

01

React 란

01. React

- FrontEnd JavaScript의 FrameWork 또는 Library
- UI 구성요소를 구축하는 도구
- 최초 공개(V0.3.0) : 2013년 7월
 - Facebook의 엔지니어 조던 워크가 만들
- SPA(Single Page Application)
- Virtual DOM 사용
- create-react-app, webpack, Babel, ESLint 등의 내장 도구가 포함

02. SPA(Single Page Application)

- 기존의 웹 프로그램은 MPA(multi Page Application)방식
 - 하나의 애플리케이션이 여러 개의 페이지로 구성
 - 클라이언트가 새로운 웹페이지를 요청할 때마다 매번 서버로부터 정적 리소스를 내려받아 전체 페이지를 리렌더링(re-rendering)
 - 페이지를 요청할 때마다 리로딩(reloading, 새로고침)이 발생
- SPA 방식은 단 한 개의 페이지만으로 구성.
- 애플리케이션에 필요한 모든 정적 리소스를 최초 단 한 번만 다운로드하여 렌더링.
- 이후 새로운 페이지에 대한 요청이 있을 때에는 페이지 갱신에 필요한 데이터만을 내려 받아 리렌더링
 - 리로딩(reloading, 새로고침)이 발생하지 않음

03. Virtual DOM

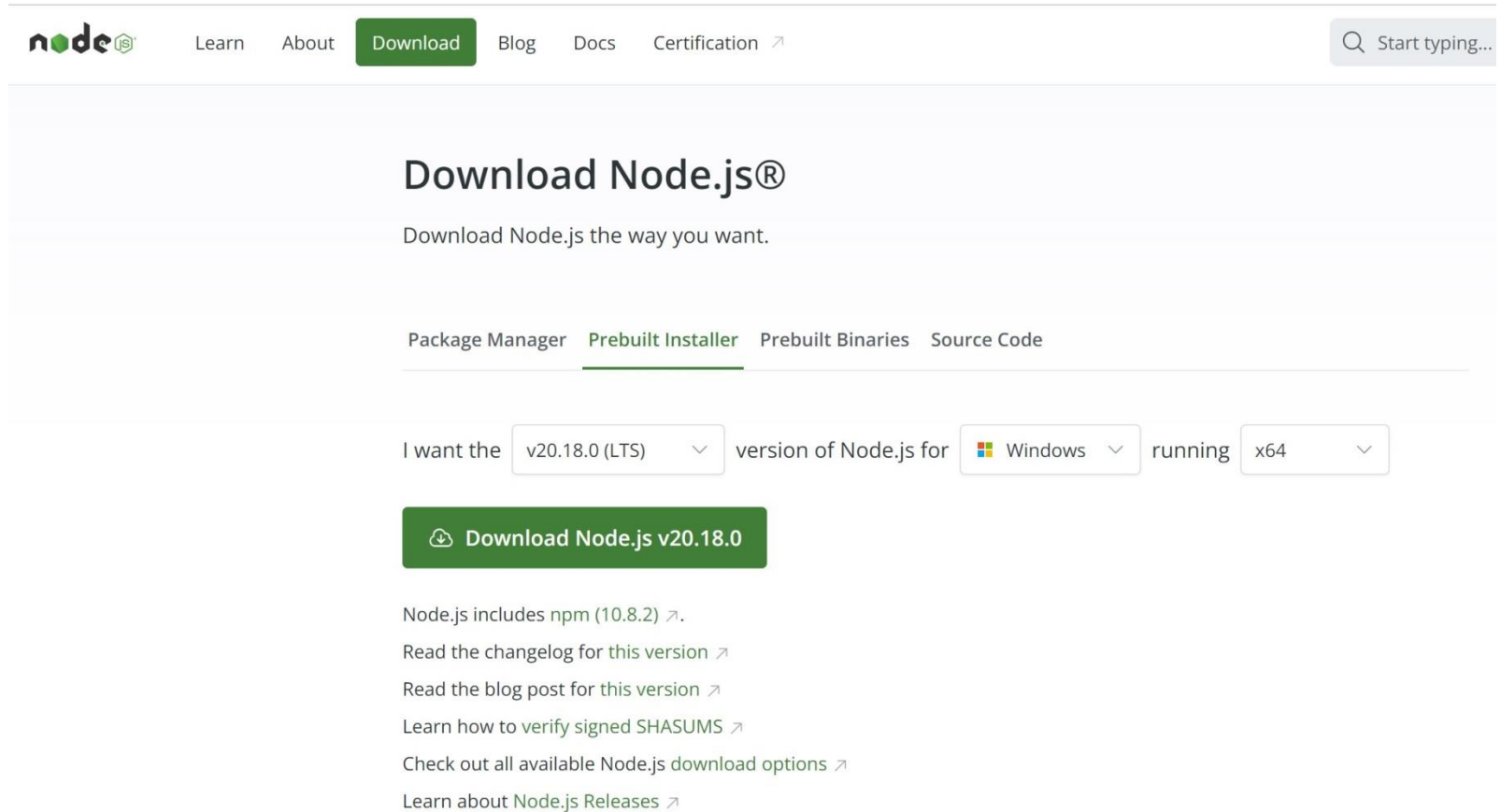
- 메모리 상에서만 존재하는 가상의 DOM
- 상태(state)나 props가 업데이트될때 마다, 변경 사항이 모두 적용된 전체 UI를 새로운 Virtual DOM을 생성하여 렌더링
- 이전 버전의 Virtual DOM과 새로 생성된 Virtual DOM을 서로 비교하여 차이점이 있는 부분만을 실제 Browser DOM에 적용
 - 데이터가 지속적으로 변화하는 동적인 애플리케이션에서는 Virtual DOM을 사용했을 때 성능 향상이 큰 편이지만, 데이터가 자주 변경되지 않는 정적인 애플리케이션에서는 오히려 성능이 느려질 수도 있음.

02

React 개발환경

01 개발환경 설정

■ Node.js 설치



The screenshot shows the Node.js download page. At the top, there is a navigation bar with links for 'Learn', 'About', 'Download' (highlighted in green), 'Blog', 'Docs', and 'Certification'. A search bar on the right contains the text 'Start typing...'. The main heading is 'Download Node.js®', followed by the subtext 'Download Node.js the way you want.' Below this, there are four tabs: 'Package Manager', 'Prebuilt Installer' (which is selected and underlined), 'Prebuilt Binaries', and 'Source Code'. The 'Prebuilt Installer' tab shows a configuration section where the user can select the version of Node.js (currently 'v20.18.0 (LTS)'), the operating system (currently 'Windows'), and the architecture (currently 'x64'). A large green button with a download icon and the text 'Download Node.js v20.18.0' is prominently displayed. Below the button, there are several links for further information: 'Node.js includes npm (10.8.2)', 'Read the changelog for this version', 'Read the blog post for this version', 'Learn how to verify signed SHASUMS', 'Check out all available Node.js download options', and 'Learn about Node.js Releases'.

nodejs Learn About **Download** Blog Docs Certification ↗

Search Start typing...

Download Node.js®

Download Node.js the way you want.

Package Manager **Prebuilt Installer** Prebuilt Binaries Source Code

I want the **v20.18.0 (LTS)** version of Node.js for **Windows** running **x64**

Download Node.js v20.18.0

Node.js includes npm (10.8.2) ↗
Read the changelog for this version ↗
Read the blog post for this version ↗
Learn how to verify signed SHASUMS ↗
Check out all available Node.js download options ↗
Learn about Node.js Releases ↗

01. 개발환경 설정

■ 확장 팩 설치

- Reactjs code snippets(자동완성 지원)
- Prettier-Code formatter(정해진 코딩 스타일로 변환)
- ..
- ..

01. 개발환경 설정

■ Vscode 새로운 폴더를 생성

■ Vscode Terminal

■ `npx create-react-app .`

- 현재 폴더 위치에 react 프로젝트 프레임 워크 생성
- 위 명령이 정상적으로 실행이 안될 시
- `npm install -g npm` (npm 설치)
- Powershell 관리자권한으로 실행 후
- `Set-ExecutionPolicy RemoteSigned` 입력

■ `npm start`

- 첫번째 : `package.json`의 "scripts"안의 "start"를 실행하는 명령
 - "react-scripts start" 가 실행되면서 개발용 서버를 시작
- 두번째 : `node index.js` 를 실행

03

React 기초

01. JSX(Javascript Syntax eXtension)

- Javascript 확장한 문법
- Html 코드와 자바스크립트를 함께 사용하기 위한 문법
- 브라우저에서 실행하기 전에 Babel을 사용하여 일반 자바스크립트 형태의 코드로 변환

01. JSX(Javascript Syntax eXtension)

- 컴포넌트가 리턴하는 JSX는 반드시 하나의 부모요소로 감싸야 한다.
- 자바스크립트 표현식을 작성하려면 JSX내부에서 코드를 { }로 감싸줘야 한다.
- {}표현식안에서는 if문(for문) 대신 삼항 연산자(조건부 연산자) 사용해야 한다.

01. JSX(Javascript Syntax eXtension)

```
function App() {  
  let desc = "";  
  const loginYn = 'Y';  
  if(loginYn === 'Y') {  
    desc = <div>홍창기 입니다.</div>;  
  } else {  
    desc = <div>비회원 입니다.</div>;  
  }  
  return (  
    <>  
      {desc}  
    </>  
  );  
}
```

01. JSX(Javascript Syntax eXtension)

```
function App() {  
  const loginYn = 'Y';  
  return (  
    <>  
      <div>  
        {loginYn === 'Y' ? (<div>홍창기 입니다.</div>) :  
          (<div>비회원 입니다.</div>)}  
      </div>  
    </>  
  );  
}
```

```
function App() {  
  const loginYn = 'Y';  
  return (  
    <>  
      <div>  
        {loginYn === 'Y' && <div>홍창기 입니다.</div>}  
      </div>  
    </>  
  );  
}
```

01. JSX(Javascript Syntax eXtension)

```
function App() {  
  const loginYn = 'Y';  
  return (  
    <>  
      {  
        (() => {  
          if(loginYn === "Y"){  
            return <div>홍창기 입니다.</div>;  
          }else{  
            return <div>비회원 입니다.</div>;  
          }  
        })()  
      }  
    </>  
  );  
}
```

01. JSX(Javascript Syntax eXtension)

- React DOM은 HTML 속성이름 대신 camelCase 프로퍼티 명명 규칙을 사용.
 - (font-size => fontSize)
- 스타일을 적용할 때에도 객체 형태로 전달

```
function App() {  
  const style = {  
    backgroundColor: 'green',  
    fontSize: '12px'  
  }  
  return (  
    <div style={style}>Hello, JinAh!</div>  
  );  
}
```

01. JSX(Javascript Syntax eXtension)

■ class 대신 className

```
function App() {  
  const style = {  
    backgroundColor: 'green',  
    fontSize: '12px'  
  }  
  return (  
    <div className="testClass">Hello, JinAh!</div>  
  );  
}
```

■ For 대신 htmlFor

- `<label htmlFor="inputId">Label Text</label>`
- `<input id="inputId" type="text" />`

■ 주석 사용법

- JSX 내에서 `{/*...*/}` 와 같은 형식을 사용

02. 엘리먼트(Element)

- html의 태그 요소에 해당

```
const element = <p className="title">Hello, World</p>
```

- 불변 객체(immutable object)이기 때문에 일단 생성된 후에는 상태나 속성을 (원칙적으로) 변경 불가
- 변경이 필요하다면 변경된 엘리먼트를 다시 생성하여 새롭게 랜더링

03. 컴포넌트(component)

- 독립적이고 재사용 가능한 코드 조각
- 화면의 UI를 기능적으로 여러 조각으로 구분하여 관리
- 태그요소들을 묶어서 처리
- 함수기반과 클래스기반으로 구분
 - 함수기반의 컴포넌트가 대세로 자리 잡음
 - React 문서에서도 함수기반을 더 활용할 것을 권고

```
function App() {  
  return (  
    <div>  
      <h1>Hello, World!</h1>  
    </div>  
  );  
}  
export default App;
```

03. 컴포넌트(component)

```
import React from "react";

export default class App extends React.Component {
  render() {
    return <h1>Hello, World!</h1>;
  }
}
```

04. export/import

- **export** : 함수를 외부로 제공
- **import** : 함수를 외부에서 가져옴

```
function App() {  
  const name="오지환";  
  return (  
    <div>  
      <Header></Header>  
      <Nav></Nav>  
      <Article></Article>  
    </div>  
  );  
}  
export default App;
```

```
export default function App() {  
  const name="오지환";  
  return (  
    <div>  
      <Header></Header>  
      <Nav></Nav>  
      <Article></Article>  
    </div>  
  );  
}
```

```
import App from "./App";
```

04. export/import

```
function GetName() {  
  return <h1>오지환입니다.</h1>;  
}  
  
function GetAge() {  
  return <h2>33살입니다.</h2>;  
}  
  
export {GetName, GetAge};
```

```
import {GetName, GetAge} from  
"./ConpFunc";  
  
export default function App() {  
  return (  
    <>  
    < GetName />  
    < GetAge />  
    </>  
  );  
}
```

05. props(properties)

- 부모 컴포넌트에서 자식 컴포넌트로 전달하는 데이터
 - `<Header title="환영합니다." body="REACT"></Header>`
 - `title`과 `body`는 묶여서 객체의 형태로 전달
- 객체의 형태로 전달
 - `props.title`, `props.body..` 와 같은 방법으로 접근
- 구조 분해 할당(destructuring assignment) 방법도 가능

```
export default function Article({title, body}){
  return(
    <article>
      <h2>{title}</h2>
      <h3>{body}</h3>
    </article>
  );
}
```

05. props(properties)

■ defaultProps로 기본값 설정하기

```
function Article({title, body}){
  return(
    <article>
      <h2>{title}</h2>
      <h3>{body}</h3>
    </article>
  );
}

Article.defaultProps={
  title:"환영합니다",
  body:"REACT"
}

Function App(){
  <Article title="안녕하세요"></Article>
}
```

06. 구조 분해 할당(destructuring assignment)

■ 배열 구조 분해 (Array destructuring)

```
let x, y, z;  
[x, y] = [1, 2];  
console.log(x, y); // 1 2  
[x, y] = [1];  
console.log(x, y); // 1 undefined  
[x, y] = [1, 2, 3];  
console.log(x, y); // 1 2  
[x, , z] = [1, 2, 3];  
console.log(x, z); // 1 3
```

// 기본값

```
[x, y, z = 3] = [1, 2];  
console.log(x, y, z); // 1 2 3  
[x, y = 10, z = 3] = [1, 2];  
console.log(x, y, z); // 1 2 3
```


06. 구조 분해 할당(destructuring assignment)

■ 객체 구조 분해 (Object destructuring)

```
const obj = { firstName: 'jihwan', lastName: 'Oh' };

// 프로퍼티 키를 기준으로 디스트럭처링 할당이 이루어진다.
//순서는 의미가 없다.

const { lastName, firstName } = obj;

console.log(firstName, lastName); // jihwan Oh
```

06. 구조 분해 할당(destructuring assignment)

```
// 프로퍼티 키가 prop1인 프로퍼티의 값을 변수 p1에 할당
// 프로퍼티 키가 prop2인 프로퍼티의 값을 변수 p2에 할당
const { prop1: p1, prop2: p2 } = { prop1: 'a', prop2: 'b' };
console.log(p1, p2); // 'a' 'b'
console.log({ prop1: p1, prop2: p2 }); // { prop1: 'a', prop2: 'b' }

// 아래는 위의 축약형
const { prop1, prop2 } = { prop1: 'a', prop2: 'b' };
console.log({ prop1, prop2 }); // { prop1: 'a', prop2: 'b' }

// default value
const { prop1, prop2, prop3 = 'c' } = { prop1: 'a', prop2: 'b' };
console.log({ prop1, prop2, prop3 }); // { prop1: 'a', prop2: 'b', prop3: 'c' }
```