Front - Week 1

Welcome to Front-end

(프엔 월드에 온 걸 환영해~)

정지원

중앙대 멋사 10기

프론트엔드?

리액트란?

DOM

브라우저 동작 원리

리액트의 기능, 목적

React 실행!

1.

npm, nvm, yarn 설치

Node 설치

macOS

1. $ curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.2/install.sh | bash
2. 터미널 재시작
3. $ nvm —version
   으로 잘 설치 되었는지 확인
4. $ nvm install —lts
   (- 대쉬 2개하고 lts)

Windows

1. node.js 홈페이지에서 Windows Installer를 내려받아 설치
2. 터미널 켜기
3. $ node -v
   로 잘 설치 되었는지 확인

Yarn 설치

macOS

1. Homebrew 설치
   $ /usr/bin/ruby -e “$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/maaster/install)”
2. Yarn 설치
   $ brew update
   $ brew install yarn
   $ yarn config set prefix ~/.yarn
   $ echo ‘export PATH=“$(yarn global bin):$PATH”’ >> ~/.bash_profile

   3. yarn —version

Windows

1. yarnpkg.com/en/docs/install#windows-stable 홈페이지에서 Download Installer를 클릭해 설치
2. 터미널 켜기
3. $ yarn —version
   (- 대쉬 2개하고 version)

   4. yarn —version

VSC extension 설치

1. ESLint : JS 문법 및 코드 스타일 검사해주는 도구
2. Reactjs Code Snippets(charalampos) : 단축 단어 사용을 간편하게 해줌
3. Prettier-Code formatter : 코드 스타일 자동 정리 도구

2.

프론트엔드는 뭐해요?

정보를 주고받기위한 ui 만들기

상상하고 구현한다

🤔 상상이 잘 되고 안되고 가 왜 다를까?

HOW

어떻게 그려야 하는지 **지시**하는 코딩

WHAT

무엇을 그려야 하는지 **묘사**하는 코딩

명령

어떻게 그려야 하는지 **지시**하는 코딩

imperative programming

선언

무엇을 그려야 하는지 **묘사**하는 코딩

declarative programming

구현해 봅시다.

방법 1

```
[0,0] 으로 이동해서 [0,100]까지 검은 직선을 그리고
[100,100]까지 검은 직선을 그리고
[100,0]까지 검은 직선을 그리고
0,0 까지 검은 직선을 그린다.
```

방법 2

```
"0,0 좌표에 위치한 100 x 100 사각형"
```

UI 구현에 적합

이 모든건 추상화를 위해!

추상화 (Abstraction)

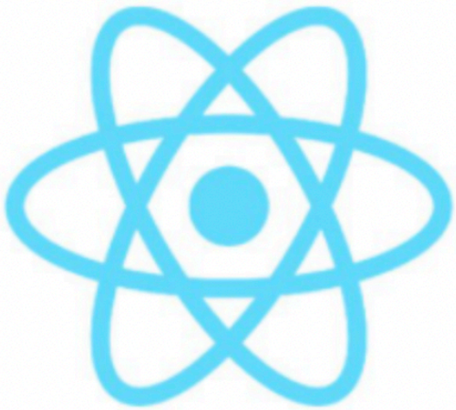
핵심적인 개념 또는 기능을 간추려 내는 것

3.
리액트란?

NETFLIX

facebook

aws



React

airbnb

Discord

리액트는 말이죠옹

리액트는 오직 view만 신경 쓰는 라이브러리

프레임워크

설계는 내가 할게
너는 구현만 해

프레임워크가 제공하는 뼈대와 가이드에
맞게 설계하고 개발
-> 시스템의 통합, 일관성 유지

흐름 제어

라이브러리

필요하면
갖다 써

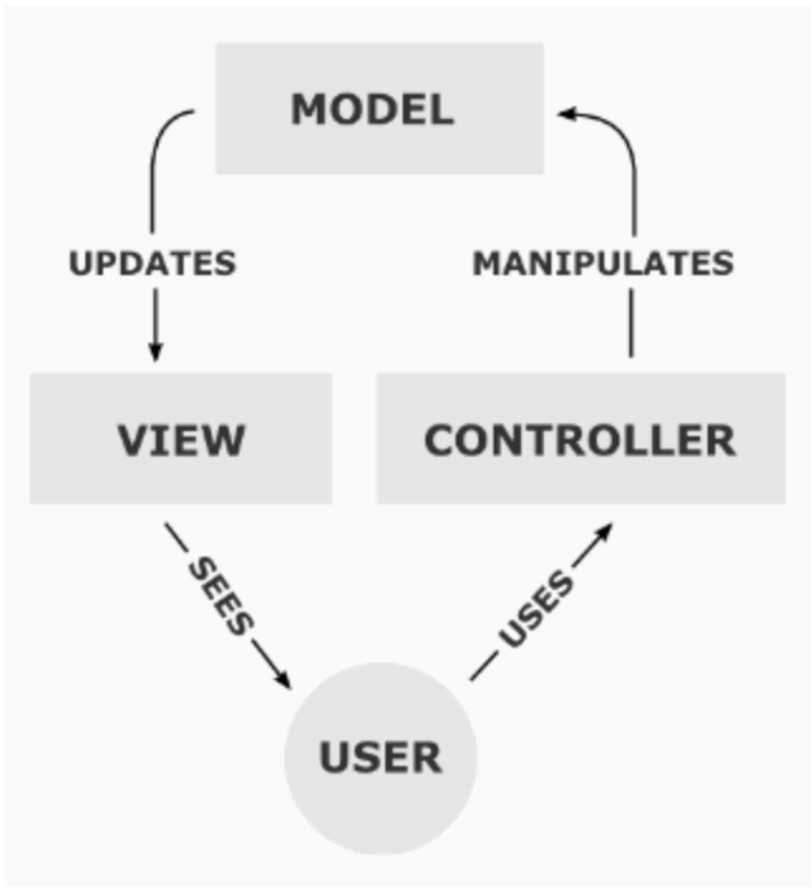
개발하기 위해 필요한 것들을 미리 구현해
놓은 도구
-> 재사용이 용이

흐름 제어 x

MVC 패턴

model-view-controller

model : 데이터를 관리하는 영역
View : 사용자에게 보이는 영역
Controller : (어떤 input을 받으면) 수정, 변경된 사항을 view에 반영(변형, mutate)



리액트는 말이죠옹

리액트는 오직 view만 신경 쓰는 라이브러리

어떤 데이터가 변할 때마다 어떤 변화를 줄지 고민하는 것 x
데이터의 변화 -> 기존 view를 날리고 새로 렌더링 => React

컴포넌트 : 특정 부분이 어떻게 생길지 정하는 선언체 (재사용 가능 API 내장)
렌더링 : 사용자 화면에 view를 보여주는 것

4.

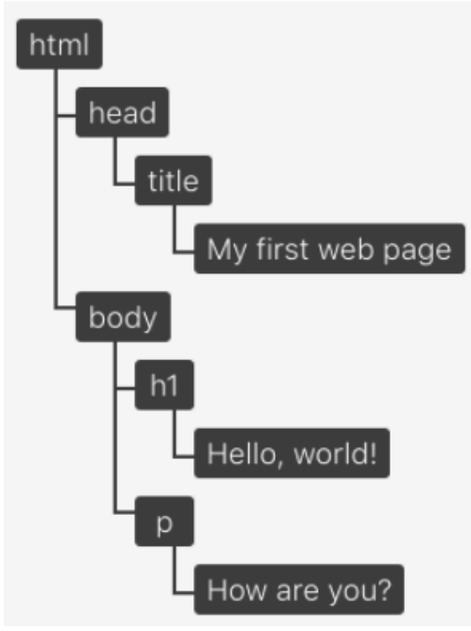
DOM이 뭐에요?

DOM
(Document Object Model)

여러 프로그램들이 페이지의 콘텐츠 및 구조, 스타일을 읽고 조작할 수 있도록 API 제공

```html
<!doctype html>
<html lang="en">
 <head>
   <title>My first web page</title>
  </head>
 <body>
    <h1>Hello, world!</h1>
    <p>How are you?</p>
  </body>
</html>
```

이 문서는 아래와 같은 노드 트리로 표현됩니다.



간단하게 말하자면
HTML 요소들의 구조화된 표현

https://github.com/Jiwon-Jeong99/TIL/blob/main/Browser/DOM.md

DOM 방식

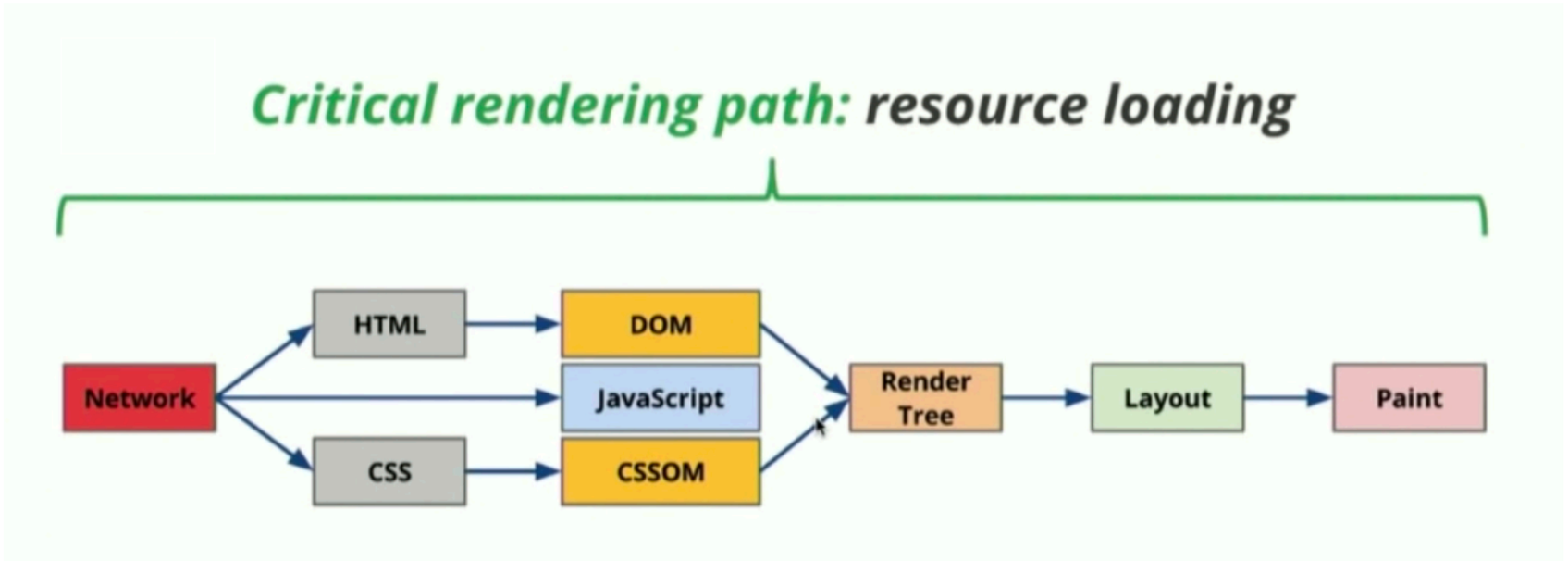
DOM에 변화 -> 웹 브라우저 css 재연산
-> 레이아웃 구성 -> 페이지 리페인트

=> virtual DOM 방식으로 DOM 처리 횟수 최소화

5.

브라우저 동작 원리

브라우저 동작원리



1. DOM 트리 형성

2. 렌더 트리 형성

: 화면에 그려지는 요소 구축

3. 레이아웃 실행

: 노드들의 크기, 위치, 레이어 간 순서를 좌표에 나타냄

4. 페인트

: 색을 입히고, 위치를 결정

6.

React의 기능, 목적

How 명령적

지시

Event handler
: 사용자 input에 따라 외부에 알려줘야 하는 데이터 change

Synchronizing
: 외부(서버)와 일치시킴

What 선언적

묘사

데이터가 변할 부분 구멍 뚫기!

서버사이드 동적 웹페이지

서버에서 구멍 뚫어놓고 요청에 따라 다른 데이터 넣어 응답
-> 화면 업데이트가 새로고침 뿐!

너무 느려!!!

브라우저 속 JS 능력 2가지

1. DOM 조작
2. XMLHttpRequest
-> 새로고침 하지 않고 URL로부터 데이터 받아오기 가능

클라이언트 사이드 동적 웹페이지

JS
1)화면이 처음 그려진 이후에는 2)XMLHTTPRequest로 데이터만 받아와서 3)DOM에 반영시키자!

네트워크 스피드, 렌더링 스피드,
UX 모두 개선!

But, DOM-JS 간의 거리(비용)이 만만치 않아 ㅠㅠ

변경되는 데이터를 서버에서 가져온 다음
기존 화면에서 변화가 생긴 부분만 바꾸자!

1) data의 변화(how, 명령)할 때 생기는
2) view의 변화(what, 선언)를 반영하자!

서버사이드	클라이언트 사이드
View - only 선언적	**View - 선언적 & 명령적**
지금 이순간만 중요! 전체 새로고침	**이전과 비교 Virtual DOM!**

Virtual DOM
(리액트 특징)

실제 DOM의 사본 같은 것

1. 데이터를 업데이트하면 전체 UI를 DOM에 리렌더링
2. 이전 virtual DOM과 비교
3. 바뀐 부분만 실제 DOM에 적용

클라이언트 사이드 렌더링 유지하면서 view 만들려니 How(명령) + What(선언)이 혼합

React의 목적

⭕매번 처음 그리는 것처럼

1) UI에 표시될 값 find
2) 구멍 뚫린 view에 데이터 put
3) 완성된 view

❌화면 업데이트

1) UI에 표시될 값 find
2) 데이터에서 달라진 부분 찾고
3) 달라진 데이터에 따라 변경해야 할 view 수정

리액트 컴포넌트 작성 규칙

1. 항상 처음 그리는 것처럼
2. 데이터는 이미 다 준비되어 있는 것처럼
3. 순수 함수 분리

다른 프레임워크와의 차별점

hook

클래스 : 컴포넌트 안에 함수까지 존재 -> 재사용 어려움

Hook : 함수를 따로 빼서 만듦 -> 재사용 용이

프엔 개발툴의 최종 목표

선언적(view의 변화, 구멍뚫기)
& 명령적(data의 변화)을 섞지마!

리액트의 최종 목표

아예 컴포넌트 만들 때도
선언 & 명령을 섞지마!

최대한 선언적으로 유지하자!

7.

create-react-app으로 프로젝트 생성

Node 설치 이유

웹 브라우저 환경이 아닌 곳에서도 JS를 사용하여 연산 가능
->웹서버, 모바일앱, 웹앱으로 확장

- ECMAScript 6 = es6+ 문법
- **바벨(babel)** | ES6를 호환시켜 줌
- **웹팩(Webpack)** | 모듈화된 코드를 한 파일로 합치고(번들링) 코드를 수정할 때 마다 웹 브라우저를 리로딩
- **npm** | node.js의 패키지 매니저 도구

  → npm으로 재사용 가능한 코드, 즉 패키지를 설치하고 버전 관리 가능

  yarn | npm을 대체할 수 있는 도구. npm보다 빠르며 효율적인 캐시 시스템 제공
- **nvm** | node.js를 여러 버전으로 설치하여 관리해주는 도구

  → 추후 node.js 버전 업데이트하거나 프로젝트별로 버전이 다를 경우 용이

Create-react-app

리액트 프로젝트 생성 시 필요한 웹팩, 바벨의 설치-설정 과정을 생략하고 바로 작업환경을 구축해 주는 도구
-> 추후 커스터마이징으로 자유롭게 설정 변경 가능

React 실행해보기

1. $ yarn create react-app <프로젝트 이름>
2. $ yarn create react-app <프로젝트 이름>
3. $ cd <프로젝트 이름>
4. $ yarn start

8.
프론트엔드 공부 추천!

중앙대 멋사 10기

클라이언트 사이드	서버 사이드 렌더링
React	next.js
Typescript	nest.js

Thank You