Работа с JSX

В этой лабораторной мы познакомимся с синтаксисом расширения языка JavaScript - JSX. JSX создаёт «элементы» React и позволяет заранее видеть код для UI.

1. Откройте страницу проекта Babel.js
2. В левое поле введите фрагмент JSX

```
<div className="some">2 + 3 = 5</div>
```

вы получите что-то похожее на

```
"use strict";

/*#__PURE__*/
React.createElement("div", {
  className: "some"
}, "2 + 3 = 5");
```

Примечание: в JSX мы явно используем `className`, потому что в DOM-объектах именно этой свойство отвечает за строку с названиями классов в узле

3. Познакомьтесь с подсказками и убедитесь, что все понятно

- `JSX` специальный синтаксис позволяющий из HTML-подобного кода получать React-элементы
- `.createElement(element,attrs,body)` - встроенный метод React для создания элемента на странице
  - `element` - название элемента, может совпадать с названием HTML-тегов или быть другим в случае создания компонентов
  - `attrs` - объект, ключи (свойства) которого становятся атрибутами при построении DOM-дерева
  - `body` - содержимое в виде текста или другого JSX, которое помещается внутрь DOM при отрисовке

4. Встройте в JSX-код выражения

```
<div className="some">2 + 3 = {2 + 3}</div>
```

Обратите внимание: в фигурные скобки JSX можно помещать любые выражения, которые возвращают числа, строки, другие типы данных. Можно даже поместить другой JSX.

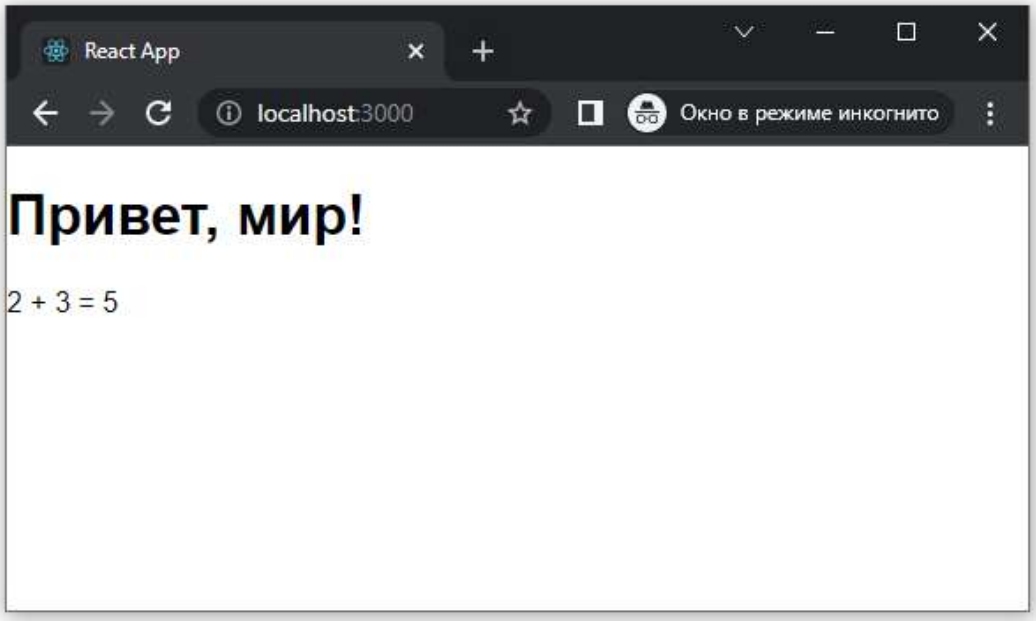
5. Встройте полученный JSX в файл `App.js`

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

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <h1>Привет, мир!</h1>
        <div className="some">2 + 3 = {2 + 3}</div>
      </header>
    </div>
  );
}

export default App;
```

6. Обновите страницу в браузере, должны увидеть строку `2 + 3 = 5`



7. Вынесите JSX-выражение внутри файла `App.js` из компонента `App` в константу `exp1` и вставьте фигурные скобки `{}` с этой константой внутрь JSX-кода компонента `App`

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

const exp1 = <div className="some">2 + 3 = {2 + 3}</div>;

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <h1>Привет, мир!</h1>
        {exp1}
      </header>
    </div>
  );
}

export default App;
```

Результат в браузере не изменится, но наш код стал удобней.

8. Определите новые JSX-выражения в произвольном месте файла `App.js`

```
const date = new Date();
const odd = <div>нечётный</div>
const even = <div>чётный</div>
const result = date.getHours() % 2 ? odd : even
```

9. Встройте эти JSX-выражения в `return` компонента `App`

```
...
  return (
    <div className="App">
      <header className="App-header">
        <h1>Привет, мир!</h1>
        {exp1}
        {date.toLocaleTimeString()}
        {result}
      </header>
    </div>
  );
...
```

10. Убедитесь, что в браузере появятся данные типа таких

```
2 + 3 = 5
15:20:22
нечётный
```

11. Создайте React-элемент на основе JSX описывающего изображения и строки с адресом изображения. Полный код будет выглядеть так

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


const exp1 = <div className="some">2 + 3 = {2 + 3}</div>;

const date = new Date();
const odd = <div>нечётный</div>
const even = <div>чётный</div>
const result = date.getHours() % 2 ? odd : even

const imageURL = `https://placekitten.com/100/100`;
const image = <img  src={imageURL}   />;

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <h1>Привет, мир!</h1>
        {image}
        {exp1}
        {date.toLocaleTimeString()}
        {result}
      </header>
    </div>
  );
}

export default App;
```

12. Познакомьтесь с утверждениями и убедитесь, что все их них понятны

- `JSX` - специальный синтаксис расширяющий JS и позволяющий создавать React-элементы
- JSX может содержать любые выражения в `{}`
- JSX-выражения (*например* `<span></span>`) могут помещаться в переменные, константы и в JSX (*например,* `<div>{<span></span>}</div>`)
- JSX можно использовать с тернарным оператором (как самостоятельно, так и внутри другого JSX)
- "атрибуты" в JSX-элементах обязательно должны быть с кавычками, либо с фигурными скобками. Например,
  - `<div className="some"></div>`
  - `<div className={100}></div>`

13. Выводы

- узнали что такое JSX и научились его использовать
- попробовали сервис Babel.js Try it out
- создали React-элементы с JSX-выражениями
- узнали, что JSX умеют работать с тернарными операторами
- обратили внимание на название свойства `className`
- запомнили, что "атрибуты" (то есть свойства JS-объектов) записываются с кавычками, либо с фигурными скобками `{}`