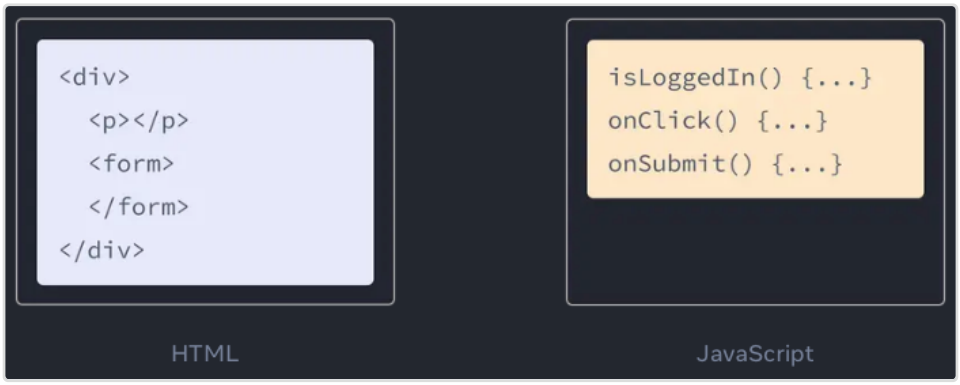
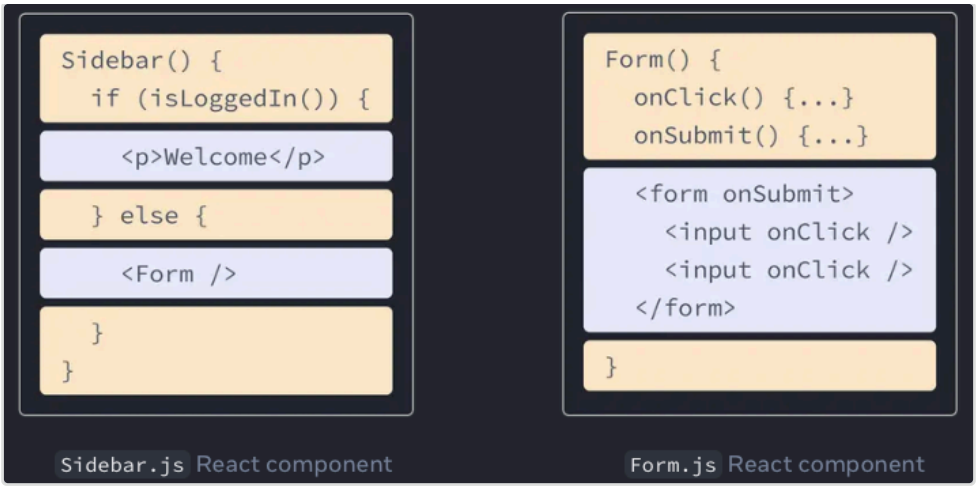
React JSX

개요

JSX는 JavaScript를 확장한 문법이다. JavaScript 파일을 HTML과 비슷하게 마크업으로 작성할 수 있도록 한다. 웹은 HTML, CSS, JavaScript를 기반으로 만들어져 왔다. HTML로 웹 문서 내용을 작성하고 CSS는 디자인, JavaScript로 로직을 작성해 왔다.



최근 Web이 더욱 인터랙티브하게 성장하면서 로직이 내용을 결정하는 경우가 많아졌다. 이런 발전에 맞게 JavaScript가 HTML을 담당하게 되었다. 이것이 React에서 렌더링 로직과 마크업이 같은 위치에 있게 된 이유이다. React Component 안에서 로직과 마크업이 같은 위치에 있게 된다.



버튼의 렌더링 로직과 마크업이 함께 있으면, 변화가 생길 때마다 서로 동기화 상태를 유지할 수 있다.

각 React component는 React가 브라우저에 마크업을 렌더링 할 수 있는 JavaScript 함수이다. React component는 JSX라는 확장된 문법을 사용하여 마크업을 나타낸다.

> JSX는 React와 서로 다른 별개의 개념이다. 함께 사용할 수도 있고 독립적으로 사용할 수도 있다. JSX는 확장된 문법이고 React는 JavaScript 라이브러리이다.

HTML을 JSX로 변환

HTML 코드를 JSX로 변환할 수 있다. 다만, HTML을 그대로 복사해서 사용할 수는 없다. JSX는 HTML 보다 더 엄격한 규칙을 준수한다.

```
<h1>Hedy Lamarr's Todos</h1>
<img
 src="https://i.imgur.com/yXOvdOSs.jpg"
 alt="Hedy Lamarr"
 class="photo"
>
<ul>
    <li>Invent new traffic lights
    <li>Rehearse a movie scene
    <li>Improve the spectrum technology
</ul>
```

JSX의 규칙

1. 하나의 Root element로 반환

한 컴포넌트에서 여러 엘리먼트를 반환하려면 하나의 부모 태그로 감싸야 한다. JSX는 HTML 처럼 보이지만 내부적으로는 일반 JavaScript 객체로 변환되기 때문에 하나의 태그로 감싸야 한다. 하나의 함수에서 두 개의 객체를 반환할 수는 없기 때문이다.

```
<div> //<>
 <h1>Hedy Lamarr's Todos</h1>
 <img
   src="https://i.imgur.com/yXOvdOSs.jpg"
   alt="Hedy Lamarr"
   class="photo"
 >
 <ul>
   ...
```

```
  </ul>
 </div> //</>
```

 태그를 추가하고 싶지 않다면, `<>`, `</>` 빈 태그로 대체할 수 있다. 빈 태그는 Fragment라고 한다. Fragments는 브라우저상 HTML 트리 구조에 흔적을 남기지 않고 그룹화한다.

2. 모든 태그는 닫아야 한다

JSX에서 태그는 항상 명시적으로 닫아야 한다.

```
<>
  <img
    src="https://i.imgur.com/yXOvdOSs.jpg"
    alt="Hedy Lamarr"
    class="photo"
   />
  <ul>
    <li>Invent new traffic lights</li>
    <li>Rehearse a movie scene</li>
    <li>Improve the spectrum technology</li>
  </ul>
</>
```

3. 대부분의 속성은 캐멀 케이스로 작성

JSX는 JavaScript로 변환되고 JSX에서 작성된 속성은 JavaScript 객체의 키가 된다. Component에서 속성을 변수로 읽어야 하는 경우가 있다. 그러나 JavaScript는 변수명에 제한이 있다. 예를 들면, 대시를 포함하거나 `class`와 같은 변수명은 사용할 수 없다.

```
<img
  src="https://i.imgur.com/yXOvdOSs.jpg"
  alt="Hedy Lamarr"
  className="photo"
/>
```

JSX 블록 안에서 JavaScript 사용

JSX를 사용하면 JavaScript 파일에 HTML과 비슷한 마크업을 작성하여 렌더링 로직과 콘텐츠를 같은 곳에 놓을 수 있다. 때로는 JavaScript 로직을 추가하거나 마크업 내부 프로퍼티를 참조하고 싶을 수 있다. 이런 경우 **JSX에서 중괄호를 사용하여 JavaScript를 사용할 수 있다.**

중괄호를 사용하여 JavaScript 호출

```
export default function Avatar() {
  const avatar = 'https://i.imgur.com/7vQD0fPs.jpg';
  const description = 'Gregorio Y. Zara';
  return (
    <img
      className="avatar"
      src={avatar}
      alt={description}
    />
  );
}
```

```
export default function TodoList() {
  const name = 'Gregorio Y. Zara';
  return (
    <h1>{name}'s To Do List</h1>
```

```
  );
}
```

```
const today = new Date();

function formatDate(date) {
  return new Intl.DateTimeFormat(
    'en-US',
    { weekday: 'long' }
  ).format(date);
}

export default function TodoList() {
  return (
    <h1>To Do List for {formatDate(today)}</h1>
  );
}
```

이중 중괄호 사용

JSX에서 JavaScript 표현식뿐만 아니라 객체를 전달할 수도 있다. JavaScript에서 객체는 `{ name: “kim”, age” 5 }`와 같이 표시된다. JSX에서 객체를 전달하려면 `person={ name: “kim”, age” 5 }`과 같이 또 다른 중괄호로 감싸야 한다. 같은 방식으로 JSX의 인라인 CSS 스타일을 적용하는 것을 예로 들 수 있다. 인라인 스타일이 필요할 때 Style Attribute 객체를 전달한다.

```
export default function TodoList() {
  return (
    <ul style={{
      backgroundColor: 'black',
      color: 'pink'
    }}>
      <li>Improve the videophone</li>
      <li>Prepare aeronautics lectures</li>
      <li>Work on the alcohol-fuelled engine</li>
    </ul>
  );
}
```

결론적으로 JSX에서 `{{ }}`는 JSX 안에 선언된 인라인 JavaScript 객체일 뿐이다.

props

React Components는 props를 이용해서 서로 통신한다. 모든 부모 컴포넌트가 props를 가짐으로써 일부 정보를 자식 컴포넌트에 전달할 수 있다. props는 객체, 배열, 함수를 포함한 모든 JavaScript 값이 될 수 있다. 쉽게 생각해서 props는 함수의 인수와 동일한 역할을 한다. props는 Components에 대한 유일한 매개변수이자 인수 역할을 한다. React Components 함수는 하나의 인자, 즉 `props` 객체를 받는다.

Component에 props 전달

아래 props를 전달하지 않은 `Avatar` Component가 있다.

```
export default function Profile() {
  return (
    <Avatar />
  );
}
```

1. 자식 컴포넌트에 props 전달

```
export default function Profile() {
  return (
    <Avatar
      person={{ name: 'Lin Lanying', imageId: '1bX5QH6' }}
      size={100}
    />
  );
}
```

2. read props

```
function Avatar({ person, size }) {
  // person과 size는 이곳에서 사용가능합니다.
}
```

React 컴포넌트 함수는 하나의 props 객체를 받는다.

```
function Avatar(props) {
  let person = props.person;
  let size = props.size;
  // ...
}
```

props 기본값 설정

```
function Avatar({ person, size = 100 }) {
  // ...
}
```

만약 size prop이 없이 `<Avatar person={...} />`가 된다면 `size`는 100으로 설정된다. 기본값은 size prop이 없거나 `size={undefined}`로 전달될 때 사용된다. 만약 `size={null}`, `size={0}`으로 전달되면 기본값은 사용되지 않는다.

JSX spread 문법으로 props 전달

```
function Profile({ person, size, isSepia, thickBorder }) {
  return (
    <div className="card">
      <Avatar
        person={person}
        size={size}
        isSepia={isSepia}
        thickBorder={thickBorder}
      />
    </div>
  );
}
```

때로는 props를 전달할 때 조금 더 간결하게 전달하고 싶을 수 있다. 위 예제에서 보면 `Profile`의 모든 props를 `Avatar`에 전달한다. 이런 경우 spread 문법을 적용할 수 있다. spread 문법을 적용한 경우 props 각각의 이름을 나열하지 않고 전달할 수 있다. (다만, 제한적으로 사용하는

것이 좋다.)

```
function Profile(props) {
  return (
    <div className="card">
      <Avatar {...props} />
    </div>
  );
}
```

자식을 JSX로 전달

JSX 태그 안에 콘텐츠를 중첩하면 부모 컴포넌트는 해당 콘텐츠를 `children` 이라는 prop으로 받는다.

```
import Avatar from './Avatar.js';

function Card({ children }) {
  return (
    <div className="card">
      {children}
    </div>
  );
}

export default function Profile() {
  return (
    <Card>
      <Avatar
        size={100}
        person={{
          name: 'Katsuko Saruhashi',
          imageId: 'YfeOqp2'
        }}
      />
    </Card>
  );
}
```

참고 자료 :

JSX로 마크업 작성하기 – React

공통 컴포넌트 (예: div) – React

컴포넌트에 props 전달하기 – React