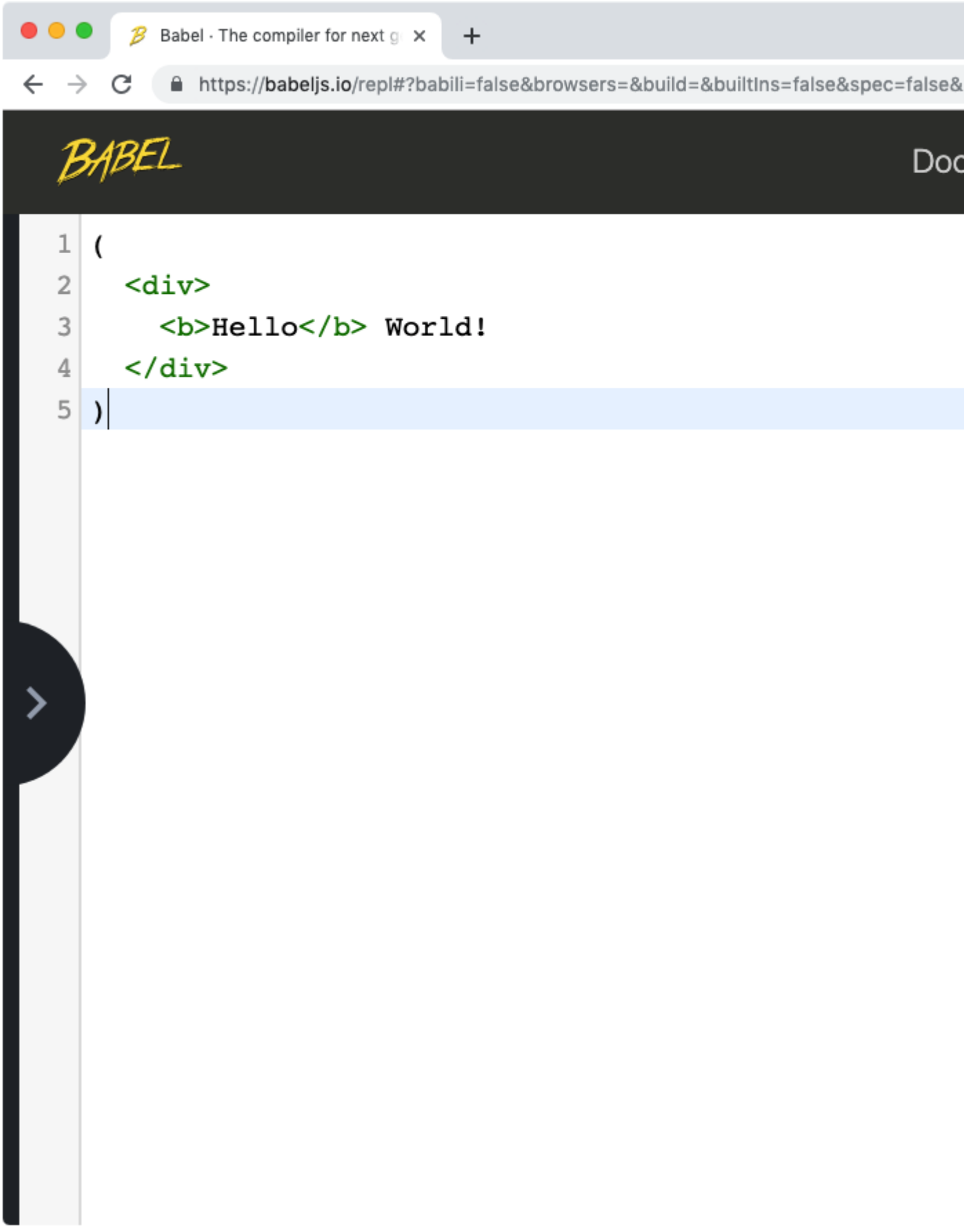
4. JSX 의 기본 규칙 알아보기

JSX 는 리액트에서 생김새를 정의할 때, 사용하는 문법입니다. 얼핏보면 HTML 같이 생겼지만 실제로는 JavaScript 입니다.

```
return <div>안녕하세요</div>;
```

리액트 컴포넌트 파일에서 XML 형태로 코드를 작성하면 babel 이 JSX 를 JavaScript 로 변환을 해줍니다.

어떻게 변환되는지 한번 예시를 볼까요?


Babel · The compiler for next g
https://babeljs.io/repl#?babili=false&browsers=&build=&builtIns=false&spec=false&
BABEL
Doc
1 (
2
3 Hello World!
4
5)

https://bit.ly/2wMpkk2

Babel 은 자바스크립트의 문법을 확장해주는 도구입니다. 아직 지원되지 않는 최신 문법이나, 편의상 사용하거나 실험적인 자바스크립트 문법들을 정식 자바스크립트 형태로 변환해줌으로서 구형 브라우저같은 환경에서도 제대로 실행 할 수 있게 해주는 역할을 합니다.

JSX 가 JavaScript 로 제대로 변환이 되려면 지켜주어야 하는 몇가지 규칙이 있습니다. 다음 문법들을 준수해주시면 앞으로 리액트 컴포넌트를 개발함에 있어서 큰 어려움이 없을 것입니다!

꼭 닫혀야 하는 태그

태그는 꼭 닫혀있어야 합니다.

다음과 같은 코드는 오류가 발생하게 됩니다.

App.js

```
import React from 'react';
import Hello from './Hello';

function App() {
  return (
    <div>
      <Hello />
      <Hello />
      <Hello />
      <div>
    </div>
  );
}

export default App;
```

Failed to compile

```
./src/App.js
  Line 9:  Parsing error: Unterminated JSX contents

   7 |         <Hello />
   8 |         <div>
>  9 |       </div>
     |             ^
  10 |   );
  11 | }
  12 |
```

태그를 열었으면 꼭, `<div></div>` 이렇게 닫아주어야 합니다.

HTML 에서는 `input` 또는 `br` 태그를 사용 할 때 닫지 않고 사용하기도 합니다. 하지만 리액트에서는 그렇게 하면 안됩니다.

App.js

```
import React from 'react';
import Hello from './Hello';

function App() {
  return (
    <div>
      <Hello />
      <Hello />
      <Hello />
      <input>
      <br>
    </div>
  );
}

export default App;
```

태그와 태그 사이에 내용이 들어가지 않을 때에는, Self Closing 태그 라는 것을 사용해야 합니다. 현재 Hello 컴포넌트를 사용 할 때에도 Self Closing 태그를 사용해주었는데요, 열리고, 바로 닫히는 태그를 의미합니다.

App.js

```
import React from 'react';
import Hello from './Hello';

function App() {
  return (
    <div>
      <Hello />
      <Hello />
      <Hello />
      <input />
      <br />
    </div>
  );
}

export default App;
```

꼭 감싸져야하는 태그

두가 이상의 태그는 무조건 하나의 태그로 감싸져있어야 합니다. 한번 다음 코드를 작성해보세요.

App.js

```
import React from 'react';
import Hello from './Hello';
```

```jsx
function App() {
  return (
    <Hello />
    <div>안녕히계세요.</div>
  );
}

export default App;
```

이런 코드는 오류가 발생하게 됩니다.

그 대신에 하나의 태그로 감싸주셔야 합니다.

```jsx
import React from 'react';
import Hello from './Hello';

function App() {
  return (
    <div>
      <Hello />
      <div>안녕히계세요</div>
    </div>
  );
```

```
}

export default App;
```

하지만, 이렇게 단순히 감싸기 위하여 불필요한 div 로 감싸는게 별로 좋지 않은 상황도 있습니다. 예를 들어서 스타일 관련 설정을 하다가 복잡해지게 되는 상황도 올 수 있고, table 관련 태그를 작성 할 때에도 내용을 div 같은걸로 감싸기엔 애매하죠. 그럴 땐, 리액트의 Fragment 라는 것을 사용하면 됩니다.

App.js

```
import React from 'react';
import Hello from './Hello';

function App() {
  return (
    <>
      <Hello />
      <div>안녕히계세요</div>
    </>
  );
}

export default App;
```

태그를 작성 할 때 이름 없이 작성을 하게 되면 Fragment 가 만들어지는데, Fragment 는 브라우저 상에서 따로 별도의 엘리먼트로 나타나지 않습니다.

```
<div id="root">
  <div>안녕하세요</div>
  <div>안녕히계세요</div>
</div>
```

JSX 안에 자바스크립트 값 사용하기

JSX 내부에 자바스크립트 변수를 보여줘야 할 때에는 `{}` 으로 감싸서 보여줍니다.

```
import React from 'react';
import Hello from './Hello';

function App() {
  const name = 'react';
  return (
    <>
      <Hello />
      <div>{name}</div>
    </>
  );
}

export default App;
```

style 과 className

JSX 에서 태그에 `style` 과 CSS class 를 설정하는 방법은 HTML 에서 설정하는 방법과 다릅니다.
우선, 인라인 스타일은 객체 형태로 작성을 해야 하며, `background-color` 처럼 `-` 로 구분되어 있는 이름들은 `backgroundColor` 처럼 camelCase 형태로 네이밍 해주어야 합니다.

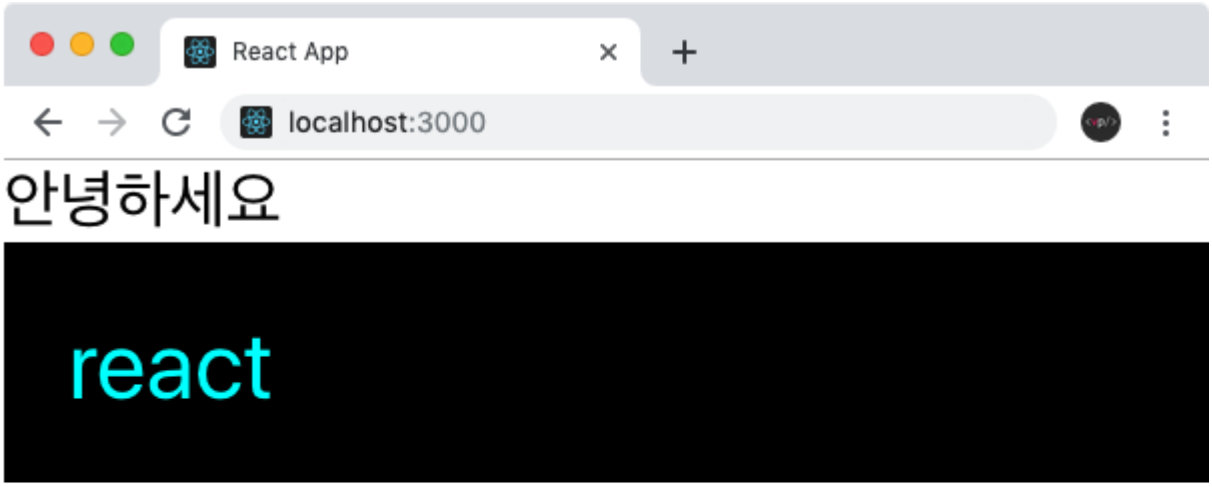
App.js

```
import React from 'react';
import Hello from './Hello';

function App() {
  const name = 'react';
  const style = {
    backgroundColor: 'black',
    color: 'aqua',
    fontSize: 24, // 기본 단위 px
    padding: '1rem' // 다른 단위 사용 시 문자열로 설정
  }

  return (
    <>
      <Hello />
      <div style={style}>{name}</div>
    </>
  );
}

export default App;
```



그리고, CSS class 를 설정 할 때에는 `class=` 가 아닌 `className=` 으로 설정을 해주어야 합니다. 한번, App.css 파일을 열어서 전체 내용을 지운 뒤 다음과 같이 수정해주세요.

App.css

```
.gray-box {
  background: gray;
  width: 64px;
  height: 64px;
}
```

그 다음, App.js 를 다음과 같이 수정해보세요.

App.js

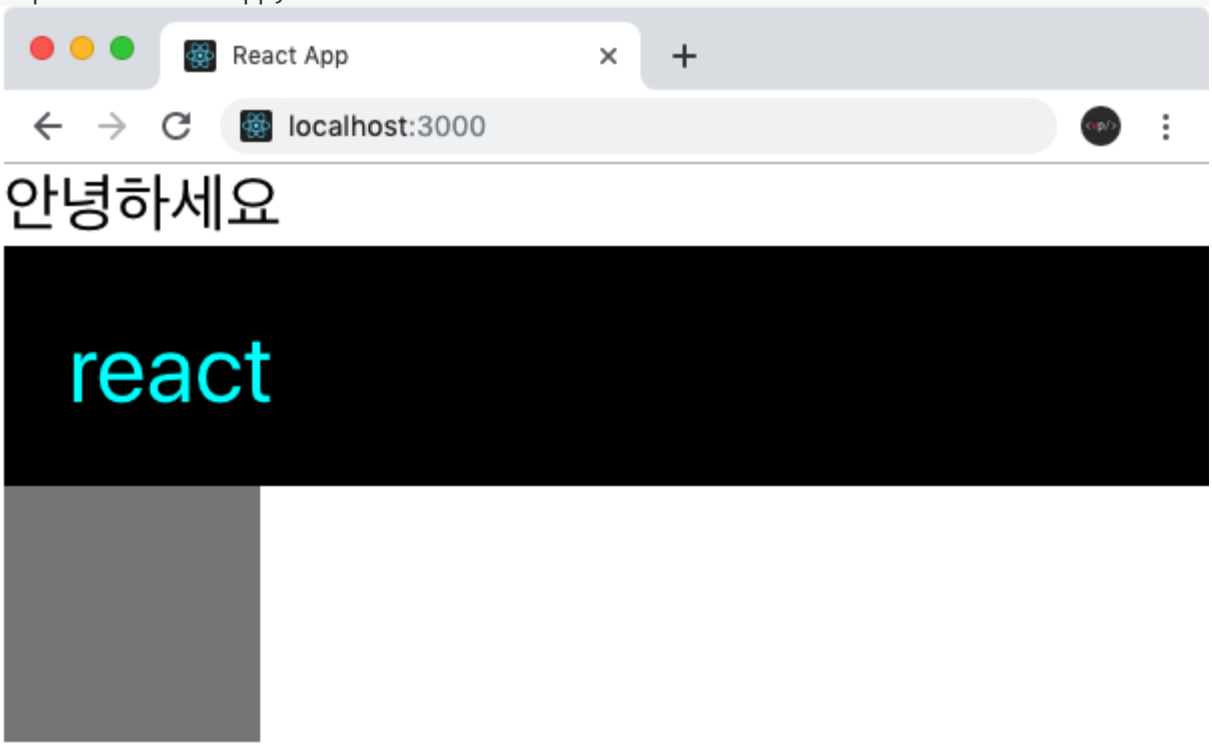
```
import React from 'react';
import Hello from './Hello';
import './App.css';


function App() {
  const name = 'react';
  const style = {
    backgroundColor: 'black',
    color: 'aqua',
```

```
    fontSize: 24, // 기본 단위 px
    padding: '1rem' // 다른 단위 사용 시 문자열로 설정
  }

  return (
    <>
      <Hello />
      <div style={style}>{name}</div>
      <div className="gray-box"></div>
    </>
  );
}

export default App;
```



주석

이제, JSX 에서 주석은 어떻게 작성하는지 알아봅시다.

JSX 내부의 주석은 `{/* 이런 형태로 */}` 작성합니다.
한번 사용해볼까요?

App.js

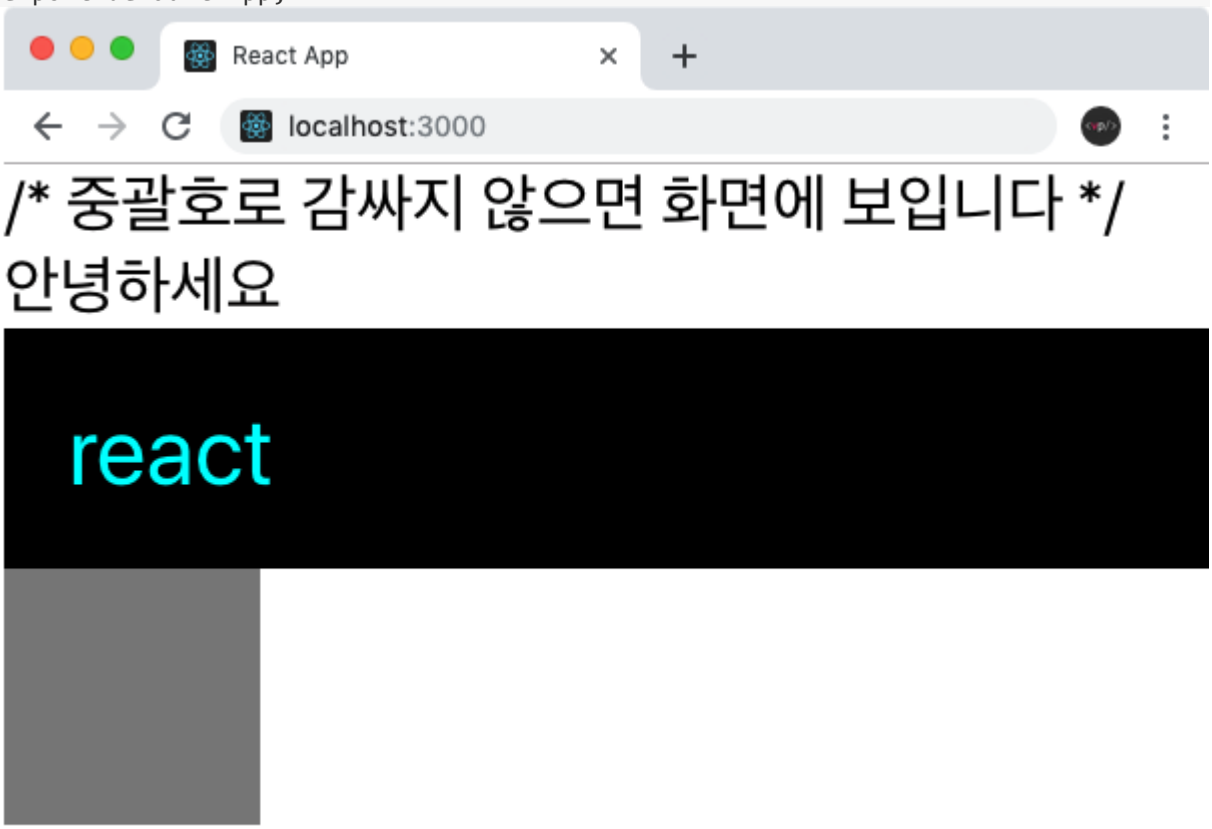
```
import React from 'react';
import Hello from './Hello';
```

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


function App() {
  const name = 'react';
  const style = {
    backgroundColor: 'black',
    color: 'aqua',
    fontSize: 24, // 기본 단위 px
    padding: '1rem' // 다른 단위 사용 시 문자열로 설정
  }

  return (
    <>
      {/* 주석은 화면에 보이지 않습니다 */}
      /* 중괄호로 감싸지 않으면 화면에 보입니다 */
      <Hello
      />
      <div style={style}>{name}</div>
      <div className="gray-box"></div>
    </>
  );
}

export default App;
```



추가적으로, 열리는 태그 내부에서는 `// 이런 형태로도` 주석 작성이 가능합니다.

```jsx
import React from 'react';
```

```jsx
import Hello from './Hello';
import './App.css';


function App() {
  const name = 'react';
  const style = {
    backgroundColor: 'black',
    color: 'aqua',
    fontSize: 24, // 기본 단위 px
    padding: '1rem' // 다른 단위 사용 시 문자열로 설정
  }

  return (
    <>
      {/* 주석은 화면에 보이지 않습니다 */}
      /* 중괄호로 감싸지 않으면 화면에 보입니다 */
      <Hello
        // 열리는 태그 내부에서는 이렇게 주석을 작성 할 수 있습니다.
      />
      <div style={style}>{name}</div>
      <div className="gray-box"></div>
    </>
  );
}

export default App;
```