React

구 버전의 인터넷 익스플로러 지원하기

인터넷 익스플로러 지원하기

- 인터넷 익스플로러 개요
  - 2015년에 11버전을 마지막으로 더 이상 메이저 버전이 바뀌지 않고 있으며, 2020년 8월을 기준으로 완전히 업데이트가 종료
  - 국내에서는 ActiveX 등의 요인으로 인해서 아직도 높은 점유율
    - 2020년 10월을 기준으로 모든 플랫폼(스마트폰, 데스크톱 포함)에서 5%, 데스크톱 웹 브라우저 9%의 점유율
  - 인터넷 익스플로러에서 웹 페이지를 제대로 운용하려면 최신 자바스크립트 기능을 사용하면 안 됨

인터넷 익스플로러 지원하기

- 최신 자바스크립트 코드를 인터넷 익스플로러에서 실행하기
  - 바벨(Babel): 트랜스 컴파일러, 최신 버전의 자바스크립트로 작성된 코드를 구 버전에서 작동하게 만들어주는 변환기
  - Babel REPL
    - Babel REPL(https://babeljs.io/repl
    - 코드를 입력할 수 있는 왼쪽 영역에 코드를 입력하면 오른쪽 영역에 구 버전에서 실행할 수 있는 코드로 변환되어 출력

```
// 간단한 코드
const a = [1, 2, 3, 4]
a.filter((v) => v % 2 ==
0).forEach(console.log)
// 이 책에서도 다루지 않은 더 최신 버
전의 코드
const b = 10
const c = b ?? 20
```

```
"use strict";
// 간단한 코드
var a = [1, 2, 3, 4];
a.filter(function (v) {
return v % 2 == 0;
}).forEach(console.log);
// 이 책에서도 다루지 않은 더 최신 버전의 코드
var b = 10;
var c = b !== null && b !== void 0 ? b : 20;
```

인터넷 익스플로러 지원하기

- 최신 자바스크립트 코드를 인터넷 익스플로러에서 실행하기
  - 폴리필(Polyfill)
    - 문법적으로 변환을 해주어도 기능적인 변환을 해주지는 않음. 예를 들어 배열의 filter(), forEach() 메소드 등은 구 버전의 인터넷 익스플로러에서 동작하지 않음.
    - 이러한 메소드 등의 기능을 추가해줌
    - 대표적인 폴리필: Modernizr, es-shims, Polyfill.io 등
  - es-shims
    - es5-shim: https://github.com/es-shims/es5-shim
    - es6-shim: https://github.com/es-shims/es6-shim

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/es5-shim/4.5.14/es5-shim.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/es6-shim/0.35.5/es6-shim.min.js"></script>
```

JSX 문법

- 컴포넌트에 여러 요소가 있을 경우 반드시 하나의 HTML 부모 요소로 감싸야 함
- 자바스크립트 표현식 사용 가능
  - { } 내부에 표현
- if 문 대신 조건부 연산자를 사용하여 조건문 기술
  - 삼항 연산식 사용
- AND 연산자를 사용한 조건부 렌더링
- Undefined를 반환하여 렌더링 하지 않기
- 인라인 스타일링
- class 대신 className 사용
- 태그는 반드시 닫기
- 주석
  - /* …… */
  - //

JSX 파일에 html 적용

- JSX는 자바스크립트 확장 문법
  - 브라우저에서 실행되기 전에 코드가 번들링 되는 과정에서 Babel을 사용하여 일반 자바스크립트 코드로 변환됨
- create-react-app으로 프로젝트 생성하면 App.js의 html 코드를 이용하여 홈 화면 그림
- HTML 코드를 작성하는 것과 유사
  - HTML 태그를 사용하는 컴포넌트 작성
- jsx 파일에 CSS 적용
  - App.css
  - import ./App.css

Component 사용

- Component는 특정 코드 내용을 다른 부분에 적용하거나 재사용 하기 위해 만들어 사용하는 코드 블록 단위
- App.js(상위 컴포넌트) 에서 사용
- 함수형 컴포넌트
  - function App() { .... }
  - 화살표 함수로 작성 가능
- 클래스형 컴포넌트
  - class App extends Component { ... }
- 컴포넌트 모듈 내보내기(export)와 불러오기(import)

생명주기 함수

- Component의 생성, 변경, 소멸 과정
- 생성 과정 함수
  - Render()
    - 리턴되는 html 형식의 코드를 화면에 그려줌
    - 화면 내용이 변경되어야 할 시점에 자동 호출됨
  - Constructor(props)
    - 생명주기 함수 중 가장 먼저 실행되면 처음 한번 만 호출됨
    - 컴포넌트 내부에서 사용되는 변수(state)를 선언하고 부모 객체(App.js)에서 전달받은 변수(props)를 초기화
    - Super() 함수는 생성자의 가장 위에서 호출해야 함
  - static getDerivedStateFromProps(props, state)
    - Constructor(props) 함수 다음에 실행
    - 컴포넌트가 새로운 props를 받게 됐을 때 state를 변경함
    - App.js에서 전달한 props_value라는 변수를 props.prop_value로 접근
  - componentDidMount()
    - 가장 마지막으로 실행되는 함수
    - Render() 함수가 return 되는 html 형식의 코드를 화면에 다 그린 후 실행됨
    - 화면이 모두 그려진 후 실행되어야 하는 이벤트처리, 초기화 등에 활용됨

생명주기 함수

- 변경 과정 함수
  - 변경이란 props나 state의 변경을 의미함
  - shouldComponentUpdate()
    - State의 변경일 발생한 경우 실행됨
    - Boolean type의 결과를 리턴하며 리턴 값이 true일 경우 render() 함수를 한번 더 호출함

템플릿 문자열의 사용

- 2015년 발표된 ES6는 많은 기능이 추가되었고 자바스크립트는 이 기술 규격을 따름
- React도 자바스크립트 기반의 언어이므로 ES6의 모든 기능 사용 가능
- 문자열과 변수 합치기
  - + 로 연결
  - 따옴표가 아닌 백틱 문자(`)로 문자열과 변수를 묶어 사용
  - 변수는 ${변수명}으로 표현
  - startsWith(), endWith(), includes() 함수

변수 선언과 접근

- Var 의 단점 보완을 위해 let, const 추가됨(ES6)
- Var
  - 재선언, 재할당 가능
- let
  - 재선언 불가능: Parsing error
  - 재할당 가능
- const
  - 재선언 불가능: Parsing error
  - 재할당 불가능: UncaughtTypeError

전개 연산자

- 배열, 객체 변수를 좀 더 직관적이고 편리하게 합치거나 추출할 수 있게 함
- 변수 앞에 ...(마침표 3개) 입력

props

- properties
  - 컴포넌트 속성 설정
  - 부모 컴포넌트에서 설정하여 자식 컴포넌트에 전달
    - Props를 바꾸려면 부모 컴포넌트에서 가능
  - Props 기본값 설정
    - DefaultProps
  - 태그 사이의 내용을 보여주는 children
    - Props.children
  - 비구조화 할당 문법을 사용하여 props 내부 값 추출
    - Const { name, children } = props;
  - Props type 검증
    - propTypes
    - isRequired를 사용하여 필수 propTypes 설정
  - 클래스 형 컴포넌트에서 props 사용
    - Render 함수에서 this.props로 접근

state

- 컴포넌트 내부에서 바뀔 수 있는 값(변수)
  - 클래스형 컴포넌트: state
  - 함수형 컴포넌트에서 useState 함수를 통해 사용하는 state