#2021_10

WebXR

- Metaverse 도전기

lovejaeuk@gmail.com

메타버스 한번 만들어볼까?

Homepage

Metaverse Client

Server

lovejaeuk@gmail.com

WebXR

WebXR?

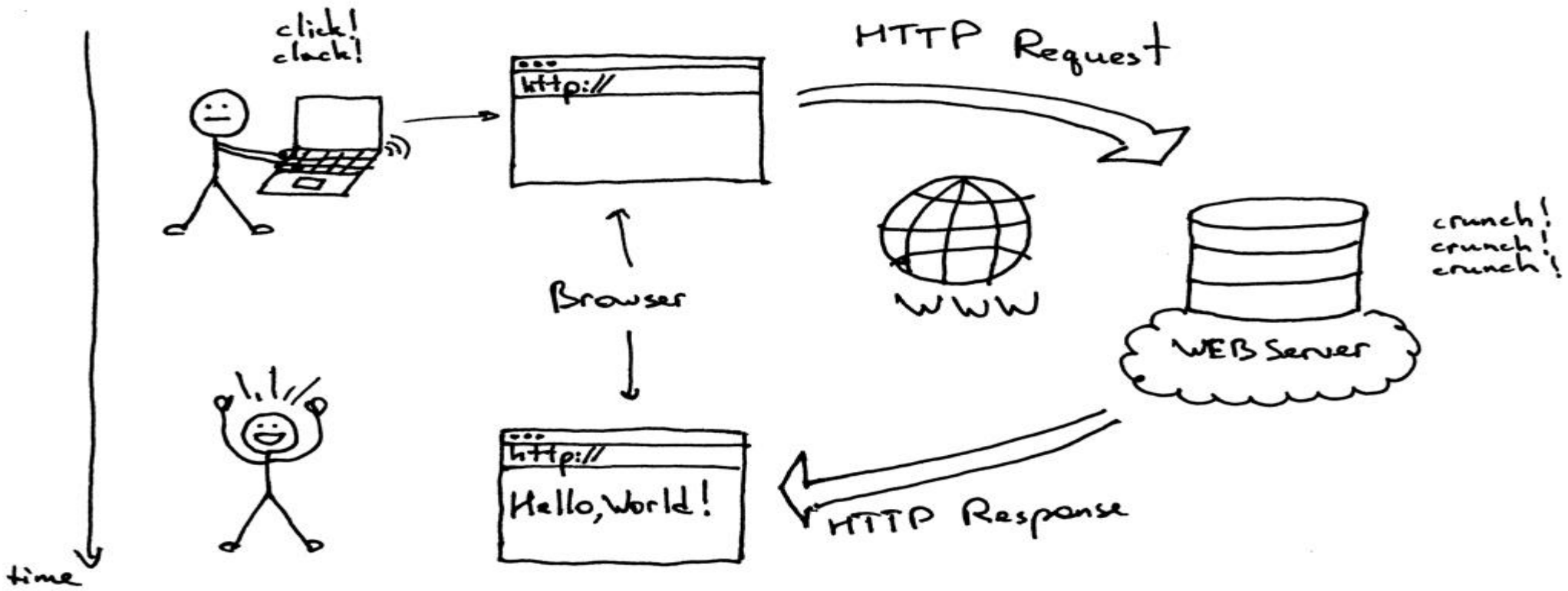
- Web + XR
- WebXR Device API
  - https://github.com/immersive-web/webxr/blob/master/explainer.md
  - https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API/Fundamentals

- Hello-WebXR
  - https://mixedreality.mozilla.org/hello-webxr/

lovejaeuk@gmail.com

Web

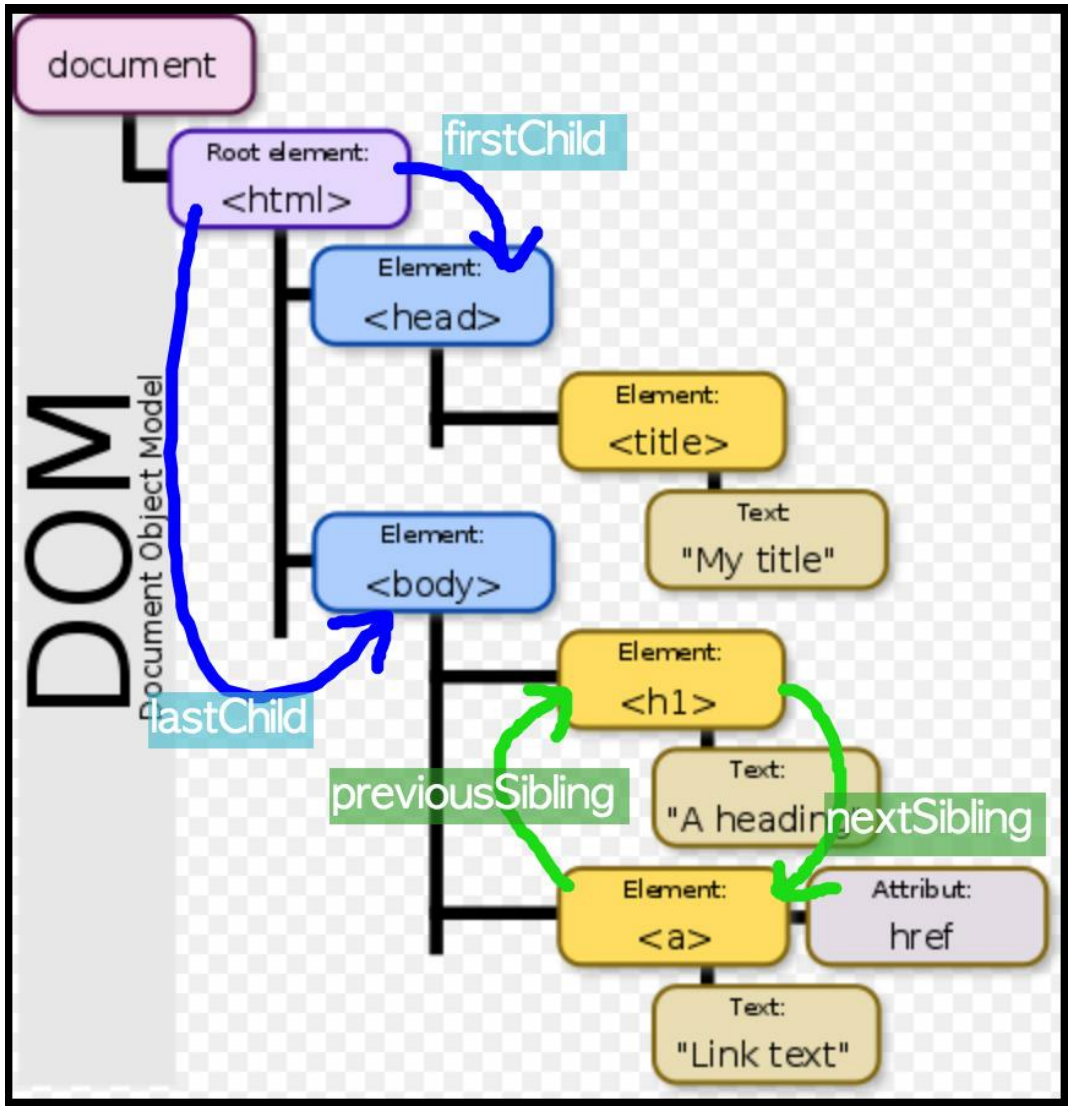
HTTP

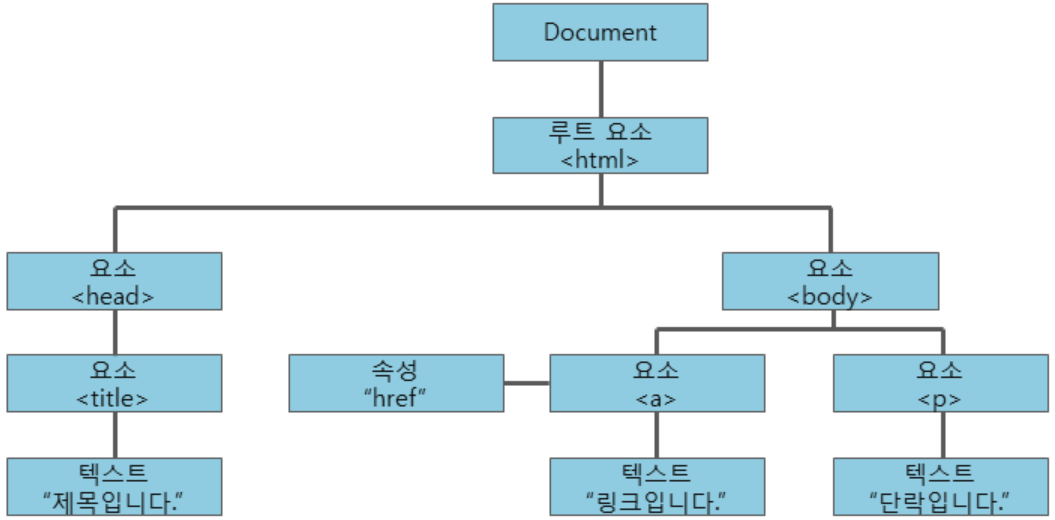


https://jaejade.tistory.com/52

https://developer.mozilla.org/ko/docs/Learn/Server-side/First_steps/Client-Server_overview

DOM


document
Root element:
<html>
firstChild
Element:
<head>
Element:
<title>
Text
"My title"
Element:
<body>
Element:
<h1>
Text:
"A heading
Element:
<a>
Attribut:
href
Text:
"Link text"
DOM
Document Object Model
lastChild
previousSibling
nextSibling


Document
루트 요소
<html>
요소
<head>
요소
<body>
요소
<title>
속성
"href"
요소
<a>
요소
<p>
텍스트
"제목입니다."
텍스트
"링크입니다."
텍스트
"단락입니다."

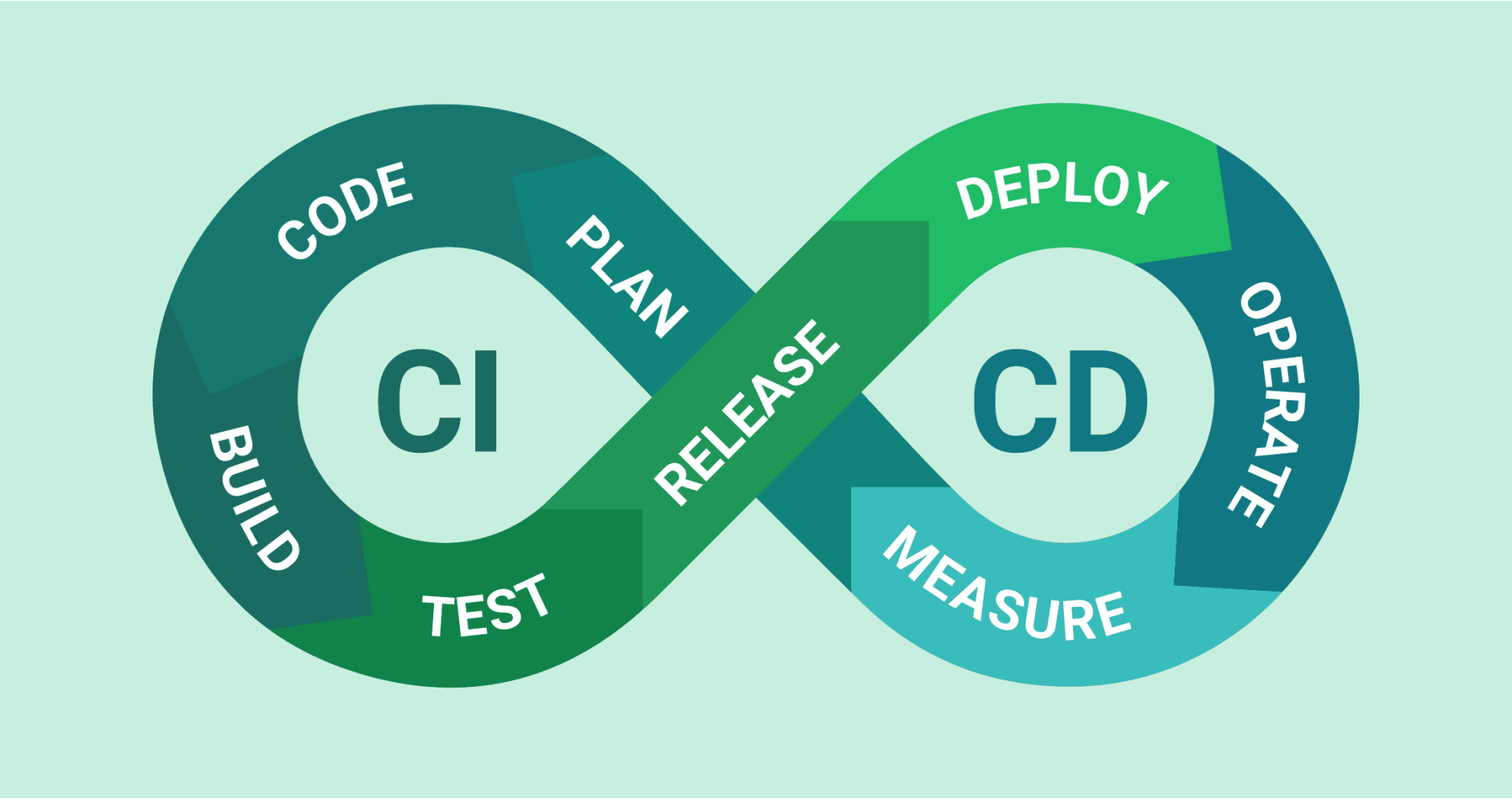
html, css, js

- 렌더링 엔진은 HTML문서 파싱 => DOM 트리 구축 => CSS 마크업을 파싱 => 앞서 구축한 DOM 트리와 함께 렌더링 트리 생성

- 서버가 준 결과물 (= HTML, CSS, JS) => 우리가 보는 웹 페이지가 완성된다.

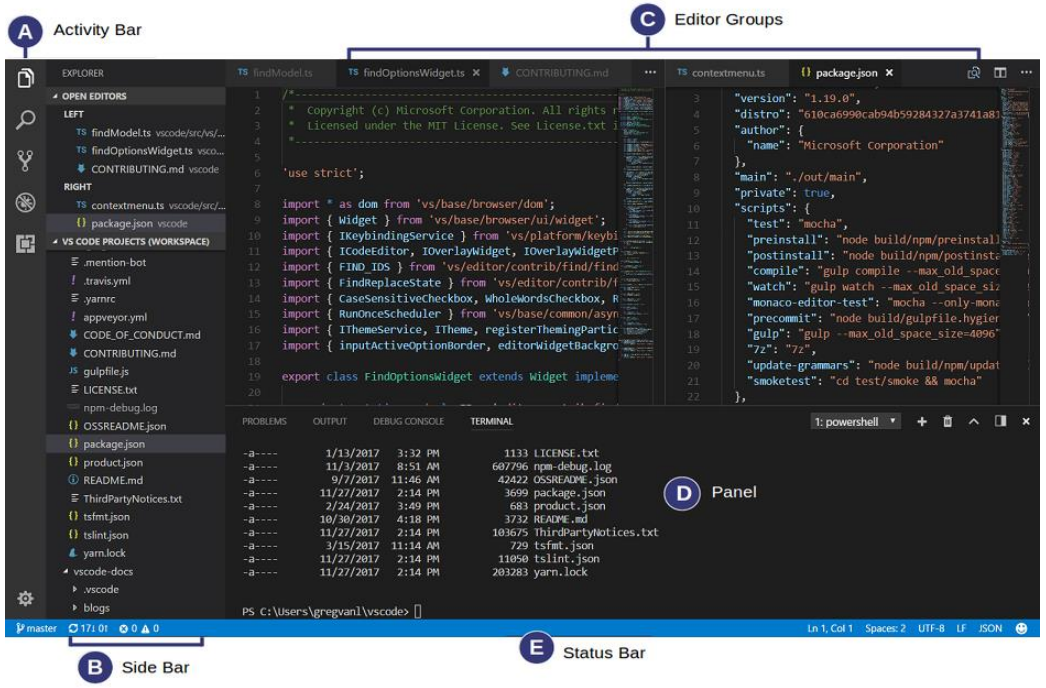
종류	특징
HTML	뼈대를 잡아주는 것
CSS	꾸미는 것, 색, 사이즈 등 예쁘게 꾸미는 것
JavaScript	움직이는 것, 클릭하면 이동하는 것

CI / CD

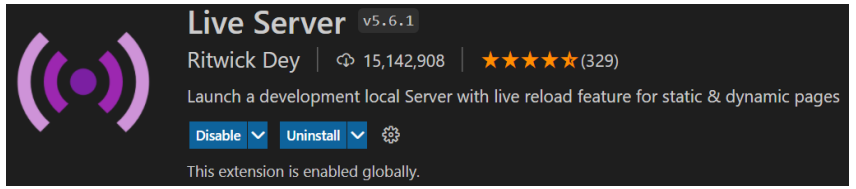


개발환경

- Visual Studio Code
  - ➢ https://code.visualstudio.com/
  - ➢ v체크: Code(으)로 열기" 작업을 Windows 탐색기 디렉터리의 상황에 맞는 메뉴에 추가



- extentions
  - ➢ Live Server



개발환경

- GitHub
  - https://github.com/

- Git
  - https://tortoisegit.org/

- Netlify
  - https://www.netlify.com/

lovejaeuk@gmail.com

참조사이트

- w3schools
  - https://www.w3schools.com/
- MDN Web Docs
  - https://developer.mozilla.org/ko/docs/Learn/HTML
- TCPSchool
  - http://tcpschool.com/html/intro

Web Page만들기

https://docs.emmet.io/abbreviations/syntax/

Debug Javascript

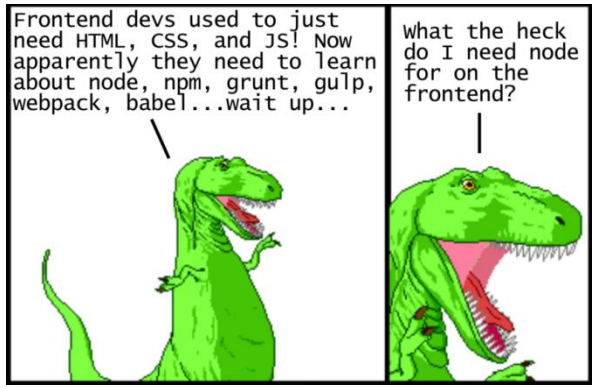
https://developer.chrome.com/docs/devtools/javascript/

https://googlechrome.github.io/devtools-samples/debug-js/get-started

Modern Javascript

Modern Javascript

- Node
  - https://nodejs.org/ko/download/
- ES2015
- Module bundler
  - Webpack



https://medium.com/the-node-js-collection/modern-javascript-explained-for-dinosaurs-f695e9747b70

Babylonjs

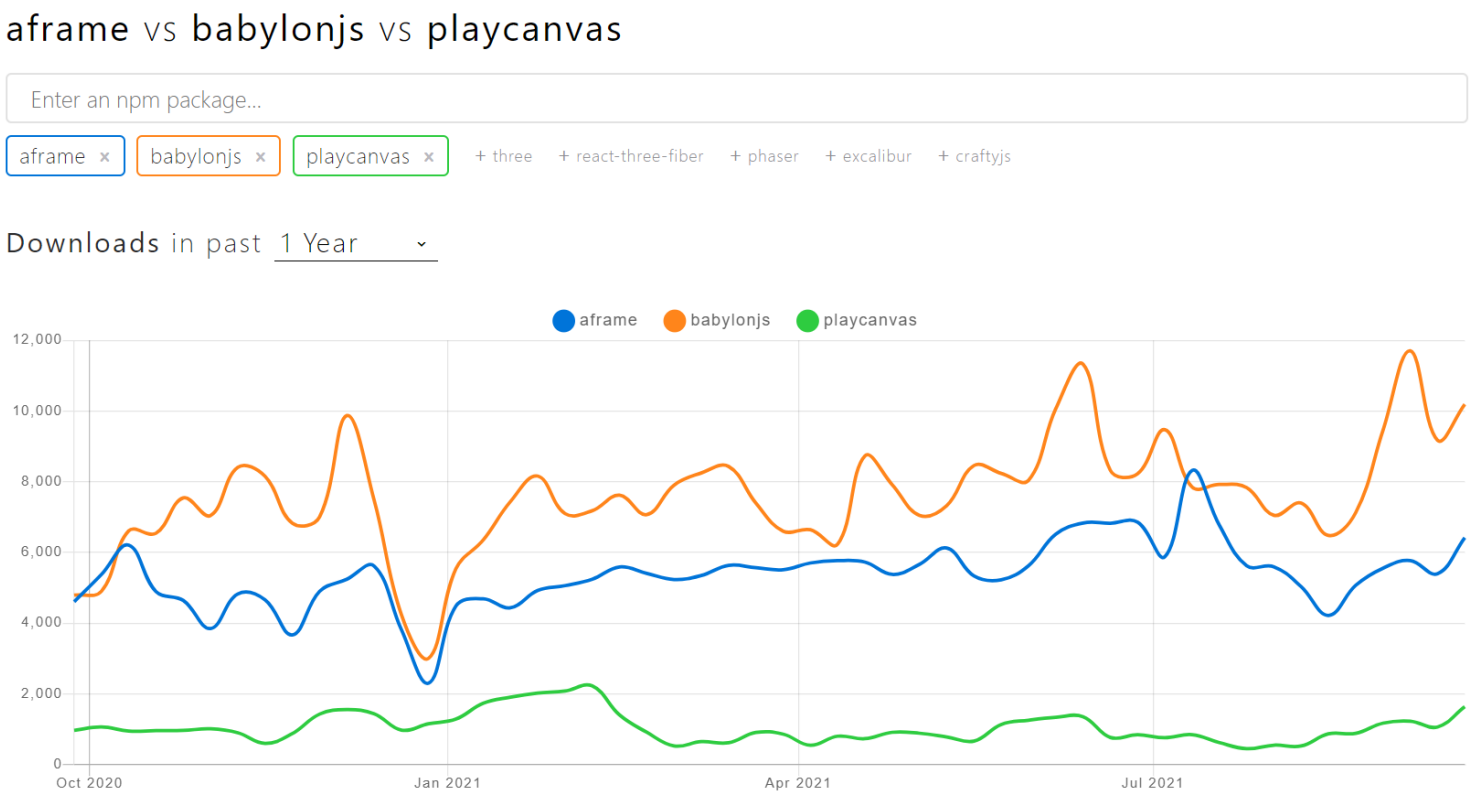
WebGL

- WebGL
  - https://developer.mozilla.org/ko/docs/Web/API/WebGL_API

- <canvas>

WebGL 프레임워크

- A-Frame
- Babylon.js
- Three.js



https://www.npmtrends.com/

npm trends

WebGL 프레임워크

- Babylon.js vs Three.js
- WebGPU

https://webdoli.tistory.com/421

https://roseline.oopy.io/dev/web-3d-graphics

https://doc.babylonjs.com/advanced_topics/webGPU

lovejaeuk@gmail.com

Babylon.js

- https://www.babylonjs.com/
- 사례

https://babylonjs.medium.com/babylon-js-and-frame-pushing-remote-work-forward-735e1a5b98ae

Babylon.js tooling

- Sandbox
  - https://sandbox.babylonjs.com

- Playground
  - https://playground.babylonjs.com/

https://babylonjs.medium.com/a-brief-summary-of-babylon-js-tooling-14fb6c0b5fec

lovejaeuk@gmail.com

Typescript

- Reference
  - https://www.typescriptlang.org/docs/handbook/2/everyday-types.html

lovejaeuk@gmail.com

Server

Express Server

- Express/Node
  - https://expressjs.com/ko/starter/hello-world.html
  - https://developer.mozilla.org/ko/docs/Learn/Server-side/Express_Nodejs/Introduction

서버 호스팅

- Heroku
  - https://id.heroku.com/login

https://velog.io/@bvv8808/heroku1

Socket.IO

- Socket.IO
  - https://socket.io/get-started/chat

- Chat Reference
  - https://github.com/bradtraversy/chatcord

https://www.youtube.com/watch?v=jD7FnbI76Hg

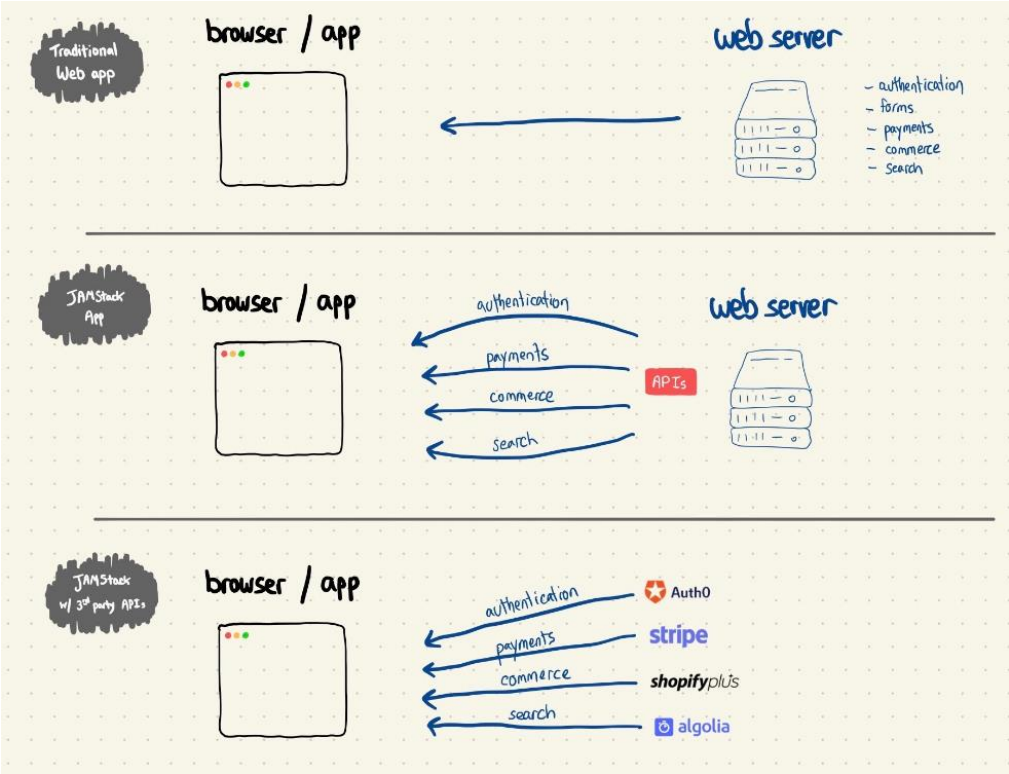
REST API

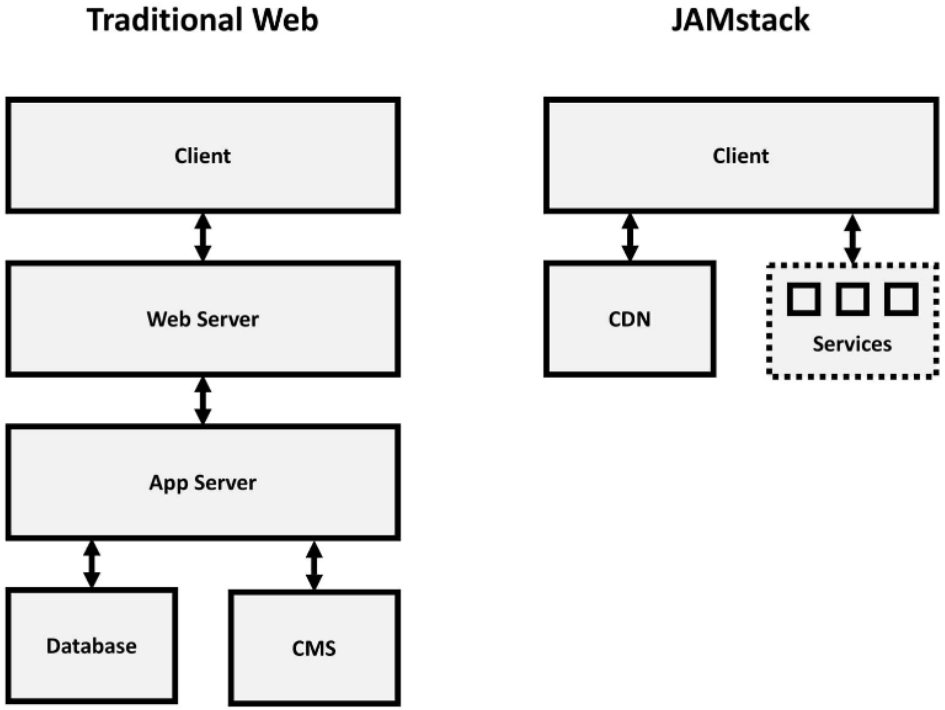
- REST의 개념
  - https://gmlwjd9405.github.io/2018/09/21/rest-and-restful.html
- Test Tool
  - insomnia: https://insomnia.rest/download

- Reference
  - https://github.com/tharindulucky/blog-api-nodejs-mysql

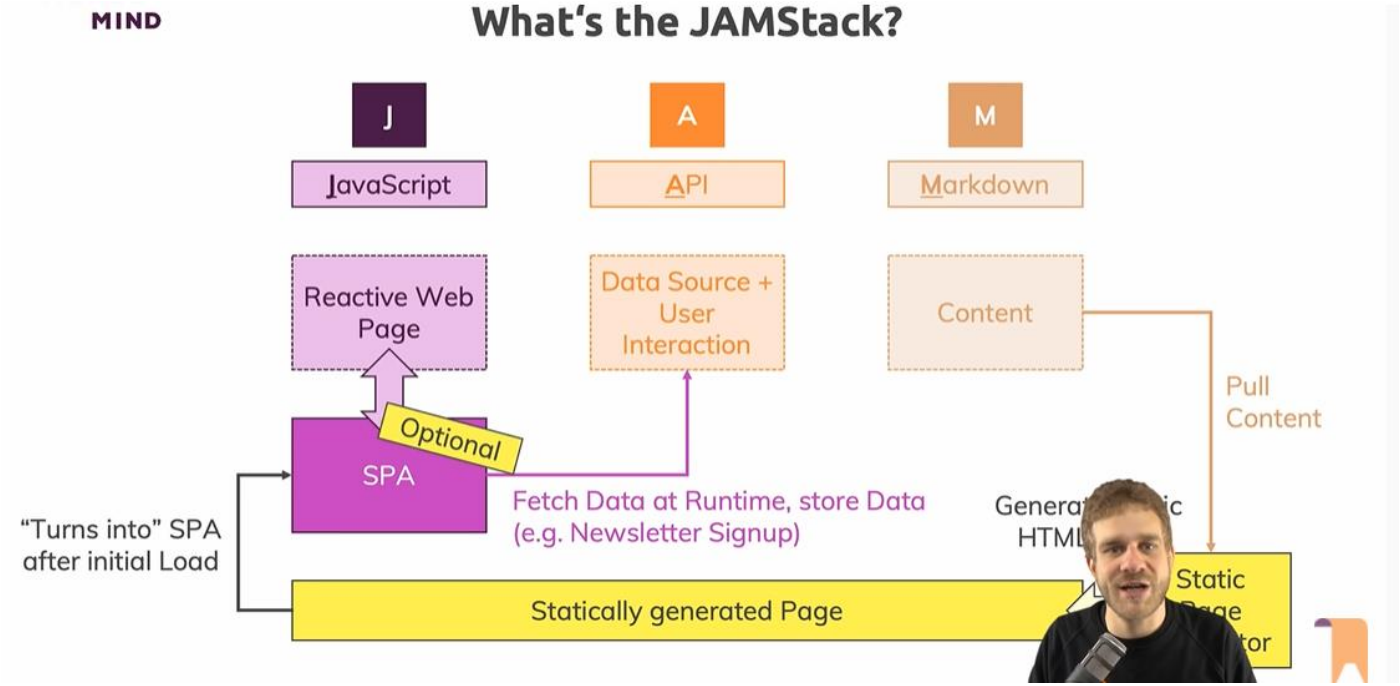
https://www.youtube.com/watch?v=5_CJIWy8uE0&list=PLG3j59vX4yLHA-wCw7KDP-i0r10ZrckqG

JAMStack


Traditional Web app
browser / app
web server
- authentication
- forms
- payments
- commerce
- search
JAMStack App
browser / app
authentication
payments
commerce
search
APIs
web server
JAMStack w/ 3rd party APIs
browser / app
authentication
payments
commerce
search
Auth0
stripe
shopifyplus
algolia

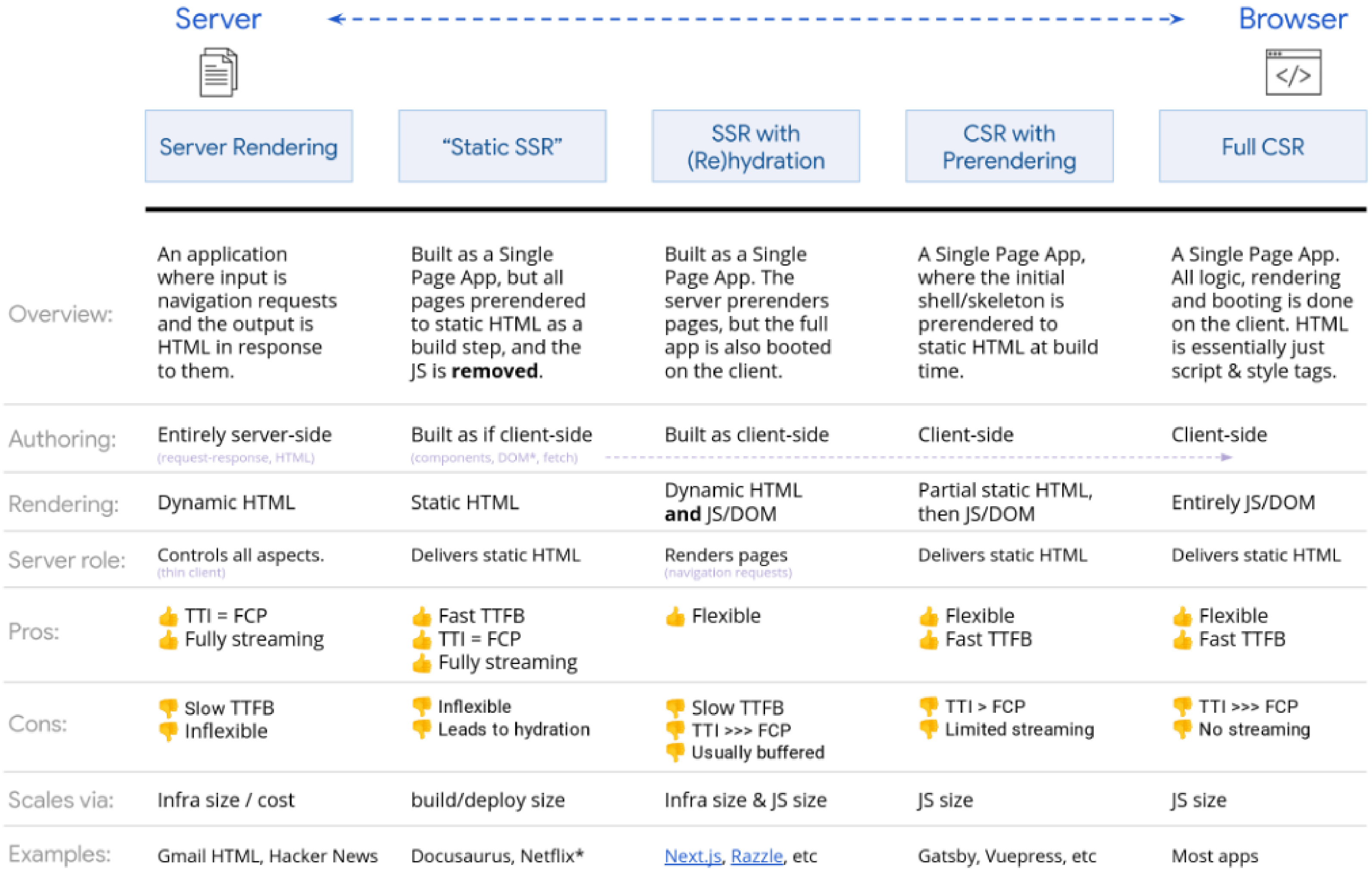

Traditional Web
Client
Web Server
App Server
Database
CMS
JAMstack
Client
CDN
Services

JAMStack



https://www.youtube.com/watch?v=Y8PXMbr0Kqo

Static Generation / Server-side Rendering



	Server Rendering	"Static SSR"	SSR with (Re)hydration	CSR with Prerendering	Full CSR
Overview:	An application where input is navigation requests and the output is HTML in response to them.	Built as a Single Page App, but all pages prerendered to static HTML as a build step, and the JS is **removed**.	Built as a Single Page App. The server prerenders pages, but the full app is also booted on the client.	A Single Page App, where the initial shell/skeleton is prerendered to static HTML at build time.	A Single Page App. All logic, rendering and booting is done on the client. HTML is essentially just script & style tags.
Authoring:	Entirely server-side (request-response, HTML)	Built as if client-side (components, DOM*, fetch)	Built as client-side	Client-side	Client-side
Rendering:	Dynamic HTML	Static HTML	Dynamic HTML **and** JS/DOM	Partial static HTML, then JS/DOM	Entirely JS/DOM
Server role:	Controls all aspects. (thin client)	Delivers static HTML	Renders pages (navigation requests)	Delivers static HTML	Delivers static HTML
Pros:	👍 TTI = FCP 👍 Fully streaming	👍 Fast TTFB 👍 TTI = FCP 👍 Fully streaming	👍 Flexible	👍 Flexible 👍 Fast TTFB	👍 Flexible 👍 Fast TTFB
Cons:	👎 Slow TTFB 👎 Inflexible	👎 Inflexible 👎 Leads to hydration	👎 Slow TTFB 👎 TTI >>> FCP 👎 Usually buffered	👎 TTI > FCP 👎 Limited streaming	👎 TTI >>> FCP 👎 No streaming
Scales via:	Infra size / cost	build/deploy size	Infra size & JS size	JS size	JS size
Examples:	Gmail HTML, Hacker News	Docusaurus, Netflix*	Next.js, Razzle, etc	Gatsby, Vuepress, etc	Most apps

감사합니다.