

먼저, 반복문에 대해 알아보자.

이미 while, for 를 알고 있는 학생들도 있을 것이다. 복습해보는 시간 가지며, map() 과 filter() 로 반복하는법도 알아본다.

while 과 for 는 크게 보면 조건문의 일부이다. 소괄호 () 안에 반복의 조건이 들어간다.

while 과 for 를 사용하는 반복을, 영어로는 loop 라고 한다.

이 조건에 일치하면 반복하고, 일치하지 않으면 반복을 탈출한다.

```
const menus = ["짜장면", "짬뽕", "탕수육"];
let i = 0;
while (menus[i]) {
  console.log(menus[i]);
  i++;
}
```

먼저 while loop 이다.

여기서, let 을 사용했다. i 가 변해야하기 때문이다. const 로 i 를 쓰면 반복이 불가능하다.

JavaScript 개발자들간의 권장사항으로, let 은 반복문에서만 사용한다.

```
const menus = ["짜장면", "짬뽕", "탕수육"];
for (let i = 0; i < menus.length; i++) {
  console.log(menus[i]);
}
```

다음은 for loop 이다.

소괄호 안에 역시 조건인데, 시작조건, 반복조건, 종료조건이 차례로 들어간다.

for (처음설정값 ; 종료조건 ; 반복 후 작동할 식)

여기선, menus 의 길이를 반복의 조건으로 정하고 i 를 0에서 증가시켜 길이에

다다르면 반복을 종료한다.

하지만 JavaScript 개발자가 알아야하는 두 가지의 반복이 더 있는데 바로 `map()` 과 `filter()` 이다.

이것을 알기 전에, 먼저 콜백함수(callback function) 의 개념에 대해 알아야 한다.

요즘 언어들은 업데이트가 이루어지면서, 콜백함수를 쓸 수 있도록 지원하는 단계이지만, JavaScript 는 애초에 callback 함수를 자유자재로 쓸 수 있는 유일한 메이저 언어였다.

콜백함수는 함수 안에 '들어와서' call 되는 함수이다. 즉, 함수 안에 함수다.

어떻게 들어가나? 파라미터로 함수를 넘긴다.

이 경우, 안에 있는 함수의 작동이 끝나야만 바깥에 있는 함수가 처리된다.

어디서 비슷한 걸 들어본것 같지 않나? 그렇다! 바로 재귀(recursion) 의 JavaScript 버전이다.

콜백은 어디서 쓰일까? 가장 많이 쓰이는건 서버와 비동기통신을 할 때이다. 이것의 사용법은 실전강의를 준비할테니 그 때 경험해보자.

그리고 우리가 지금 써볼 `map()`, `filter()` 와 그 원형인 `reduce()` 에서도 콜백은 쓰인다.

흔히 쓰이는 콜백의 다른 예는 `setTimeout()` 함수이다. 실행을 의도적으로 딜레이시키기 위해 사용한다. 다음 코드를 보자.

```
const sampleFunction = () => {  
  console.log("짜잔");  
};  
  
setTimeout(sampleFunction, 2000);
```

여기서, `setTimeout` 에 들어가는 파라미터는 총 두개인데, 그 중에 하나가 무엇인가? 그렇다. 함수다! JavaScript에선 함수도 `const` 선언이 가능하다고 했는데, 이것은 함수가 변수임을 의미한다.

첫번째 파라미터를 2000 밀리초, 즉 2초 뒤에 실행시킨다는 의미다.

주의할 것은, `sampleFunction` 함수를 파라미터로 넣을 때 소괄호 () 를 쓰지

않았다는 것이다. 소괄호 () 를 쓸 경우 함수를 즉시 실행시킨다는 의미이다. 우린 함수 자체만을 넘겨줄 것이기 때문에 소괄호를 쓰지 않는다.

심지어, 이런 것도 가능하다. 방금 코드를 한 줄로 줄여보자.

```
setTimeout(sampleFunction = () => console.log("짜잔"), 2000);
```

즉, 함수를 바깥으로 빼지 않고 안에서 함수 정의까지 해버린 경우다. 이 전체가 첫번째 파라미터가 된다. 이 때는 앞에 const 를 붙이지 않는다는것에 주목하자.

더 줄일수도 있다. 아예 이름을 떼버리면 다음과 같다.

```
setTimeout(() => console.log("짜잔"), 2000);
```

이름이 없다. 이런 형태를 람다함수(Lambda function)라고 한다.

JavaScript에선 함수를 따로 선언하지 않고 “한번 쓰고 다시는 안 쓸” 일회용 함수를 만들수도 있다.

함수의 정의를 다시 생각해보면,

재사용(reuse) 가능한 기능(function) 이다.

그런데, 이 경우 재사용은 불가능하다.

여러분은 이런 식으로 절대 안 쓸거 같아도, 정말정말 흔하게 쓰는 기법이다.

람다함수는 화살표 함수 이전엔 만드는 것 자체가 불가능했다. function 키워드 다음엔 무조건 함수 이름이 나와야하기 때문이다.

그러나, 화살표함수를 이용하면 이름 없는 람다함수도 만들 수 있다.

이제 map() 에 대해 알 준비가 됐다.

map() 은 배열(array)에서만 사용 가능하다.

콜백을 이용해 평소 반복을 쓸 때 의도한 결과를 낸다.

정확하게는, 배열을 받아 요소, 즉 엘리먼트의 갯수가 같은 새 배열을 만든다.

```
const menus = ["짜장면", "짬뽕", "탕수육"];
```

```
const newMenus = menus.map(menu => `${menu} 좋아!`);
```

```
// ["짜장면 좋아!", "짬뽕 좋아!", "탕수육 좋아!"]  
console.log(newMenus);
```

menus ["짜장면", "짬뽕", "탕수육"]



newMenus ["짜장면 좋아!", "짬뽕 좋아!", "탕수육 좋아!"]

배열이름.map() 으로 쓰는데, 파라미터는 하나다.

근데 그 파라미터가 함수다.

함수가 파라미터로 들어간다? 그렇다. 콜백이다.

menus 는 ["짜장면", "짬뽕", "탕수육"] 총 세 개의 엘리먼트로 이루어져 있다.

여러개라서 복수형으로 menus 라고 썼다.

안에 들어간 콜백을 자세히 보면, menu 라고 단수로 시작한다.

이것은 하나하나의 요소를 의미한다.

짜장면

짬뽕

탕수육

그리고 이 하나하나를 사용해서 리턴값을 만드는데, 이것이 모여서 새 배열이 된다.

즉, map() 은 배열을 받아서 새 배열을 만드는 함수다.

짜장면, 짬뽕, 탕수육은 각각

짜장면 좋아!

짬뽕 좋아!

탕수육 좋아!

가 되며, 이것들이 모여서 새 배열,

["짜장면 좋아!", "짬뽕 좋아!", "탕수육 좋아!"]

가 되는 것이다.

map은 새 배열을 리턴하고, 원래의 배열은 그대로 보존한다.

그러나, 상황에 따라 리턴값이 필요없을수도 있다. 내가 단순히 세 개의 콘솔로그를 찍고싶다고 가정하자.

for 문으로 쓰면 다음과 같다.

```
const menus = ["짜장면", "짬뽕", "탕수육"];
for (let i = 0; i < menus.length; i++) {
  console.log(menus[i]);
}
```

결과는 당연히

짜장면

짬뽕

탕수육

이다. 이것을 map 을 사용할 시, 다음과 같이 간단히 할 수 있다.

```
const menus = ["짜장면", "짬뽕", "탕수육"];

menus.map(menu => console.log(menu));
```

다음은 filter() 다. 이것은, 배열을 받아서 조건에 맞는 요소만 모아 새 배열로 만드는 함수다. 역시 콜백을 사용한다.

```
const students = ["조교행님", "지연이", "혁주", "민지", "은혁"];

const newStudents = students.filter(student => student.length > 2);

console.log(newStudents); // ["조교행님", "지연이"]
```

students ["조교행님", "지연이", "혁주", "민지", "은혁"]



filter length > 2

newStudents ["조교행님", "지연이"]

파라미터는 map 과 마찬가지로 하나의 함수다. 즉, 콜백이다.

아까랑 비슷하지만 다른 점은, 리턴에 조건이 들어가고 이 조건에 알맞은 것만 새 배열로 만든다는 것이다.

필터는 걸러낸다는 뜻이다. string 의 길이가 2 초과인 것만 걸러내므로, ["조교행님", "지연이"] 만 새 배열이 된다.

map(), filter() vs loop (for, while)

성능은 별 차이 없다.

map() 과 filter() 가 더 좋은 이유는, 시작조건, 종료조건, 증감식 등을 생각할 필요가 없기 때문이며, 원래 데이터의 불변성(immutability) 을 보장하기 때문이다. 우리는 map 과 filter 를 쓰면서 i 를 한번도 쓰지 않았고, 원래 배열의 값을 바꾼 적 없이 새로운 배열을 만든 것을 기억하자.

map과 filter는 훨씬 더 깔끔한 코드, 즉, 읽기 쉬운 코드를 만든다.

JavaScript 에서 99% 의 반복작업은 서버로부터 받는 데이터를 처리하는 일인데, 서버에서 배열로만 들어온다고 확신할 수 있다면 for 와 while 보다 훨씬 유리하게 작업 가능하다.

그러나, map() 과 filter 는 배열이 존재해야만 사용 가능하다. 당연히 원래 데이터가 배열이 아니라면 사용이 어렵다.

구구단이나 별찍기 코드를 만드는 데 map 이나 filter 를 쓰긴 어려울 것이다.

또한, 조금 있다가 배울 DOM 조작의 경우에도, 배열처럼 보이지만 타입이 배열이 아니라서 map 을 사용하기 힘들다.

이 경우, 기존의 loop (for, while) 을 사용하는 것이 훨씬 편하다.

뭐든지 장단점이 있다. 상황에 알맞게 사용하자.

또한, `map()` 과 `filter()` 에 대해 익숙해진다면 반드시 `reduce()` 를 연습해보는걸 추천한다.

이제, 우리는 콘솔창에서 벗어날 때가 됐다. 우리가 배우는 JavaScript 는 브라우저를 조작하기 위한 언어다.

사실, 요즘 기준에선 틀린 말이다. node.js 탄생 이후, 브라우저 바깥에서도 쓰이기 때문이다.

여러분이 쓰고있는 vscode 에디터는 놀랍게도 JavaScript 로 코딩되었다.

그러나, JavaScript 는 태생이 브라우저를 조작하기 위한 언어다.

정확하게는 event 를 받는 event handler 를 사용해 DOM 조작을 하기 위한 언어다.

모르는 개념부터 알고 가자.

event 는 브라우저에서 일어나는 각종 “사건” 이다.

클릭, 데이터 변화, 마우스 올리기, 키 입력, 스크롤 등등... 사용자가 무수히 일으키는 “사건” 들이 있다.

이런 event 를 다루려면 이벤트 핸들러(event handler) 라는 것을 사용하는데,

이벤트 핸들러는 on 으로 시작하며, 종류는 엄청나게 많다.

사용할 때는 html 태그 안에서 attribute 로 사용하면 된다.

정말 많은 이벤트 핸들러가 있지만, 우린 자주 쓰는 두 가지만 다뤄볼 것이다.

onclick 마우스 클릭

onchange 데이터 변화

일단, 빈 html 에 기본 태그들을 ! + tap 으로 입력하고, 평소 하던 대로 타이틀 바꿔주고 h1 하나 달아준다.

그리고 우린 두 가지를 추가해줄 것인데, 다음과 같다.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta
    name="viewport"
    content="width=device-width, initial-scale=1.0">
  <script src="./index.js"></script>
  <title>DOM 조작 연습</title>
</head>
<body>
  <h1>DOM 조작 연습</h1>
  <button onclick="test()">클릭</button>
</body>
</html>

```

자세히 보면, head 안에 script 라는 태그가 있고, src 에 .js 라는 파일이 연결되어 있다.

즉, css 처럼 JavaScript 도, 단 한줄을 쓰더라도 파일로 분리해 import 한다.

그리고 버튼을 하나 달아두는데, 위와 똑같이 일단 입력한다.

onclick="test()"

on으로 시작했다. 즉, 이벤트 핸들러, 그 중에서도 클릭을 받는 onclick 이벤트 핸들러이다.

누군가가 우리의 버튼을 클릭하면, test() 함수가 실행되는 것이다.

함수는 소괄호를 꼭 써주고, 쌍따옴표로 감싼다.

즉, 이벤트 핸들러는 사용하고 싶은 태그 안에 Attribute 로 쓴다.

* 클릭 하면 대표적으로 버튼이라서 그렇지, 그 어떤 태그에도 다 쓸수 있다.

h1, div, span 등등, 이벤트 핸들러는 모든 태그에서 사용 가능하다.

그리고, index.js 파일에 test() 함수를 작성한다.


```
const test = () => {  
  console.log("클릭함!");  
};
```

이렇게 쓸 경우, 사용자가 버튼을 클릭하면 우리 콘솔에 “클릭함!”이라는 글자가 찍힐 것이다.

F12 눌러서 확인해보자.

다음은 onchange 이벤트 핸들러다. 즉, 변경이 일어날 때 실행된다.

```
<body>  
  <h1>입력창을 떠나면 콘솔에 찍힌다</h1>  
  <input type="text" onchange="test()">  
</body>
```

```
const test = () => {  
  console.log("change!!!");  
};
```

입력창을 떠나면 콘솔에 찍힌다

6 change!!!

무엇이 변경되었을 때 실행된걸까? input 의 데이터가 변경되었기때문에 실행된 것이다.

그런데 여러분들이 꼭 쓸 때마다 실행되진 않는다. 입력창을 떠나야만 실행된다는 점을 알아두자.

* 이것은 여러분들이 정적 웹을 개발하고 있기 때문이다. 만약 react, vue, svelte

같은 기술을 배우게 되면 변경이 일어나는 즉시 화면이 다시 그려지므로 onchange 가 여러분이 입력할때마다 실행될 것이다.

이런 이벤트들을 가지고 뭘 할수 있을까?

JavaScript 는 이벤트를 받아서, 화면의 내용을 바꿀 수 있다.

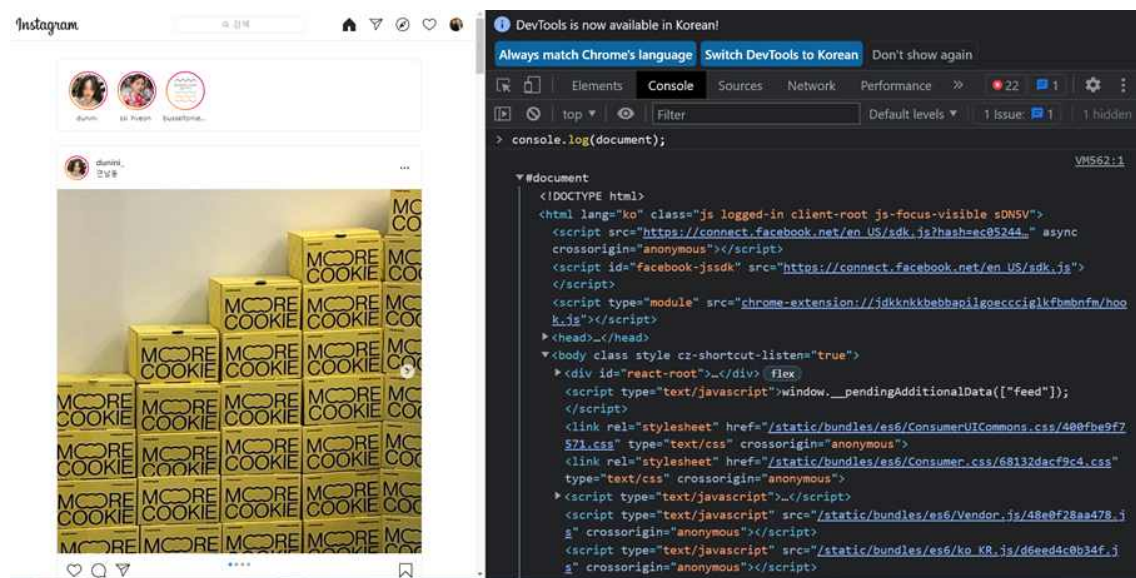
이를 위해, DOM 에 대해 알아야 한다.

DOM (Document Object Model)

문서 객체 모델이다.

우린 하나의 html 파일을 “문서(document)” 라고도 한다 했다.

하나의 문서를 이루는 수많은 요소들을, 하나의 거대한 “객체(object)” 로 만든 것이 DOM 이다.



인스타그램에서 console.log(document); 를 입력해보자.

태그들이 쭉 나온다. 이 모든 정보들이 여러분이 왼쪽에 보고 있는 사이트들을 이루는 정보다.

하지만, 이런 식으로 보면 객체라는 게 와닿지 않을 것이다.

태그의 형태로 출력되는것을 객체로 보고싶을 땐,

console.log() 대신 console.dir() 을 사용한다.

```

▼ #document ⓘ
  URL: "https://www.instagram.com/"
  ▶ activeElement: body
  ▶ adoptedStyleSheets: [CSSStyleSheet]
  alinkColor: ""
  ▶ all: HTMLAllCollection(1045) [html.js.logged-in.client-root.js-focus-visible.s
  ▶ anchors: HTMLCollection []
  ▶ applets: HTMLCollection []
  baseURI: "https://www.instagram.com/"
  bgColor: ""
  ▶ body: body
    characterSet: "UTF-8"
    charset: "UTF-8"
    childElementCount: 1
  ▶ childNodes: NodeList(2) [<!DOCTYPE html>, html.js.logged-in.client-root.js-fo
  ▶ children: HTMLCollection [html.js.logged-in.client-root.js-focus-visible.sDN5V
    compatMode: "CSS1Compat"
    contentType: "text/html"
    cookie: "mid=YNveIwALAAHUCOUMJVEdEvrqmqzQ-; ig_nrcb=1; fbm_124024574287414=base
    currentScript: null
  ▶ defaultView: Window {window: Window, self: Window, document: document, name: "
    designMode: "off"
    dir: ""
  ▶ doctype: <!DOCTYPE html>
  ▶ documentElement: html.js.logged-in.client-root.js-focus-visible.sDN5V
    documentURI: "https://www.instagram.com/"
    domain: "www.instagram.com"

```

이것은 인스타그램에서 `console.dir(document);` 입력 시 나오는 결과의 “일부분”이다.

즉, 하나의 페이지는 거대한 객체임을 알 수 있다.

그렇다면, 이렇게 생각할 수 있다.

하나의 html 파일이 하나의 거대한 객체라면,

이벤트가 발생했을 때,

바꾸고싶은 부분을 “선택” 해서, 원하는 값으로 바꾸면

html 을 바꿀 수 있지 않을까?

이것을 DOM Manipulation, 한국어로 “돔 조작” 이라고 한다.

돔 조작을 하기 위한 여러가지 옛날 방법들이 있다.

- `document.getElementsByTagName("div")`
 - 문서의 div 태그들을 선택
- `document.getElementById("idName")`
 - 문서의 idName 이라는 아이디(#) 로 지정된 태그 선택
- `document.getElementsByClassName("myClass")`
 - 문서의 myClass 라는 클래스(.) 로 지정된 태그들을 선택

그러나 위 세가지는, 개발자들에게 매우 불편해서 선배 개발자들은 jQuery 라는 기술을 따로 배워서 사용했다. 그러나 여러분은 더이상 그럴 필요가 없다.

다음 두 가지만 알면 모든 걸 할 수 있다.

```
document.querySelector()  
document.querySelectorAll()
```

안에 들어갈 파라미터로 css 선택자를 string 값으로 쓰면 된다.

먼저, 다음과 같은 css 와 html 을 작성하자.

당연히, css 는 하나의 파일로 분리해 link 태그를 사용해서 head 안에 import 한다.

```
.square {  
    background-color: aqua;  
    height: 200px;  
    width: 200px;  
    margin-bottom: 10px;  
}
```

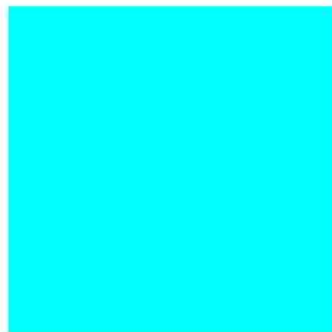
```

<body>
  <h1>DOM 조작 연습</h1>
  <div class="square"></div>
  <button onclick="changeSquareColor()">클릭</button>
</body>

```

우리가 하고 싶은 건, 다음과 같은 것이다.

DOM 조작 연습



클릭



DOM 조작 연습



클릭

클릭을 했을 때, 상자의 색깔이 바뀌도록 하고 싶다.
그렇다면, changeSquareColor 는 어떻게 작성해야 할까?

이 때가 바로 querySelector 를 써야 할 순간이다.

```

const changeSquareColor = () => {
  document.querySelector(".square").style.backgroundColor = "red";
};

```

querySelector() 안에 쓰고자하는 “css 선택자” 를 써준다.
태그면 앞에 아무것도 안 붙이고 태그 이름만,
클래스라면 점 붙이고 쓰고
아이디라면 #을 붙이고 쓴다.

그럼 저건 무슨 뜻인가? square 클래스의 DOM 정보를 가져온다는 뜻이다.
우리가 원하는 건 백그라운드 컬러를 바꾸는 일이므로
.style 로 들어가서, 쓰고자 하는 프로퍼티를 낙타체로 써준다.

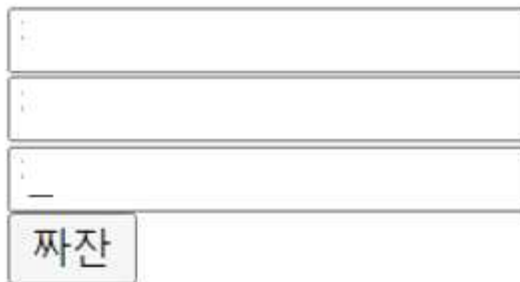
예를 들면, background-color 는 backgroundColor 이런 식이다.
이건 일일이 외울 필요는 없는데, querySelector 로 받은 정보를 console.dir 로
찍어보고 접근해도 되기 때문이다.
그리고, 값을 red 로 썼다.
그러면 이제 클릭했을 때 값이 바뀔 것이다.

querySelectorAll() 을 배워보자.
우선, css 는 작성하지 않고 다음 html 만 작성해보자.

```
<body>
  <h1>DOM조작 연습</h1>
  <div>
    <input type="text" class="test">
  </div>
  <div>
    <input type="text" class="test">
  </div>
  <div>
    <input type="text" class="test">
  </div>
  <button onclick="boom()">짜잔</button>
</body>
```

이것은 다음 화면을 만들 것이다.

DOM 조작 연습



The image shows a simple web form. It consists of three stacked input fields, each with a small 'x' icon in the top-left corner. Below the input fields is a button with the Korean text '짜잔' (Ja-ja-ja).

잘 보면 button 에 onclick 이벤트핸들러가 달려있고, 버튼을 클릭하면 boom() 이라는 함수를 실행시키도록 되어있다.

boom 은 js 파일에 다음과 같이 작성한다. 물론 script 태그를 통해 import 해야한다.

```
const boom = () => {  
  const tests = document.querySelectorAll(".test");  
  for(let i = 0 ; i < tests.length ; i++ ) {  
    tests[i].value = "짹";  
  }  
};
```

querySelectorAll 은 test 클래스를 전부 받아서 배열로 만든다.

test 라는 이름을 가진 클래스는 현재 총 3개니깐 3개로 이루어진 배열이 만들어질 것이다.

그러나, 사실 배열이 아니라 NodeList 라는 타입이라서, map 함수는 사용 불가능하다는것만 알아두자.

document.querySelector 또는 querySelectorAll 의 리턴값을 받으면, 번거롭게 계속 document 객체를 쓸 필요 없이 돔 조작이 가능하다. 여기서 tests 라는, 복수 s를 붙인 이름을 지어준 변수로 한꺼번에 받아두었다가 쓸 것이다.

그 다음 오는 건 반복문이다. tests 의 길이만큼 반복할것이다. 그리고 tests[i] 의

value 를 “짠” 으로 수정할 것이다.

그럼, 여러분들이 인풋 창에 무엇을 입력했든지, 클릭을 하면 다음 화면으로 바뀔 것이다.

DOM조작 연습

짠
짠
짠
짜잔

DOM 조작을 배웠는데, 말해두고싶은게 있다.

첫번째 예제에선,

`querySelector("~").style.backgroundColor` 를,

두번째 예제에선,

`querySelectorAll("~~")` 로 받은 다음 value 를 변경했다.

이런걸 일일이 외워야할까? 아니다!

구글 검색을 적극 활용해 자신에게 맞는 DOM 조작을 익히는 게 중요하다.

사실 본 조교는 시간만 허락한다면 무수히 많은 예제를 다뤄보면서 DOM 조작을 경험시키고 싶지만, 아무리 많이 가르쳐도 모든 상황을 다 가르칠 순 없다.

여러분들이 끊임없이 연습해보고, 여러분이 원하는 상황에서 어떻게 사용하면 될지를 직접 찾아보는 연습도 중요하다.

* 하나 주의할 점은, React.js, Vue.js 를 사용할 시엔 document 객체를 직접 건드려서 DOM 조작을 하면 안 된다는 것이다. 왜냐면, 이런 프레임워크들의 경우엔 virtual DOM 이라는, 가상의 DOM 을 사용하는데, 직접 돔 조작을 했을 때는 여러분의 페이지에 치명적인 영향을 끼칠 것이기 때문이다. React, Vue 에서 권장하는 방법이 있으니 그 방법을 따르도록 하자.

여기까지가 JavaScript 의 기본이다.

남은 시험공부 열심히 하고, 다음 시간까지 여러분들이 해야할 게 있다.

자료실에 다음 두 개의 강의를 올려두었다.

<php 수업 듣기 전에 필요한 MySQL>

<보너스. Bootstrap 사용>

이 두 가지 강의를 이러닝에 업로드했으니 11월 첫 수업 전까지 공부해 오길 바란다.
특히, Bootstrap 은 안 보더라도 <php 수업 듣기 전에 필요한 MySQL> 이 수업은
반드시 듣길 바란다.

이걸 듣지 않으면 진행이 불가능하다.

그리고, 제발 팀 프로젝트 시작하라.

지금까지 배운것만 활용해도 엄청나게 많은 것들을 만들 수 있다.