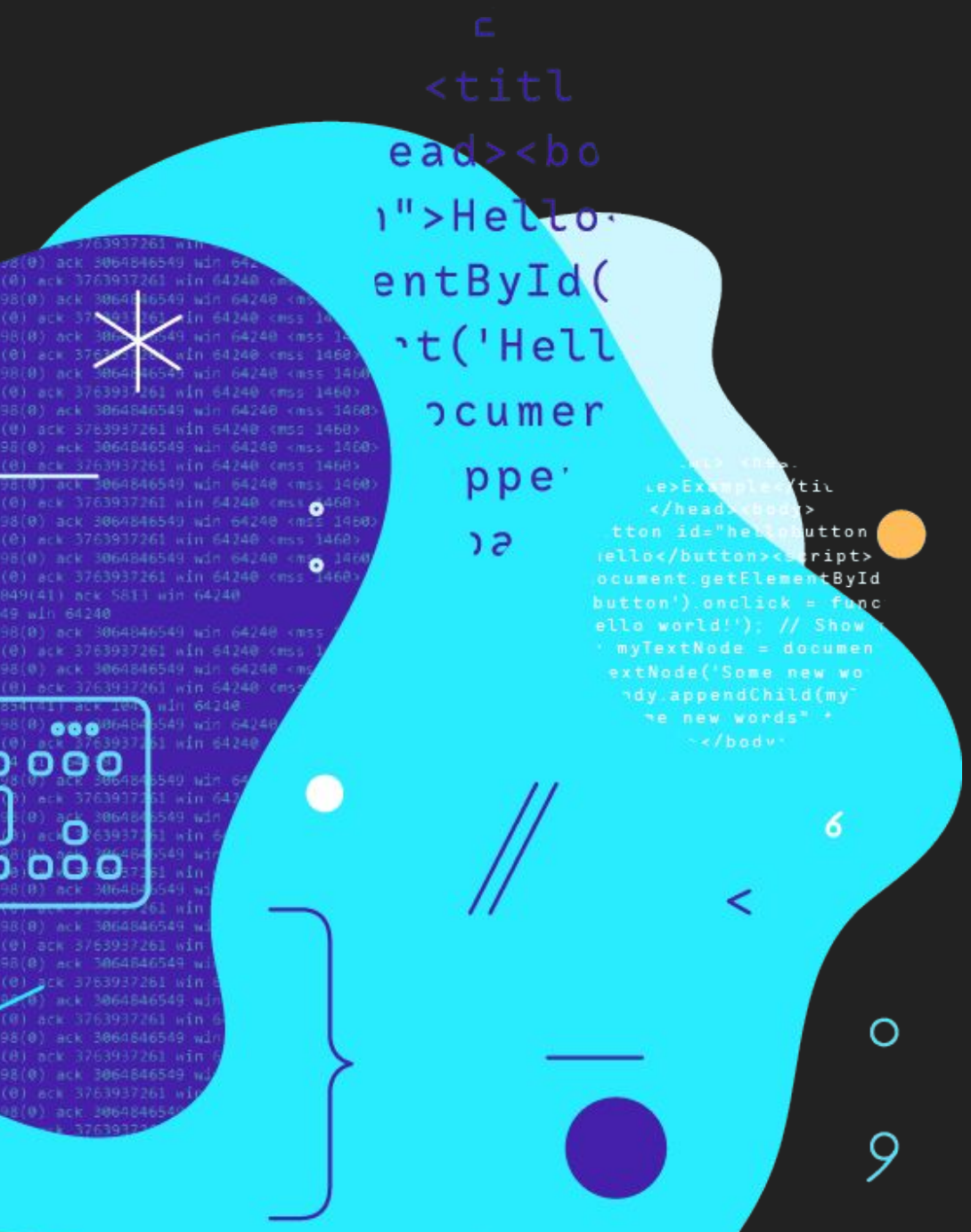


PHASE 3
WEEK 1

DAY 3

ELBRUS/BOOTCAMP

План

1. React Router, React Router hooks
2. CSS modules, Sass/SCSS
3. Class component & Lifecycle Methods

React Router

React Router

React Router — библиотека для работы с маршрутами в React.

React Router DOM — обёртка для React Router в веб-приложениях.

```
npm install react-router-dom
```

https://reactrouter.com/docs/en/v6/getting-started/overview

Основные компоненты

Основные компоненты React Router:

- `BrowserRouter` — маршрутизатор для веба
- `Routes` — сопоставляет маршруты
- `Route` — маршрут с определённым путём
- `Link`, `NavLink` — ссылки

https://reactrouter.com/docs/en/v6

Пример

```tsx
// App.tsx
<BrowserRouter>
  <Navigation />

  <Routes>
    <Route path="/" element={
      <div>Главная страница</div>
    }>
  </Routes>
</BrowserRouter>
```

```tsx
// Navigation.tsx
<nav>
  <ul>
    <li>
      <Link to="/">Главная</Link>
    </li>
    <li>
      <Link to="/about">О сайте</Link>
    </li>
  </ul>
</nav>
```

Динамический маршрут

`Route` может быть параметризованный (динамический):

```
<Route path="/users/:id" element={<UserCard />} />
```

React Router hooks

useParams

Хук из React Router DOM.

Возвращает объект с параметрами маршрута.

https://reactrouter.com/en/6.4.4/hooks/use-params

```
// App.tsx
<Route path="/users/:id" element={
  <UserCard />
} />

// UserCard.tsx
const { id } = useParams();

return <h1>{id}</h1>;
```

useNavigate

Хук из React Router DOM.

Возвращает функцию, которая

позволяет осуществлять программные

перемещения (история, прямые ссылки)

https://reactrouter.com/en/6.4.4/hooks/use-navigate

```
// UserCard.tsx
const navigate = useNavigate();

// Может принимать в качестве
аргумента либо путь
navigate('/users')

// либо число, как положительное,
так и отрицательное
navigate(-1)
```

CSS module, Sass/SCSS

CSS

Импорт обычного CSS-файла:

```
import './App.css';
```

Использование класса из обычного CSS-файла:

```
<div className="card"></div>
```

CSS modules

Импорт CSS-модуля:

```
import styles from './Style.module.css'
```

Использование класса из CSS-модуля:

```
<div className={styles.card}></div>
```

Результат после компиляции:

```
<div class="App_card__3tefE"></div>
```

CSS modules: настройка проекта

Установи плагин typescript-plugin-css-modules:
`npm install -D typescript-plugin-css-modules`
в корневую папку твоего проекта!

В секцию compilerOptions файла tsconfig.json добавь:

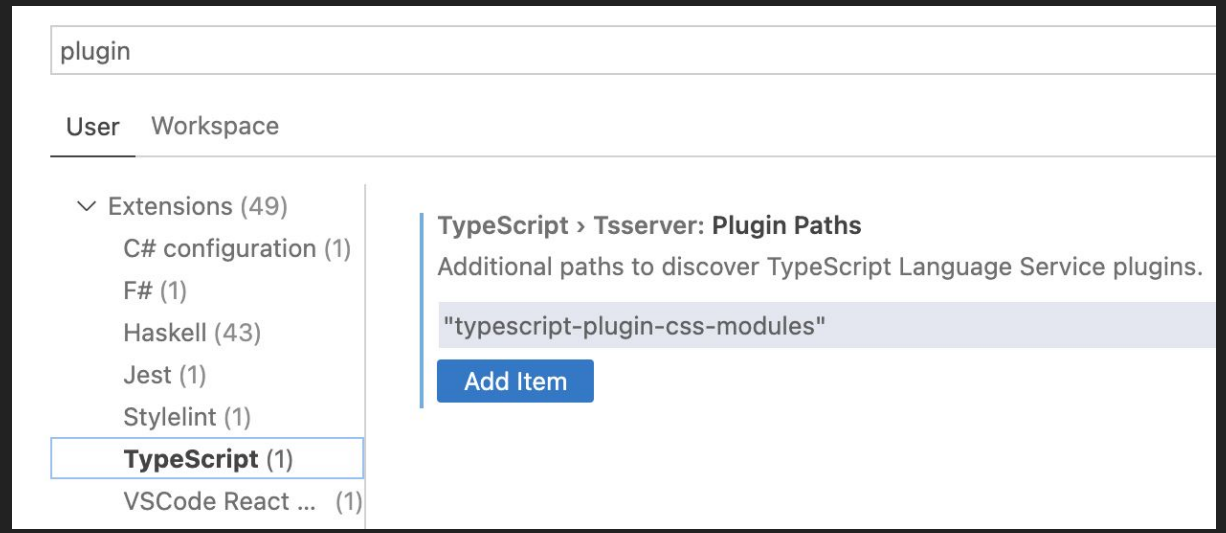
```
{
 "compilerOptions": {
    ...
    "plugins": [{
      "name": "typescript-plugin-css-modules"
    }]
 },
 ...
}
```

Также добавь этот плагин в настройки VS Code, секция TypeScript > Tsserver > Plugin Paths

Создай файл src/global.d.ts с таким содержанием:

```
declare module '*.module.css' {
 const classes: { [key: string]: string };
 export default classes;
}

declare module '*.module.scss' {
 const classes: { [key: string]: string };
 export default classes;
}
```



Sass/SCSS

Sass — препроцессор языка CSS.

Поддерживает два синтаксиса — Sass и SCSS.

```
npm install -D sass
```

Sass/SCSS

```
// CSS-синтаксис
.card {
 border: 1px solid;
}

.card .title {
 font-size: 1.2em;
}

.card .image {
 display: flex;
}
```

```
// SCSS-синтаксис
.card {
  border: 1px solid;

  .title {
    font-size: 1.2em;
  }

  .image {
    display: flex;
  }
}
```

```
// Sass-синтаксис
.card
  border: 1px solid

  .title
    font-size: 1.2em

  .image
    display: flex
```

Sass/SCSS

Файлы Sass/SCSS импортируются так же, как и CSS:

- через модули

```
import styles from './App.module.scss'
```

- …или обычным файлом

```
import './App.scss'
```

Class Component

Class Component: main methods

```
class ClassComponent extends Component {
  constructor(props) {
    super(props)
    this.state = { count: 0 } // так формируется state
  }

  componentDidMount() { } // при монтировании компонента
  componentDidUpdate(prevProps, prevState) { } // при обновлении
компонента
  componentWillUnmount() { } // при размонтировании компонента

  // метод формирующий React-элемент
  render() { return <section>Hello, React!</section> }
}
```

Class Component: custom methods

```
class ClassComponent extends Component {
  constructor(props) {
    super(props)
    this.state = { count: 0 } // так формируется state
  }

  // необходимо применять bind при запуске с контекстом this
  incrementCount() { this.setState({ count: this.state.count + 1 }) }

  render() {
    return (
      <button onClick={this.incrementCount.bind(this)}>
         {this.state.count}
      </button>
   )
  }
}
```

Lifecycle Methods

Lifecycle Methods

Mounting

Update

Unmounting

Интерактивная диаграмма:
https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/