

바닐라 js 클론코딩

자바스크립트는 웹에서 쓰이는 유일한 언어

유일하므로 기존 미비한것에 계속 업데이트되는 언어, 덧붙여지고..

모든 브라우저에 내장 , 폰도

3d 모델링, 게임 등 js로 가능, 여러가지..

- 프레임워크 - 리액트 네이티브
- 일렉트론 - 앱만들수있음 - vscode로 만듦
- js로 백엔드도 가능
- replit - 온라인 ide - vscode랑 동일 기능하게 해줌

2. js 기초 문법 - 넘어감

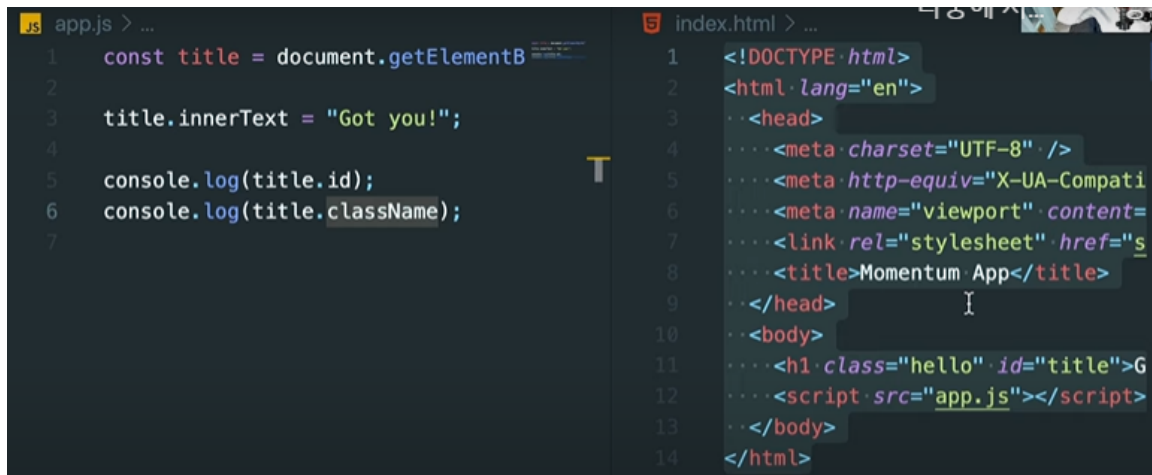
- console에 document 통해 작성한 html 가져올 수 있음, 자바스크립트 관점으로 보여 주는 것임

브라우저에 이미 정의된 객체임 = document

자바스크립트는 html을 읽어온다

자바스크립트에서 ex) document.title - title은 html에 내가 적어놓은 값임

- document.getElementById("id1")
html 태그 중 id 찾아서 가져옴
태그 하나에도 아주많은 속성들이 있음
title, innerText 등



```
js app.js > ...
1  const title = document.getElementById('title');
2
3  title.innerText = "Got you!";
4
5  console.log(title.id);
6  console.log(title.className);
7
index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  <meta charset="UTF-8" />
5  <meta http-equiv="X-UA-Compatible" content="IE=edge">
6  <meta name="viewport" content="width=device-width, initial-scale=1">
7  <link rel="stylesheet" href="style.css">
8  <title>Momentum App</title>
9  </head>
10 <body>
11 <h1 class="hello" id="title">Got you!</h1>
12 <script src="app.js"></script>
13 </body>
14 </html>
```

이렇듯 js 에서 html을 읽고 쓸수있음

- `const hellos = document.getElementsByClassName("hello");`
위는 id로 가져오는 거고 class로 가져오는 것 - 리턴이 array로 되겠지
- `document.querySelector(".hello h1")`
hello라는 클래스에 있는 h1 태그를 불러온다
쿼리셀은 단 하나의 엘리먼트만 리턴해준다 - 클래스와 달리
같은 조건 엘리가 여러개면 첫번째거만 나옴
`querySelectorAll` 이면 모든거 어레이로 리턴
id도 가능 ("`#id1`")
- 이 리턴된 ob 정보를 보려면 `console.dir` (log 대신)
함수 뭐써야할지 모르겠으면 on~ 인것들을 찾아보면됨
`onmouseenter` 있으면 아래 click 있는 부분에 `mouseenter` 넣으면됨
- on~은 다 이벤트들
`title.addEventListener("click", handleTitleClick);`
클릭하면 저 함수를 실행함
자바스크립트가 대신 평션을 실행하도록 넘겨주는 것임

이러면 나중에 리스너 remove 도 가능

title.onclick = handleClick; 동일 기능

- style 내부엔 css? 속성들이 있음

title.style.color 등

- 여러 다른 이벤트

window.addEventListener("resize", ~) (window는 기본제공)

resize offline copy 등

- document.body head title 만 가능

div h1은 안됨, 위에건 특별하게 제공하는 것임

이런건 다 쿼리로 찾아와야함

- style은 js에서 건드리지 않는게 좋음 - css에서 하는게 좋음

html 문서엔 link href로 css와 / script src로 js와 연결되어있음

- 그냥 내가 만든 class 이름을 적으면 error의 위험이 있다.

스펠링을 잘못 적어도 뭐가 잘못 되었는지 모를 수 있기 때문.

const clickedClass = "clicked" 이렇게 적어야 된다.

변수를 생성하면 이 변수를 적는데 실수한다면, 자바스크립트는

이 변수가 정의되어 있지 않다고 알려줄 것이기 때문에 오류를 줄일 수 있음.

그냥 내가 바로 적으면 자바스크립트가 알 수 없음.

- 동일!

```
<div id="login-form">
  <input type="text" placeholder="put in your name">
  <button> Login </button>
</div>
```

```
const loginForm = document.getElementById("login-form");
const loginInput = loginForm.querySelector("input");
const loginButton = loginForm.querySelector("button");
```

```
loginInput = document.querySelector("#login-form input");
loginButton = document.querySelector("#login-form button");
```

- input의 유효성 검사 (원래는 onclickhandler 함수에서 if else로 해줬어야 하던 것을 form으로 편하게 하는 것)

```
<input
  required
  maxLength="15"
  type = "text"
/>
```

하려면 input이 form 안에 있어야함

근데 form에서 input 있을 때 엔터나 버튼 누르면 submit 됨

그러면 웹사이트 전체가 새로고침이 됨

그렇다면?

클릭이벤트 처리가 아닌 submit 이벤트를 처리해줘야

form submit은 브라우저에서 기본값으로 새로고침이 되게끔 되어있음

이벤트리스너에선 무조건 함수의 이름만 적어야, ()까지 적으면 바로 실행되버림

◦ ㅎㅎ

```
<script src="app.js"></script>
```

html body 내에서 이걸 맨 아래에 적자

- local storage

유저가 한번 입력 or 설정한 것을 기억하려는 것

key value 쌍으로 개발자 도구 application에 저장되어있음

setItem으로 저장

getItem으로 불러옴

removeItem

- setInterval(); //실행할 함수, 몇 ms 간격으로 call할지 arg

- "1".padStart(2,"0")

"01"

- todo에서의 의미있는 코드들

```
const todoInput = todoForm.querySelector("input")
                  = document.querySelector("#todo-form input")

const li = document.createElement("li") //만들고 변수 li에 저장하겠단 마인드
li.appendChild(span); //html에서 자기 하위(들여쓰기)로 놓겠다

console.dir(event.target.parentNode.innerText)
//target은 클릭된 html ele (버튼)
//클릭된 ele의 부모 (li)

todoForm.addEventListener("submit", todoHandler); //todo에 입력 후 엔터 시 (엔터=submit 됨)
```

- 로컬 storage엔 텍스트만 저장됨

하지만 어레이로 저장하고싶다

js obj나 array나 어떤 것이든 string으로 바꿔주는 기능

JSON.stringify(~)

반대로!

JSON.parse(string) = array나 obj로

- console.log(todoForm.input); 으로는 안되네

- 같은거 커서 잡기

컨 슈!

- forEach는 어레이 각 엘리먼트마다 뭔가를 해줄수 있게하는 함수

parsedTodos.forEach(sayHello)

어레이.forEach(item을 파라미터로 받는 함수)

item을 통해서 어레이의 각 엘리먼트의 정보를 받을 수 있음

```
function sayHello(item) {  
  console.log("this", item)  
}
```

```
parsedTodos.forEach(sayHello)
```

```
parsedTodos.forEach((item)=>console.log("this", item))
```

둘은 동일

- date.now()

랜덤 id 할당 시 좋음

- 어레이[1,2,3,4].filter(f1)

새 array에서 기존 array 엘리먼트를 유지하고 싶으면 f1은 반드시 true 리턴해야, 제외하고 싶으면 false 리턴

js는 f1을 4번 호출할것임

```

> function sexyFilter(item){return item !== 3 }
< undefined
> [1, 2, 3, 4, 5].filter(sexyFilter)
< ▼ (4) [1, 2, 4, 5] ⓘ

```

```

> const todos = [{text:"lalala"}, {text:"lololo"}]
< undefined
> function sexyFilter(todo){ return todo.text !== "lololo"}
< undefined
> todos.filter(sexyFilter)
< ▼ [{...}] ⓘ
  ▶ 0: {text: "lalala"}

```

filter는 기존 array를 변경하지 않음!

- console.log(typeof li.id)
타입을 알아볼때
- navigator.geolocation.getCurrentPosition()
브라우저에서 위치좌표 줘서 gps, wifi, 위치 등 모두 가능하게 함
- fetch는 promise, 당장 뭔가 일어나는 것이 아닌 시간이 걸리 뒤에 일어나는 것
서버의 응답을 기다리는 것임

```

fetch(url)
  .then((response)=> response.json())
  .then((data)=> {
    console.log(data.name, data.weather[0].main)
  })

```