1. Keyword

- **Angular vs React vs Vue**

- View를 다루는 라이브러리

- Only Rendering & Update (렌더링과 업데이트에만 관여함.)

\* Not included another functionality (ex. http client, ...)

\* http나 style에 대한 기능이 들어있지 않다.

- Component Based Development

\* 독립적인 코드 블럭 (HTML + CSS + JavaScript)

\* 작업의 단위

- Virtaul DOM

\* 이제는 DOM을 직접 다루지 않음.

\* DOM을 다루는 일을 React에게 위임.

- JSX

\* NOT Templates

\* 치환되는 Template 이 아님.

\* transpile to JS (Babel, TypeScript)

\* 순수한 JS로 transpile 됨.

- CSR & SSR

\* Client Side Rendering

\* Server Side Rendering

2. Angular vs Vue vs React 의 특징

- Angular : 크로스플랫폼. 어떤 플랫폼에서든지 하나의 코드베이스로 결과물을 낼 수 있게 해줌. 모든 기능이 앵귤러 프레임워크 안에서 하나를 이루고 있다. (ex - https, 인증 등..)

- Vue : Progressive JavaScript Framework. 뷰를 리액트처럼 뷰를 다루는 라이브러리처럼 사용할 수도 있고, 앵귤러처럼 여러가지 뷰의 생태계를 끌어와서 Framework 처럼 사용할 수 있게도 해준다. 두마리의 토끼를 잡게 해주는 역할.

- React : UI(User Interface)를 만들기 위한 JavaScript Library. 유저 인터페이스를 표현하고, 동작시켰을 때 변화하는 일에 초점을 맞추고 있다.

3. Component

<!-- HTMLElement --> => 이미지태그

<img src="이미지 주소"/>

<button class="클래스 이름"> 버튼 </button>

<!-- 내가 만든 컴포넌트 -->

<내가지은이름1 name="Mark" /> => 내가 지은 이름을 가지고 내가 지은 attribute를 설정해서 표현해줄 수 있는 하나의 구성.

<내가지은이름 prop={false}>내용</내가지은이름> => 내가 지은 이름의 태그에 prop을 어떤 값으로 지정해준 상태에서 화면에 표현하겠다. 내가 정의한 컴포넌트를 사용하는 것.

<!--

- src, class, name, props : 밖에서 넣어주는 데이터 (밖에서 컴포넌트 태그 안으로 넣어주는 데이터)

- 문서(HTML), 스타일(CSS), 동작(JS) 를 합쳐서 내가 만든 일종의 태그 (형태와 값을 넣은 다음, 문서, 스타일, 동작을 합쳐서 남들에게 사용하라고 제시하는 것.)

-->

4. Component Based Development - Version

가장 위에 링크, 타이틀, 콘텐크, card

5. Component Tree => DOM Tree

- DOM Tree : 내가 만든 컴포넌트가 아닌, 브라우저가 이미 내장되어있는 목록.

- Component Tree : 이미 내장되어있는 태그를 조합, 스타일을 매기고, 동작을 입혀서 문서(스타일, 표현, 동작) 3가지를 한 번에 다루는 형태를 만든 다음 재활용하는 방식을 의미.

6. Virtaul DOM : 가상의 돔

- DOM을 직접 제어하는 경우

\* 바뀐 부분만 정확히 바꿔야 한다.

- DOM을 직접 제어하지 않는 경우

\* 가상의 돔 트리를 사용해서

\* 이전 상태와 이후 상태를 비교하여

\* 바뀐 부분을 찾아내서 자동으로 바꾼다.

7. Virtual DOM - diff 로 변경

8. React Cliet Side Rendering(CSR) <-> Server Side Rendering(SSR)

- 클라이언트 사이드 렌더링 : 어떤 도메인(웹서비스)에 접속 시, html 안에 있는 돔 요소가 보이거나 스타일이 적용되어 예쁜 화면이 보일 것이다.

- 리액트는 JS로 이루어진 하나의 커다란 웹애플리케이션이며, 자바스크립트 전체가 다운로드가 되어야지 사용 가능하다.

- 자바스크립트가 다운로드되면 빈 껍데기의 html이 나오고, 경로로써 JS 파일을 요청한다. JS 파일 안에는 React 웹애플리케이션 전체가 들어있다. 이것을 다운로드 받은 후의 스크립트가 브라우저에서 로드된 후 실행될 것이다. 리액트가 실행된 후 컴포넌트들이 화면에 그려지고, 그때서야 화면이 보이고 유저가 인터렉션을 할 수 있다.

9. React Server Side Rendering(SSR)

- 최초의 이미 HTML로 표현된 HTML을 다운 받게 됨.

- HTML이 다운로드 된 후, JS를 다시 주라고 요청하는 과정에서 HTML 구성요소가 이미 렌더되고 있으므로 화면에 보인다. 대신, JS가 실행되지 않았기 때문에 동작이 불가능하다. 이 화면을 보는 중에 JS가 다운로드가 될 것이다.

- 소스코드를 실행하면 React가 실행될 것이다. React app 이 실행되기 전까지는 여전히 화면이 동작하지 않는다.

- React가 실행되고 난 후에 실제로 JS를 통해서 컴포넌트가 HTML 상에서 다시 동작하는 순간부터 유저는 페이지를 동작할 수 있게 된다.

- React가 SSR 기능을 활용하지 않는다면, 기본적으로 CSR을 기반으로 돌아가게 된다.

10 CSR vs SSR

- CSR

\* JS가 전부 다운로드 되어 리액트 애플리케이션이 정상 실행되기 전까지는 화면이 보이지 않음.

\* JS가 전부 다운로드 되어 리액트 애플리케이션이 정상 실행된 후, 화면이 보이면서 유저가 인터렉션 가능.

- SSR

\* JS가 전부 다운로드 되지 않아도, 일단 화면은 보이지만 유저가 사용할 수 없음.

\* JS가 전부 다운로드 되어 리액트 애플리케이션이 정상 실행된 후, 유저가 사용 가능.

11. 개발환경

- Node.js

\* Installer

\* nvm (node version manager)

# nvm install 14.16.1 -> 특정 버전 설치.

# nvm use 14.16.1 -> 다른 버전 사용.

# nvm alias default 14.16.1 -> 기본버전 설정.

\* LTS(Long Term Support, 안정적임.) vs Current(최신판, 불안정함)

- Browser (Chrome)

- Git

- VSCode

12. 리액트가 하는 일

- 리액트의 핵심 모듈 2개로 리액트가 하는 일 알아보기

\* 1. 리액트 컴포넌트 => HTMLElement 연결하기

import ReactDOM from 'react-dom';

\* 2. 리액트 컴포넌트 만들기

import React from 'react';

============================================================

=> 리액트 컴포넌트를 웹페이지에 표현하려면, HTML 페이지에 컴포넌트를 그려라. 라고 명령하는 함수가 필요함. 이러한 행위가 리액트 프로그램의 시작지점이라고 할 수 있다. 리액트의 메인(시작함수)로 생각하는 것이 좋다.

=> 간단하게 말하자면, ReactDOM.render() 는 React 컴포넌트(JS, JSX 문법을 사용)를 HTMLElement에 연결해주는 역할을 한다.

ReactDOM.render(

<HelloMessage name="Taylor" />,

document.getElementById('hello-example'); => 실제 HTML 엘레먼트를 얻어옴.

);

============================================================

class HelloMessage extends React.Component {

render() {

return (

<div>Hello {this.props.name}</div>

);

}

}

============================================================

13. {React 컴포넌트} - JS, JSX

- "만들어진 리액트 컴포넌트" 를 실제 HTMLElement에 연결할 때 ReactDOM 라이브러리를 이용합니다.

14. {React 컴포넌트} 만들기

- 리액트 컴포넌트를 만들 때 사용하는 API 모음.

15. Use React, ReactDOM Library with CDN

- CDN을 통한 리액트 라이브러리 사용

- 과정 :

C:\Users\youlo>cd C:\React

C:\React> mkdir what-is-react

C:\React>cd what-is-react

C:\React\what-is-react>npm init -y

Wrote to C:\React\what-is-react\package.json:

{

"name": "what-is-react",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC"

}

C:\React\what-is-react>npx serve

Need to install the following packages:

serve

Ok to proceed? (y) y

┌───────────────────────────────────────────────────┐

│ │

│ Serving! │

│ │

│ - Local: [http://localhost:3000](http://localhost:3000/) │

│ - On Your Network: [http://192.168.0.159:3000](http://192.168.0.159:3000/) │

│ │

│ Copied local address to clipboard! │

│ │

└───────────────────────────────────────────────────┘

16. ~~고전~~ 프론트엔드

- HTML로 문서 구조를 잡고,

- CSS로 스타일을 입히고,

- JavaScript로 DOM을 조작합니다.

17. 컴포넌트를 활용한 프론트엔드

- 컴포넌트를 정의하고, 실제 DOM에 컴포넌트를 그려준다.