

반복

- while, for 를 복습
- map(), filter() 배우기

while, for

- 영어로 loop 라고 한다.
- map 과 filter 를 통한 반복은 loop 라고 하지 않는다.

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

while (condition)

조건이 true 일때만 실행됨

i 가 변해야하기때문에 let 을 쓴 것.

* 개발자 권장사항으로, let은 반복문에서만 사용

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

for (처음설정값 ; 종료조건 ; 반복 후 작동할 식)

map(), filter() 를 알기 전에...

- 콜백함수 (callback function)
 - 함수 안에 '들어와서' call 되는 함수. 즉, 함수 안에 함수.
 - 안에 있는 함수의 작동이 끝나야만 바깥에 있는 함수가 처리된다.
 - 어디서 비슷한 걸 들어본거같은데?
 - 재귀(recursion) 의 JavaScript 버전이다.
- 어디서 쓰일까?
 - 비동기(asynchronous) 작업을 할 때 주로 쓰임 (Promise)
 - map(), filter(), reduce()

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

```
setTimeout(sampleFunction, 2000);
```

setTimeout 은 의도적으로 작동을 딜레이시키기위해 쓰인다.
뒤에 붙는 건 밀리초. 즉, 2초 뒤를 의미함.
2초 뒤에 sampleFunction 함수가 실행된다.

JavaScript에선 함수가 변수이기 때문에, **파라미터로 당연히 쓰일 수 있다.**
다른 언어들과는 구별되는 JavaScript 의 독특한 기능

여기서, sampleFunction 에 소괄호를 안쓴것에 주의하라!
왜냐면, 함수 호출할 때 () 는 함수를 실행시킨다는 의미이기 때문.

심지어, 이것도 가능하다.

- 방금 코드를 한 줄로 줄여보자.

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

이 전체가 setTimeout 의 첫번째 파라미터

- 더 줄일수도 있다. 아예 이름을 떼버리자.

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

이렇게, 한번 쓰고 다시는 안 쓸 함수, 즉 재사용(reuse) 이 굳이 필요하지 않는 함수는 이름 없는 무명함수(Lambda function) 로 쓸 수 있다.

여러분은 절대 이런거 안 쓸거같지만, 정말정말 흔하게 쓰는 기법이다.

화살표함수를 쓰지 않으면 lambda function 만드는것은 불가능

map()

- 배열(array)에서만 사용 가능
- 콜백을 이용해, 평소 반복을 쓸 때 의도한 결과를 낸다.
- 정확하게는, 배열을 받아 요소의 개수가 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);
```

map() 은 배열에서 사용 가능한 내장함수다.
파라미터는 하나다.
함수 전체가 파라미터로 들어간다. 즉, 콜백이다.
menus (복수)의 각각의 요소는 menu (단수)다.
리턴값을 모아, 새로운 배열을 만든다
원래의 menus 는 보존한다.

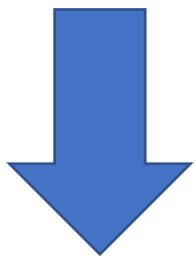
menus ["짜장면", "짬뽕", "탕수육"]



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

map 은 새 배열을 리턴하지만, 상황에 따라 필요하지 않을 수도 있다.

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



```
const menus = ["짜장면", "짬뽕", "탕수육"];  
  
menus.map(menu => console.log(menu));
```

결과는 동일하다.

짜장면
짬뽕
탕수육

filter()

- 배열을 받아, 조건에 맞는 요소만 모아 새 배열로 만든다.

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

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

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

filter() 역시 배열에서 사용 가능한 내장함수다.

파라미터는 하나다.

함수 전체가 파라미터로 들어간다. 즉, 콜백이다.

menus (복수)의 각각의 요소는 menu (단수)다.

밑줄 친 콜백함수의 리턴값은 조건이다.

조건에 맞는것만 새로운 배열로 만든다

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



filter length > 2

newStudents ["조교행님", "지연이"]

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

- 성능은 별 차이없다.
- map(), filter() 가 더 좋은 이유
 - 시작조건, 종료조건, 증감식등을 생각하지 않아도 됨
 - 원래 데이터의 불변성(immutability) 보장
 - 반복을 사용하기 위한 훨씬 더 깔끔한 코드 -> 읽기 쉬움
 - JavaScript 에서 99% 의 반복작업은, 서버로부터 받는 데이터 처리
 - 즉, 서버로부터 데이터를 배열로만 받으면 for와 while 보다 훨씬 유리하게 작업
- 그러나, map() 과 filter() 는 배열이 존재해야만 사용가능
 - 배열 이외의 연산에선 기존 반복문이 낫다. (구구단, 별찍기 등등)
 - 조금 있다가 배울 querySelectorAll() 에서도 map을 쓰기 힘들다.
- reduce() 도 배워보길 추천

event 와 DOM 조작

JavaScript 는 브라우저를 "조작"하기 위한 언어

- 요즘 기준에선 틀린말이다. node.js 탄생 이후 브라우저 바깥에서도 쓰이기 때문 ex) vscode
- 그러나, JavaScript 는 브라우저 조작을 위해 태어난 언어다.

event 와 event handler

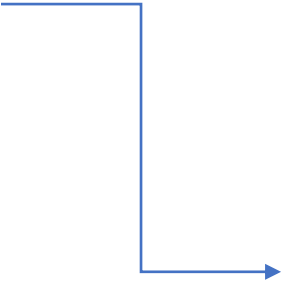
- 브라우저에서 일어나는 "사건" 을 말한다.
 - 클릭, 데이터 변화, 마우스 올리기, 키 입력, 스크롤 등등...
 - 이벤트를 다루려면 이벤트 핸들러를 사용한다.
 - on 으로 시작하며, 종류는 엄청나게 많다.
 - html 태그 안에서 attribute 로 사용하면 된다.
 - 우린 자주 쓰이는 것 두가지만 다뤄보자.
 - onclick 마우스 클릭
 - onchange 데이터 변화


```
<!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>
```

태그 안에 Attribute 로 쓴다.

onclick="클릭하면실행할함수()"

클릭 하면 대표적인게 버튼이니깐, button 태그를 쓴 것일 뿐이다.
즉, h1 이든 div 든 span 이든, 다른 태그에서도 onclick 사용 가능



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

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

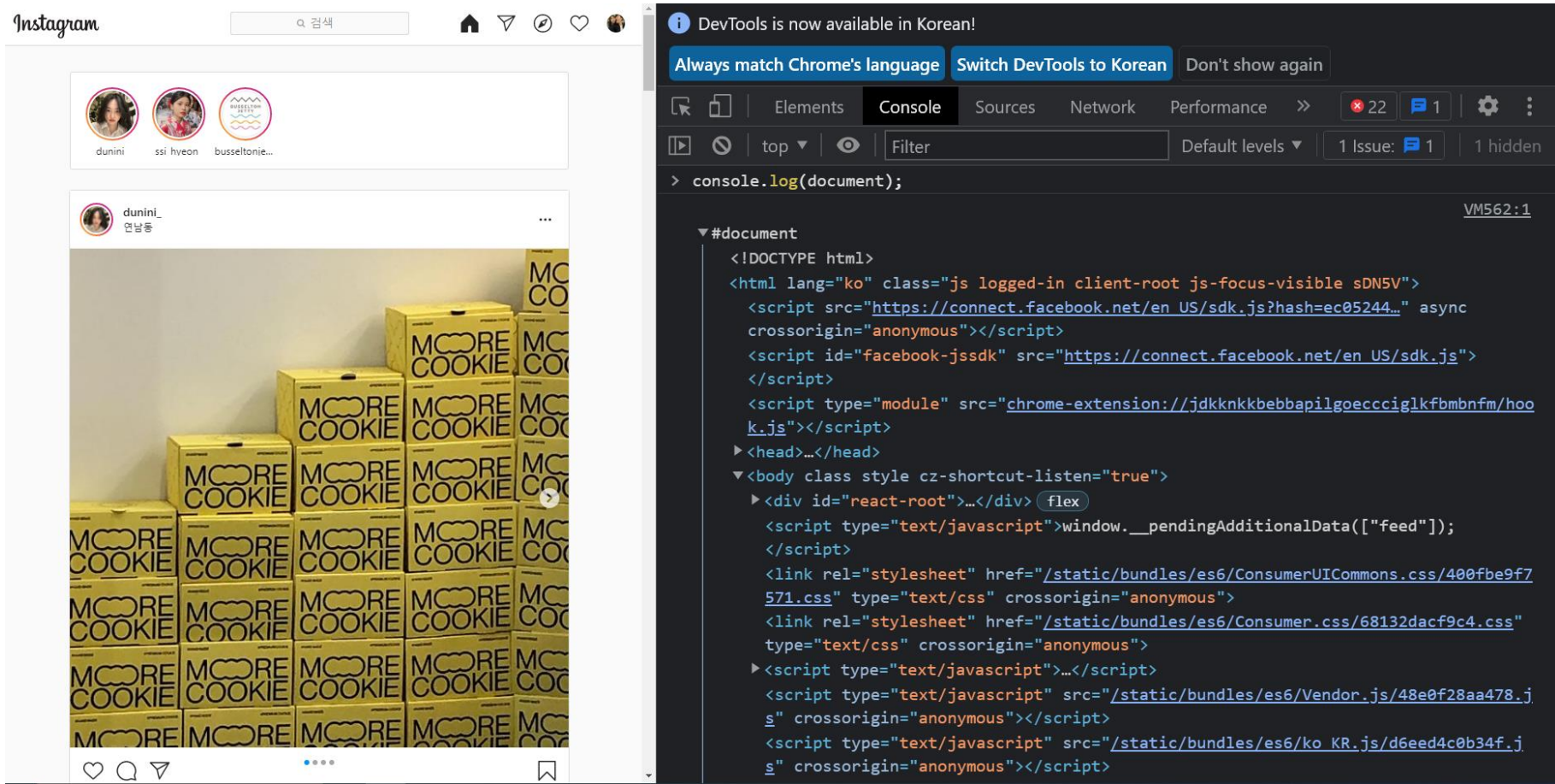
```
const test = () => {
  console.log("change!!!");
};
```

입력창을 떠나면 콘솔에 찍힌다

6 change!!!

DOM (Document Object Model)

- 하나의 html 파일은 "문서(document)" 라고도 한다.
- 하나의 html 파일 전체를, 하나의 "객체" 로 만든 것!



Instagram 에서,
console.log(document);
를 입력했을때의 결과.
하지만, 이런식으로 보면 객체라는게 와닿지 않는다.
이런식으로, 태그의 형태로 나오는것을 객체로 보고싶을땐,
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-foc
  ▶ 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"

```

인스타그램에서

`console.dir(document);`

입력 시 나오는 결과의 "일부"

즉, 하나의 페이지는 거대한 객체다.

그렇다면, 이렇게 생각할 수 있다.

- 하나의 html 파일이 하나의 거대한 객체라면,
 - 이벤트가 발생했을 때,
 - 바꾸고 싶은 부분을 "선택" 해서, 원하는 값으로 바꾸면,
 - html 을 바꿀 수 있지 않을까?
-
- 그걸 DOM Manipulation, 한국어로 "돔 조작"이라고 한다.

여러가지 "옛" 방법

- `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
- 위 세가지는, 개발자들에게 불편해서,
선배 개발자들은 jQuery 라는 기술을 따로 배워서 사용했다.

다음 두 가지만 알면, 모든 걸 할 수 있다

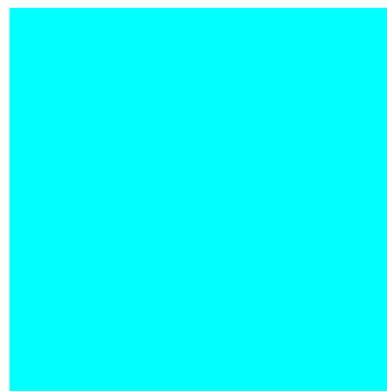
- `document.querySelector()`
- `document.querySelectorAll()`
- 파라미터로, css 선택자를 쓰면 된다.


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

?

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

DOM 조작 연습



클릭



DOM 조작 연습



클릭

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

쓰고자 하는 선택자.

여기선, square 라는 클래스를 선택하므로,

".square" 라고 써준 것.

css를 변경하고 싶으면, style 로 들어가서,

쓰고자 하는 프로퍼티를 낙타체(camelCase)로 쓴다

background-color 는 backgroundColor

일일이 외울 필요는 없다. console.dir 로 찍어보고 객체 접근해도 된다.

```

<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>

```

DOM조작 연습

| |
|----|
| 짬 |
| 짬 |
| 짬 |
| 짜잔 |

document 를 변수로 따로 받아도 조작 가능
여기선 tests 라는 변수로 받았다.

클래스 각각을 받아 배열로 만든다.

그러나, 일반적인 배열은 아니고
NodeList 객체라서, map 함수 사용 불가

```

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

```

주의

- 첫번째 예제에선, `.style.backgroundColor` 를,
 - 두번째 예제에선, `.value` 를 변경했다.
 - 이런걸 일일이 외워야할까? 아니다!
 - 구글 검색을 적극 활용해 자신에게 맞는 DOM 조작을 익히자
-
- React.js , Vue.js 사용 시 `document` 객체를 직접 건드리면 안됨!

여기까지, JavaScript 기본

- <php 수업 듣기 전에 필요한 MySQL>
- <보너스. Bootstrap 사용>
- 이 두가지 강의를 이러닝에 업로드했으니,
- 11월 첫 수업 전까지 공부해 올 것
- 그리고, 제발 팀 프로젝트 시작하라.
- 지금까지 배운것만 활용해도 엄청나게 많은 것을 만들 수 있다