

Hello,

KDT 웹 개발자 양성 프로젝트

5기!

with





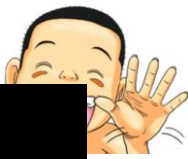
배열

Array



length : 배열의 길이

students.length // 30



push() : 배열 끝에 추가

```
let days = ['월', '화', '수'];
```

```
days.push('목')
```

```
console.log(days) // ['월', '화', '수', '목']
```



pop() : 배열 끝 요소 제거

```
let days = ['월', '화', '수'];
```

```
days.pop()
```

```
console.log(days) // ['월', '화']
```



shift, unshift 배열 앞에 제거/추가

추가

```
days.unshift('일');  
console.log(days) // ['일', '월', '화', '수'];
```

제거

```
days.shift();  
console.log(days) // ['월', '화', '수'];
```



반복문 : **for**

```
let days = ['월', '화', '수'];
```

```
for(let index = 0; index < days.length; index++){  
  console.log(days[index])  
}
```



객체!

Object



Object - 접근, 추가, 삭제

```
const superman = {  
  name : 'clark',  
  age : 33,  
}
```

접근

```
superman.name // 'clark'
```

```
superman['age'] // 33
```

추가

```
superman.gender = 'male';
```

```
superman['hairColor'] = 'black';
```

삭제

```
delete superman.hairColor;
```



Object - 프로퍼티 존재 여부 확인

```
const superman = {  
  name : 'clark',  
  age : 33,  
}
```

```
superman.birthDay;  
// undefined
```

```
'birthDay' in superman;  
// false
```

```
'age' in superman;  
// true
```



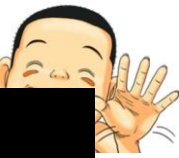
for ... in 반복문

```
for(let key in superman){  
  console.log(key)  
  console.log(superman[key])  
}
```



객체 Object

메소드 Method



Object

method : 객체 프로퍼티로 할당 된 함수

superman.fly():
// 날아갑니다.

```
const superman = {  
  name : 'clark',  
  age : 33,  
  fly : function( ){  
    console.log('날아갑니다.')  
  }  
}
```



this

Object



```
const user = {  
  name : 'Mike',  
  sayHello : function() {  
    console.log(`Hello, I'm ${this.name}`);  
  }  
}
```

```
user.sayHello();
```

```
// Hello, I'm Mike
```



화살표 함수와

this

Object



```
let boy = {  
  name : 'Mike',  
  sayHello : () => {  
    console.log(this); // 전역객체  
  }  
}
```

- 브라우저 환경 : **window**
- Node js : **global**

```
boy.sayHello( );  
this !== boy
```



객체의 불변성

원시 (Primitive) 타입

String

Number

Boolean

Null

Undefined

BigInt

Symbol

객체 (Object) 타입

원시 타입을 제외한 모든 것

Object

Array

...

원시 (Primitive) 타입

```
const location = "korea"
```

location

"korea"

객체 (Object) 타입

```
const location = {  
  country: "korea"  
}
```

location

#12345



#12345

{ country: "korea" }

원시 (Primitive) 타입

```
const locationOne = "korea"
```

```
const locationTwo = "korea"
```

```
locationOne === locationTwo
```

```
> true
```

객체 (Object) 타입

```
const locationOne = {  
  country: "korea"  
}
```

```
const locationTwo = {  
  country: "korea"  
}
```

```
locationOne === locationTwo
```

```
> false
```



생성자

함수



생성자 함수



```
function User(name, age) {  
  this.name = name;  
  this.age = age;  
}
```

```
let user1 = new User('Mike', 30);  
let user2 = new User('Jane', 22);  
let user3 = new User('Tom', 17);
```





클래스



class



Car

objects



Audi



Nissan



Volvo

Class 예제 코드



```
// Class 선언
class Car {
  // 생성자 전달
  constructor(brand, color) {
    this.brand = brand;
    this.color = color;
  } // 메소드 선언

  drive() {
    console.log(`${this.brand}의 ${this.color}색 자동차가 주행 중입니다`);
  }
}

const hyundai = new Car('hyundai', 'blue');
const porsche = new Car('porsche', 'black');
const toyota = new Car('toyota', 'silver');

console.log(hyundai.brand);
porsche.drive();
toyota.drive();
```

```
tetz@DESKTOP-P7Q40LL MINGW64 ~/Desktop/node-setting
$ node js/main.js
hyundai
porsche의 black색 자동차가 주행 중입니다
toyota의 silver색 자동차가 주행 중입니다
```



Strict Mode



상속!



class Car

this.brand

this.color

drive()

상속

class ElecCar

this.brand

this.color

this.fuel

drive()

showFuel()



Over-riding



```
class Car {
  // 생성자 전달
  constructor(brand, color) {
    this.brand = brand;
    this.color = color;
  } // 메소드 선언

  drive() {
    console.log(`${this.brand}의 ${this.color}색 자동차가 주행 중입니다`);
  }
}
```

// 상속 선언

```
class ElecCar extends Car {
  constructor(brand, color, fuel) {
    super(brand, color);
    this.fuel = fuel;
  }

  drive() {
    console.log(
      `${this.brand}의 ${this.color}색 자동차가 ${this.fuel}의 힘으로 주행 중입니다`
    );
  }
}
```

```
const hyundai = new Car('hyundai', 'white');
hyundai.drive();
```

```
const tesla = new ElecCar('tesla', 'red', 'electricity');
tesla.drive();
```

```
tetz@DESKTOP-P7Q40LL MINGW64 ~/Desktop/node-setting
$ node js/main.js
hyundai의 white색 자동차가 주행 중입니다
tesla의 red색 자동차가 electricity의 힘으로 주행 중입니다
```



super



```
class Car {
  // 생성자 전달
  constructor(brand, color) {
    this.brand = brand;
    this.color = color;
  } // 메소드 선언

  showSpec() {
    console.log(
      `이 차의 브랜드는 ${this.brand}이고 색상은 ${this.color} 입니다.`
    );
  }
}

// 상속 선언
class ElecCar extends Car {
  constructor(brand, color, fuel) {
    super(brand, color);
    this.fuel = fuel;
  }

  showSpec() {
    super.showSpec();
    console.log(`그리고 이 차는 ${this.fuel}의 힘으로 움직입니다!`);
  }
}

const hyundai = new Car('hyundai', 'white');
hyundai.showSpec();

const tesla = new ElecCar('tesla', 'red', 'electricity');
tesla.showSpec();
```

```
tetz@DESKTOP-P7Q40LL MINGW64 ~/Desktop/node-setting
$ node js/main.js
이 차의 브랜드는 hyundai이고 색상은 white 입니다.
이 차의 브랜드는 tesla이고 색상은 red 입니다.
그리고 이 차는 electricity의 힘으로 움직입니다!
```



instance of



생성자 함수로 구현!

ES5 버전



```
function Car(brand, color) {  
  this.brand = brand;  
  this.color = color;  
  this.drive = function () {  
    console.log(`${this.brand}의 ${this.color}색 자동차가 주행 중입니다`);  
  };  
}
```

```
function ElecCar(brand, color, fuel) {  
  // Car 생성자 함수의 this 와 생성자를 가져오기 위한 call 메소드 사용  
  Car.call(this, brand, color);  
  this.fuel = fuel;  
  // 오버라이딩 구현  
  this.drive = function () {  
    console.log(  
      `${brand}의 ${color}색 자동차가 ${this.fuel}의 힘으로 주행 중입니다`  
    );  
  };  
}
```

```
// 상속 받을 ElecCar 의 프로토 타입을 Object 객체(최상위 개념)를 사용해서 Car 의 프로토 타입과 동일하게 만들어 낸다.  
ElecCar.prototype = Object.create(Car.prototype);  
// 상속 받을 ElecCar 의 생성자는 위에서 선언한 생성자 함수의 것을 따르도록 설정  
ElecCar.prototype.constructor = ElecCar;
```

```
const tesla = new ElecCar('tesla', 'white', 'electricity');  
tesla.drive();
```



지금 시작합니다



메소드

체이닝



```
// .split: 문자를 인수 기준으로 쪼개서 배열로 반환
let helloTest = "H-e-l-l-o";
let helloTestArr = helloTest.split("-");
console.log(helloTestArr);

// .reverse: 배열의 순서를 뒤집어서 반환
// .join: 배열을 인수 기준으로 문자로 병합해서 반환
let hello = "Hello~";
helloArr = hello.split("");
console.log(helloArr);

let reverseHello = helloArr.reverse();
console.log(reverseHello);

let helloJoin = reverseHello.join("");
console.log(helloJoin);

console.log(hello.split("").reverse().join(""));
```

```
▶ (5) ['H', 'e', 'l', 'l', 'o']
▶ (6) ['H', 'e', 'l', 'l', 'o', '~']
▶ (6) ['~', 'o', 'l', 'l', 'e', 'H']
~olleH
~olleH
```



메소드 체이닝

- 각각의 메소드를 연결해서 사용하는 개념! 단, 사용한 메소드가 반환 (return) 값을 가지고 있는 경우에만 사용이 가능!
- `hello.split("")` → `['H', 'e', 'l', 'l', 'o']` 라는 배열이 반환 됨
- 배열에는 `reverse()` 라는 메소드가 존재
- `hello.split("").reverse()` 는 `['H', 'e', 'l', 'l', 'o'].reverse()` 와 동일한 것!
- `['H', 'e', 'l', 'l', 'o'].reverse()` → `['o', 'l', 'l', 'e', 'H']` 와 동일
- `hello.split("").reverse().join("")` → `['o', 'l', 'l', 'e', 'H'].join("")` 과 동일

메소드 체이닝



- `['H', 'e', 'l', 'l', 'o'].reverse()` → `['o', 'l', 'l', 'e', 'H']` 와 동일
- `hello.split('').reverse().join('')` → `['o', 'l', 'l', 'e', 'H'].join('')` 과 동일

실습!



```
const kdtCurriculum = ["HTML", "CSS", "JS", "BACKEND", "REACT"];
```

- 위의 배열에서 첫번째와 마지막 데이터의 문자열을 메소드 체이닝을 이용하여 반대로 뒤집어 주세요.

LMTH

TCAER

```
const kdtCurriculum = ["HTML", "CSS", "JS", "BACKEND", "REACT"];
const result1 =
const result2 =
console.log(result1);
console.log(result2);
```



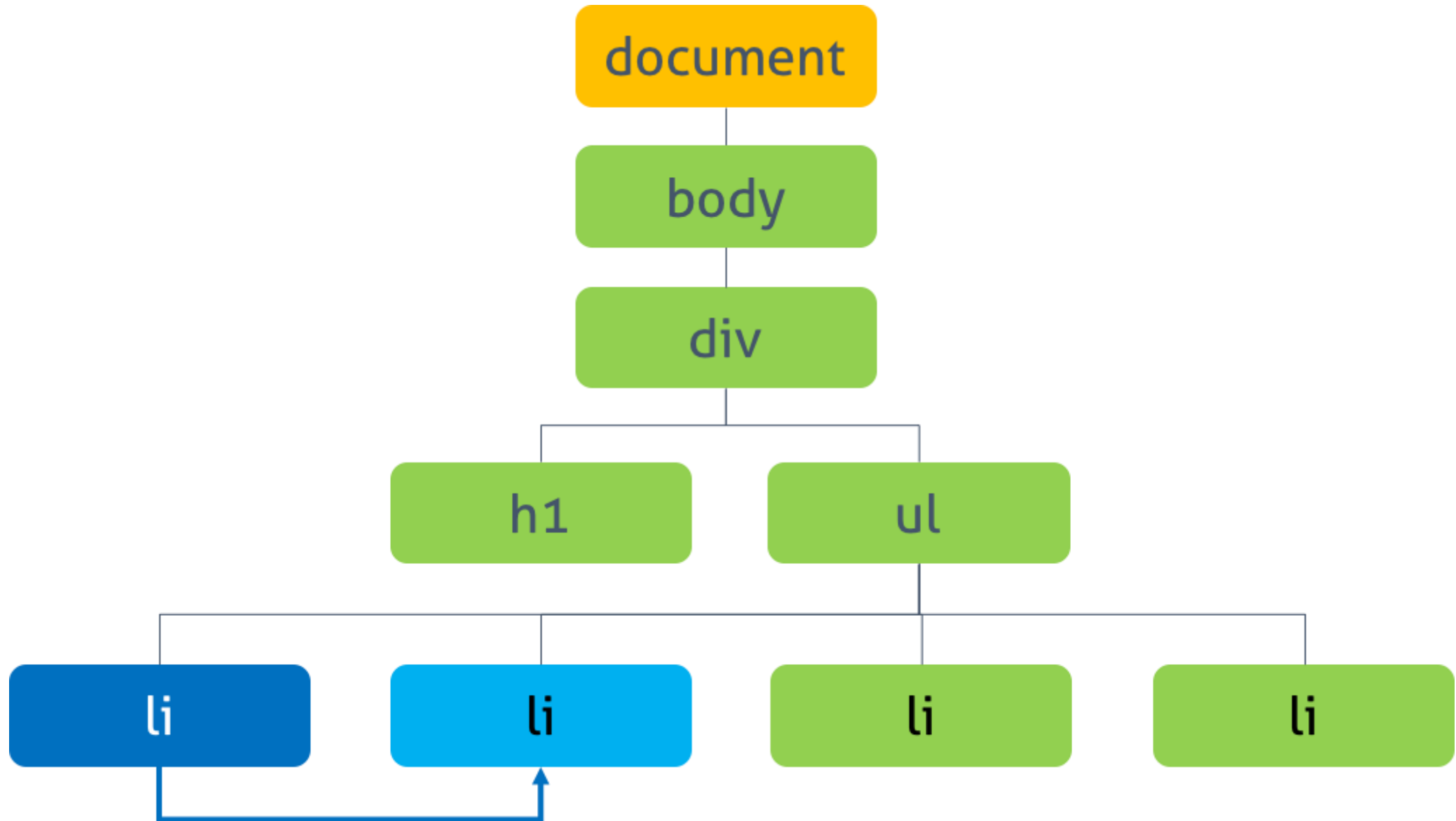
DOM

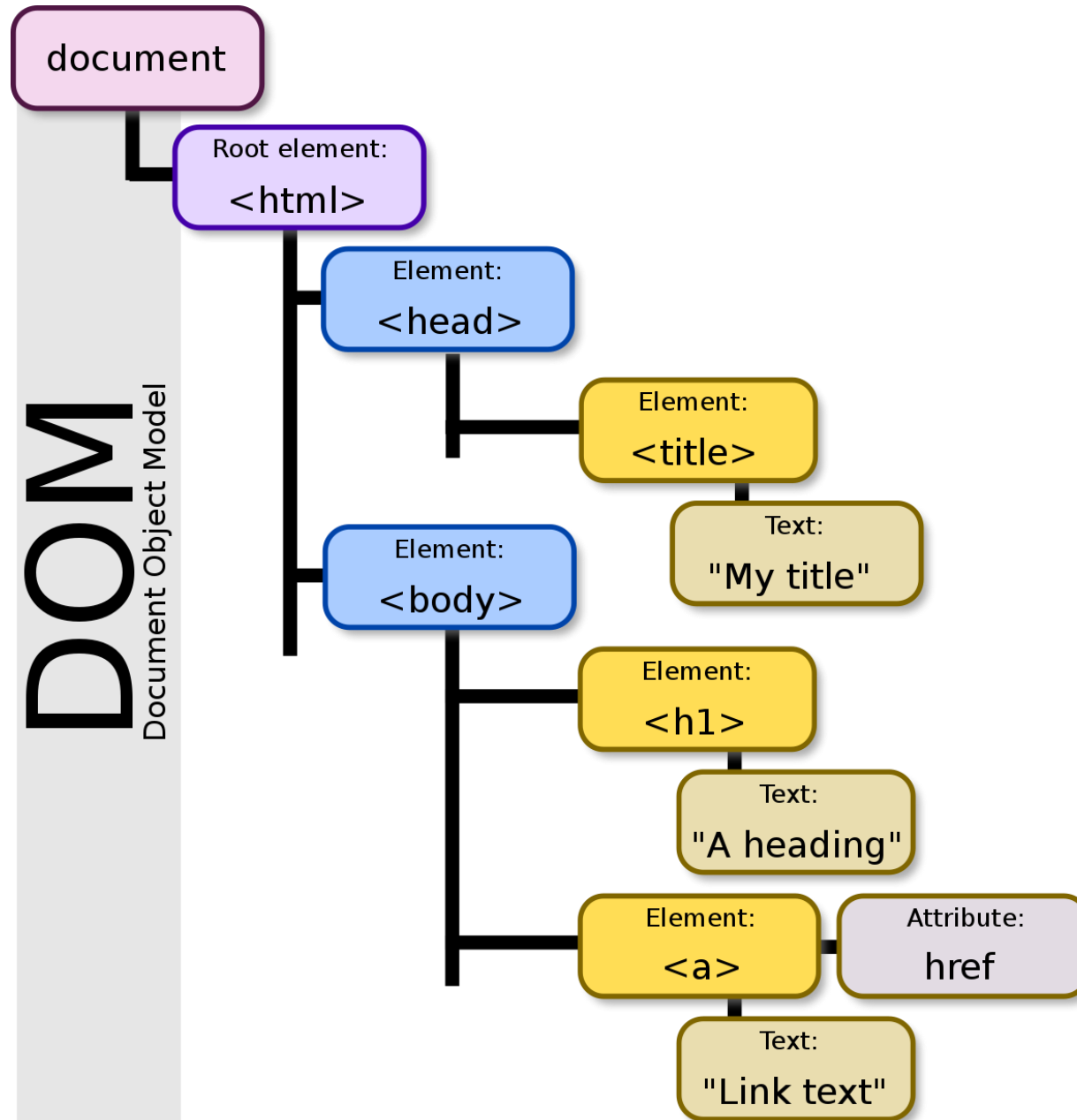
(Document Object Model)

DOM(Document Object Model)



- HTML 문서 요소의 집합!
- HTML 문서는 각각의 node 와 object 의 집합으로 문서를 표현
- 따라서 각각 node 또는 object 에 접근하여 문서 구조 / 스타일 / 내용 등을 변경 할 수 있도록 하는 것!







DOM API

Document Object Model, Application Programming Interface



```
// HTML 요소(Element) 1개 검색/찾기
const boxEl = document.querySelector('.box');

// HTML 요소에 적용할 수 있는 메소드!
boxEl.addEventListener();

// 인수(Arguments)를 추가 가능!
boxEl.addEventListener(1, 2);

// 1 - 이벤트(Event, 상황)
boxEl.addEventListener('click', 2);

// 2 - 핸들러(Handler, 실행할 함수)
boxEl.addEventListener('click', function () {
    console.log('Click~!');
});
```




querySelector

querySelector("요소 선택자")



- 요소 선택자를 사용해서 자신이 가져오고 싶어하는 요소를 가져오는 메소드
- 문서에서 만나는 **제일 첫번째 요소**를 반환 합니다!

```
let boxEl = document.querySelector(".box");  
console.log(boxEl);
```



getElementById

getElementById("ID")



- ID 이름을 불러서 해당 ID를 가지는 요소를 불러오는 메소드

```
const inputEl = document.getElementById("input");
```

```
getElementById  
getElementsByClassName  
getElementsByName  
getElementsByTagName  
getElementsByTagNameNS  
getSelection
```



classList

classList.add / remove / contain



- 선택 요소에 class 를 더하거나, 빼거나, 클래스가 존재하는지 체크하는 메소드
- 해당 기능과 CSS 를 잘 활용하면 변화무쌍(?)한 웹페이지 구성이 가능



```
// HTML 요소(Element) 검색/찾기
```

```
const boxEl = document.querySelector('.box');
```

```
// 요소의 클래스 정보 객체 활용!
```

```
boxEl.classList.add('active');
```

```
let isContains = boxEl.classList.contains('active');
```

```
console.log(isContains); // true
```

```
boxEl.classList.remove('active');
```

```
isContains = boxEl.classList.contains('active');
```

```
console.log(isContains); // false
```

실습



```
<ul>
  <li class="orange">1</li>
  <li>2</li>
  <li>3</li>
  <li>4</li>
  <li>5</li>
  <li id="skyblue">6</li>
  <li>7</li>
  <li>8</li>
  <li>9</li>
  <li>10</li>
</ul>
```

- 클래스가 orange 인 DOM 요소를 선택해서 출력해 주세요
- ID 가 skyblue 인 DOM 요소를 선택하여 출력해 주세요
- UI 리스트의 2번째 자식 li 요소를 선택한 다음, 해당 DOM 요소에 orange 클래스를 부여해 한 다음 출력해 주세요

실습



```
<ul>
  <li class="orange">1</li>
  <li>2</li>
  <li>3</li>
  <li>4</li>
  <li>5</li>
  <li id="skyblue">6</li>
  <li>7</li>
  <li>8</li>
  <li>9</li>
  <li>10</li>
</ul>
```

- Orange 클래스를 가지는 첫번째 DOM 요소를 선택한 다음, 해당 요소가 orange 클래스를 가지는지 if 문으로 확인 후, 가지고 있으면 orange 클래스를 제거해 주세요



setAttribute

setAttribute, html 요소 속성 추가



- 선택한 요소의 속성 값을 직접 지정할 수 있는 메소드
- 요소.setAttribute("속성명", "지정할 속성")

```
searchInputEl.setAttribute("placeholder", "통합검색");
```

```
boxEl.setAttribute("style", "background-color: orange");
```



textContent



```
let boxEl = document.querySelector(".box");  
console.log(boxEl.textContent);  
  
boxEl.textContent = "KDT";  
console.log(boxEl.textContent);
```

1

KDT



innerHTML,
innerText



```
let box1El = document.querySelector(".box1");  
let box2El = document.querySelector(".box2");  
  
box1El.innerHTML = "<button>test</button>";  
box2El.innerText = "<button>test</button>";
```

test

<button>test</button>

실습



```
<body>
  <div class="box1">1</div>
  <div class="box2">2</div>
  <input type="text" />
</body>
```

1

2

- 왼쪽 코드를 JS 와 DOM 을 사용하여 아래와 같이 변경해 주세요~!

[네이버로 이동](#)

박스 내용 및 스타일 변경

아이디를 입력하세요



createElement

createElement, html 요소 생성



- Html DOM 요소를 만들어내는 메소드
- document.createElement("요소 이름")

```
const li = document.createElement("li");
```

```
const box = document.createElement("div");
```



append,
appendChild

append / appendChild, 요소 붙이기



- 특정 DOM 요소에 다른 요소를 자식으로 붙이는 메소드
- DOM요소.append(추가할 내용)
- DOM요소.appendChild(추가할 요소)

```
const li = document.createElement("li");
const checkBtn = document.createElement("input");
checkBtn.setAttribute("type", "checkbox");
li.append(checkBtn);
```

append()



- append() 는 Node 와 String 을 전부 추가 할 수 있어요
- append() 는 여러 가지 값을 한번에 붙일 수 있어요
- append() 는 반환(리턴) 값이 없어요!

appendChild()



- appendChild() 는 Node 만 추가할 수 있어요
- appendChild() 는 한 번에 하나만 추가할 수 있어요
- appendChild() 는 추가한 Node 를 반환(리턴) 합니다!



prepend

prepend()



- **prepend() 는 append() 와는 반대로 부모 노드의 첫번째 요소로 추가합니다!**
- **단, prependChild() 는 존재 하지 않아요!**



remove

remove, 선택한 DOM 을 지우는 메소드



- DOM요소.remove

```
<ul>
  <li>1</li>
  <li class="list">2</li>
  <li>3</li>
  <li>4</li>
</ul>
```

```
const list = document.querySelector(".list");
console.log(list.);
```

- 1
- 3
- 4



parentNode

parentNode, 부모 요소 확인하기



- 특정 DOM 요소의 부모 노드를 가져오는 메소드
- DOM요소.**parentNode**

```
<ul>
  <li>1</li>
  <li class="list">2</li>
  <li>3</li>
  <li>4</li>
</ul>
```

```
const list = document.querySelector(".list");
console.log(list.parentNode);
```

```
▼ <ul>
  ▶ <li>_</li>
  ▶ <li class="list">_</li>
  ▶ <li>_</li>
  ▶ <li>_</li>
  </ul>
```



childNodes



childNodes, 자식 요소 확인하기

- 특정 DOM 요소의 자식 요소를 전부 확인하는 메소드
- DOM요소.**childNodes**

```
<ul>
  <li>1</li>
  <li class="list">2</li>
  <li>3</li>
  <li>4</li>
</ul>
```

```
const list = document.querySelector("ul");
console.log(list.childNodes);
```

```
tt.html:57
NodeList(9) [text, li, text, li.list, text, li, text, li,
text]
  ▶0: text
  ▶1: li
  ▶2: text
  ▶3: li.list
  ▶4: text
  ▶5: li
  ▶6: text
  ▶7: li
  ▶8: text
  length: 9
  ▶[[Prototype]]: NodeList
```



children

children, 자식 요소 확인하기



- 특정 DOM 요소의 자식 노드만을 확인하는 메소드
- DOM요소.children

```
<ul>
  <li>1</li>
  <li class="list">2</li>
  <li>3</li>
  <li>4</li>
</ul>
```

```
const list = document.querySelector("ul");
console.log(list.childNodes);
```

```
▼ HTMLCollection(4) [li, li.list, li, li] ⓘ
  ▶ 0: li
  ▶ 1: li.list
  ▶ 2: li
  ▶ 3: li
    length: 4
  ▶ [[Prototype]]: HTMLCollection
```




Onclick!

onclick



- 각각의 HTML 요소에 속성 값으로 JS 함수를 연결

```
<body>  
  <div class="box" onclick="test();">click</div>  
</body>
```

```
function test() {  
  alert("TEST");  
}
```

이 페이지 내용:

TEST

확인



AddEventListener

addEventListener(이벤트, 명령)



- 선택 요소에 지정한 이벤트가 발생하면, 약속 된 명령어를 실행시키는 메소드

```
let boxEl = document.querySelector(".box");  
  
console.log(boxEl);  
  
boxEl.addEventListener("click", function() {  
    alert("click!");  
})
```

```
document.querySelector(".box").addEventListener("click", function() {  
    alert("click");  
})
```



```
let boxEl = document.querySelector(".box");
console.log(boxEl);
console.log(boxEl.classList.contains("orange"));

boxEl.addEventListener("click", function() {
  boxEl.classList.add("orange");
  console.log(boxEl);
  console.log(boxEl.classList.contains("orange"));
})
```

```
<div class="box">Box!!</div>
```

```
false
```

```
<div class="box orange">Box!!</div>
```

```
true
```

addEventListener 이벤트의 종류



- Click : 클릭
- Mouse 계열
 - Mouseover : 요소에 커서를 올렸을 때
 - Mouseout : 마우스가 요소를 벗어날 때
 - Mousedown : 마우스 버튼을 누르고 있는 상태
 - Mouseup : 마우스 버튼을 떼는 순간
- Focus : 포커스가 갔을 때
- Blur : 포커스가 벗어나는 순간

addEventListener 이벤트의 종류



- Key 계열
 - Keypress : 키를 누르는 순간 + 누르고 있는 동안 계속 발생
 - Keydown : 키를 누르는 순간에만 발생
 - Keyup : 키를 눌렀다가 떼는 순간
- Load : 웹페이지에 필요한 모든 파일(html, css, js 등)의 다운로드가 완료 되었을 때
- Resize : 브라우저 창의 크기가 변경 될 때
- Scroll : 스크롤이 발생할 때
- Unload : 링크를 타고 이동하거나, 브라우저를 닫을 때
- Change : 폼 필드의 상태가 변경 되었을 때

실습



- .box1 를 최초 클릭하면 배경을 orange 색으로 변경하기
- .box1 를 다시 클릭 했을 때 배경이 orange 색이면 skyblue 로 변경 하거나 skyblue 면 orange 로 변경하는 페이지 만들기!
- 색상 변경은 CSS 를 이용하시면 됩니다!
- 해당 실습을 addEventListener 와 onclick 으로 각각 구현하기!
(box1, box2)



querySelectorAll

querySelectorAll("요소 선택자")



- 문서에 존재하는 모든 요소를 찾아주는 메소드
- 모든 요소를 가져와서 배열(같은) 데이터로 만들어 줍니다!

```
<body>
  <div class="box">1</div>
  <div class="box">2</div>
  <div class="box">3</div>
  <div class="box">4</div>
  <div class="box">5</div>
  <div class="box">6</div>
  <div class="box">7</div>
</body>
```

```
let boxEls = document.querySelectorAll(".box");
console.log(boxEls);
```

```
▼ NodeList(7) [div.box, div.box, div.box, div.box, div.box, div.box, div.box]
  ▶ 0: div.box
  ▶ 1: div.box
  ▶ 2: div.box
  ▶ 3: div.box
  ▶ 4: div.box
  ▶ 5: div.box
  ▶ 6: div.box
    length: 7
  ▶ [[Prototype]]: NodeList
```

forEach(function (반복중인 요소, 인덱스) {})



- 찾은 요소 전부에게 명령을 반복적으로 실행해 주는 메소드

```
let boxEls = document.querySelectorAll(".box");
console.log(boxEls);

boxEls.forEach(function (boxEl, index) {
    boxEl.classList.add(`box_${index + 1}`);
})

console.log(boxEls);
```

```
▼ NodeList(7) [div.box, div.box,
  ▶ 0: div.box.box_1
  ▶ 1: div.box.box_2
  ▶ 2: div.box.box_3
  ▶ 3: div.box.box_4
  ▶ 4: div.box.box_5
  ▶ 5: div.box.box_6
  ▶ 6: div.box.box_7
  length: 7
  ▶ [[Prototype]]: NodeList
```

```
▼ NodeList(7) [div.box.box_1, di
  ▶ 0: div.box.box_1
  ▶ 1: div.box.box_2
  ▶ 2: div.box.box_3
  ▶ 3: div.box.box_4
  ▶ 4: div.box.box_5
  ▶ 5: div.box.box_6
  ▶ 6: div.box.box_7
  length: 7
  ▶ [[Prototype]]: NodeList
```



```
// HTML 요소(Element) 모두 검색/찾기  
const boxEls = document.querySelectorAll('.box');  
console.log(boxEls);
```

```
// 찾은 요소들 반복해서 함수 실행!
```

```
// 익명 함수를 인수로 추가!
```

```
boxEls.forEach(function () {});
```

```
// 첫 번째 매개변수(boxEl): 반복 중인 요소.
```

```
// 두 번째 매개변수(index): 반복 중인 번호
```

```
boxEls.forEach(function (boxEl, index) {});
```

```
// 출력!
```

```
boxEls.forEach(function (boxEl, index) {  
    boxEl.classList.add(`order-${index + 1}`);  
    console.log(index, boxEl);  
});
```

실습



- 최 상단에 버튼 하나 만들기
- 7 개의 box 클래스를 가지는 <div> 요소 만들기
- 버튼을 클릭하면 각각의 박스의 배경색이 변경 되는 페이지 만들기
- 박스 색상을 순서에 맞추어 무지개색으로 변경!
- 색상 적용은 CSS 로 하시면 됩니다!



this 와
e.target

this 활용하기



- DOM 요소에서 this 를 사용하면, this 는 자기 자신 Node 를 가르킵니다! → onclick 에서 주로 사용!

```
<ul>
  <li onclick="showThis(this);">1</li>
  <li onclick="showThis(this);">2</li>
  <li onclick="showThis(this);">3</li>
  <li onclick="showThis(this);">4</li>
</ul>
```

```
function showThis(t) {
  console.log(t);
}
```

```
▶ <li onclick="showThis(this);">...</li>
```

e.target 활용하기



- Document 에서 발생하는 이벤트는 이벤트 객체 e 를 통해 확인 할 수 있습니다. → addEventListener 에서 주로 사용!

```
<ul>
  <li class="list">1</li>
  <li>2</li>
  <li>3</li>
  <li>4</li>
</ul>
```

```
const list = document.querySelector(".list");

list.addEventListener("click", function (e) {
  console.log(e.target);
});
```

```
▶ <li class="list">...</li>
```




Defer, Async





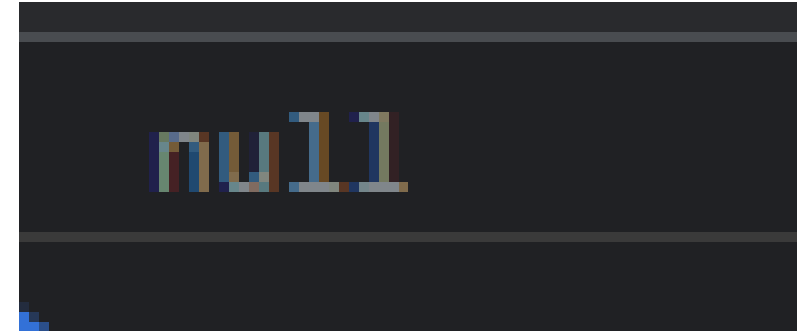
dom.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Document</title>
  <script src="./dom.js"></script>
</head>

<body>
  <div class="box">Box!!</div>
</body>

</html>
```



```
let boxEl = document.querySelector(".box");

console.log(boxEl);
```

dom.js



dom.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Document</title>
</head>

<body>
  <div class="box">Box!!</div>
  <script src="./dom.js"></script>
</body>

</html>
```

```
<div class="box">Box!!</div>
```

```
let boxEl = document.querySelector(".box");
```

```
console.log(boxEl);
```

dom.js



dom.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Document</title>
  <script defer src="./dom.js"></script>
</head>

<body>
  <div class="box">Box!!</div>
</body>

</html>
```

```
<div class="box">Box!!</div>
```

```
let boxEl = document.querySelector(".box");

console.log(boxEl);
```

dom.js





```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
  <script src="./main.js"></script>
</head>
<body>
  <div class="box">Box!!</div>
</body>
</html>
```



```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
</head>
<body>
  <div class="box">Box!!</div>
  <script src="./main.js"></script>
</body>
</html>
```




```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
  <script defer src="./main.js"></script>
</head>
<body>
  <div class="box">Box!!</div>
</body>
</html>
```



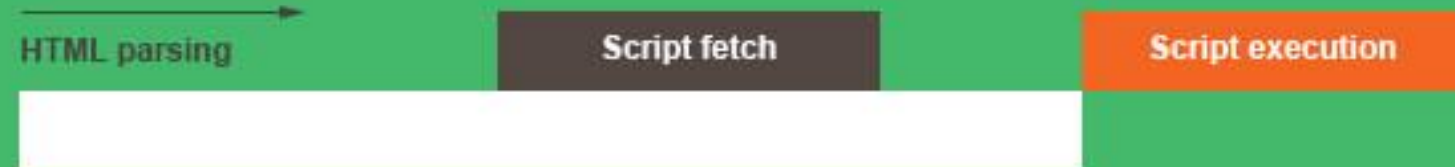
<script>



<script async>



<script defer>





TodoList

만들기!

HTML

```
<!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" />
    <title>Document</title>
    <script defer src="./todo.js"></script>
    <link rel="stylesheet" href="./todo.css" />
  </head>

  <body>
    <div class="header">
      <h1>Tetz Todo List</h1>
    </div>
    <div class="contents">
      <div class="input-part">
        <input type="text" class="input-task" />
        <input type="button" class="input-btn" value="추가" />
      </div>
      <div class="list-part">
        <ul class="todo-list"></ul>
      </div>
    </div>
  </body>
</html>
```



```
* {  
  margin: 0;  
}  
  
.input-part {  
  padding: 30px 0px;  
}  
  
ul {  
  list-style: none;  
  padding-left: 0;  
}  
  
.list-part .todo-list {  
  text-align: center;  
}
```

```
.header {  
  background-color: orange;  
  height: 60px;  
}  
  
const addBtn = document.querySelector(".input-  
add");  
const todoList = document.querySelector(".todo-  
list");  
const inputTask =  
document.querySelector(".input-task");
```

```
.header h1 {  
  text-align: center;  
  line-height: 60px;  
}
```

```
.contents {  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
}
```

CSS

```
const addBtn = document.querySelector(".input-btn");  
const todoList = document.querySelector(".todo-list");  
const inputTask = document.querySelector(".input-task");
```



CSS



- 인풋에 내용을 입력하고 추가 버튼을 클릭하면 할 일이 추가 되는 웹 페이지를 만들어 봅시다!

Tetz Todo List

추가

Tetz Todo List

추가

☐ 투두리스트 만들기

삭제

☐ DOM 마스터하기

삭제



Tetz Todo List

☐ DOM 마스터하기

- 할 일을 입력하지 않고 추가를 누르면
placeholder 에 내용을 입력하세요가 뜹니다!



Tetz Todo List

☒ 투두리스트 만들기

☐ DOM 마스터하기

Tetz Todo List

☐ DOM 마스터하기

- 체크 버튼을 체크하면 할 일 목록에 line-through 가 생깁니다
- 삭제 버튼을 클릭하면 해당 할 일 목록은 삭제됩니다

힌트



- Check 박스가 check 되었는지 여부 체크

```
const checkBtn = document.createElement("input");  
if (checkBtn.checked === true) {}
```

- 특정 요소를 먼저 붙이고 기능을 등록하는 방법은 어려우므로, 먼저 기능을 `addEventListener` 로 등록한 다음 붙이는 방법이 편합니다!

힌트



- 특정 요소를 먼저 붙이고 기능을 등록하는 방법은 어려우므로, 먼저 기능을 `addEventListener` 로 등록한 다음 붙이는 방법이 편합니다!

```
const checkBtn = document.createElement("input");
checkBtn.setAttribute("type", "checkbox");

checkBtn.addEventListener("click", function () {
  if (checkBtn.checked === true) {
    checkBtn.parentNode.style.textDecoration = "line-through";
  } else {
    checkBtn.parentNode.style.textDecoration = "none";
  }
});

addLi.append(checkBtn);
```

미래의 그날이 온다!



유니버설 픽처스 • 유니버설 픽처스 • 유니버설 픽처스
A Universal Picture © 1985 & 2015 Universal Studios.







JS





JS 파일 연결하기

Main.js 파일 연결하기!



- 안전하게 속성 값으로 defer 넣고 시작!

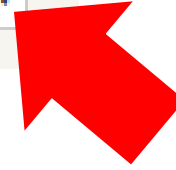


서브메뉴

Search



[Sign In](#) | [My Starbucks](#) | [Customer Service & Ideas](#) | [Find a Store](#)





서브 메뉴, search 변경하기

- 기존은 CSS 의 :focus 를 사용하여 구현
- 따라서, 돋보기를 피해서 input 을 클릭 했을 때에만 작동 → 불편
- 돋보기를 클릭하면 기능이 동작하도록 변경
- 돋보기를 클릭하면 focus 가 가도록 설정하여 기존의 CSS 사용
- 포커스 시 → “통합 검색” 이라는 placeholder 도 추가!(by JS)



```
// SEARCH
const searchEl = document.querySelector(".search");
const searchInputEl = searchEl.querySelector("input");

searchEl.addEventListener("click", function () {
  searchInputEl.focus();
});

searchInputEl.addEventListener("focus", function () {
  searchInputEl.setAttribute("placeholder", "통합검색");
});

searchInputEl.addEventListener("blur", function () {
  searchInputEl.setAttribute("placeholder", "");
});
```



공지 사항



공지사항 스타벅스 e-Gift Card 구매 가능 금액 안내



공지 사항, 슬라이드 메뉴 적용하기



- 해당 기능을 전부 구현하는 것은 매우 어렵고 귀찮은 작업이죠?
- 이럴 때 쓰는 것이 바로! Library
- Swiper 라는 라이브러리를 사용해 봅시다!



Swiper

The Most Modern Mobile Touch Slider


[Get Started](#) [API](#) [Element](#) [React](#) [Vue](#) [Demos](#)

MIT Licensed, v9.0.5 released on February 13, 2023 [Changelog](#)

 34,064 stars

Top Notch Features

- ✓ Library Agnostic
- ✓ Flexbox Layout
- ✓ Multi Row Slides Layout
- ✓ Two-way Control
- ✓ Rich API
- ✓ Parallax Transitions
- ✓ Virtual Slides
- ✓ Mutation Observer
- ✓ Full True RTL Support
- ✓ 3D Effects
- ✓ Full Navigation Control
- ✓ Most Flexible Slides Layout Grid
- ✓ Images Lazy Loading
- ✓ And many more

 Adobe Stock

Get 10 Free Images From
Adobe Stock. Start Now.

<https://swiperjs.com/>



Use Swiper from CDN

If you don't want to include Swiper files in your project, you may use it from CDN. The following files are available:

```
<link
  rel="stylesheet"
  href="https://cdn.jsdelivr.net/npm/swiper@8/swiper-bundle.min.css"
/>

<script src="https://cdn.jsdelivr.net/npm/swiper@8/swiper-bundle.min.js"></script>
```



If you use ES modules in browser, there is a CDN version for that too:

```
<script type="module">
  import Swiper from 'https://cdn.jsdelivr.net/npm/swiper@8/swiper-bundle.esm.browser.min.js'

  const swiper = new Swiper(...)
</script>
```



Add Swiper HTML Layout

Now, we need to add basic Swiper layout to our app:

```
<!-- Slider main container -->
<div class="swiper">
  <!-- Additional required wrapper -->
  <div class="swiper-wrapper">
    <!-- Slides -->
    <div class="swiper-slide">Slide 1</div>
    <div class="swiper-slide">Slide 2</div>
    <div class="swiper-slide">Slide 3</div>
    ...
  </div>
  <!-- If we need pagination -->
  <div class="swiper-pagination"></div>

  <!-- If we need navigation buttons -->
  <div class="swiper-button-prev"></div>
  <div class="swiper-button-next"></div>

  <!-- If we need scrollbar -->
  <div class="swiper-scrollbar"></div>
</div>
```



Swiper CSS Styles/Size

In addition to [Swiper's CSS styles](#), we may need to add some custom styles to set Swiper size:

```
.swiper {  
  width: 600px;  
  height: 300px;  
}
```



Initialize Swiper

Finally, we need to initialize Swiper in JS:

```
const swiper = new Swiper('.swiper', {  
  // Optional parameters  
  direction: 'vertical',  
  loop: true,  
  
  // If we need pagination  
  pagination: {  
    el: '.swiper-pagination',  
  },  
  
  // Navigation arrows  
  navigation: {  
    nextEl: '.swiper-button-next',  
    prevEl: '.swiper-button-prev',  
  },  
  
  // And if we need scrollbar  
  scrollbar: {  
    el: '.swiper-scrollbar',  
  },  
});
```

Swiper 라이브러리 추가



```
<!-- Icons -->
<link rel="stylesheet"
      href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />

<!-- SWIPER -->
<link rel="stylesheet" href="https://unpkg.com/swiper@8/swiper-bundle.min.css" />
<script src="https://unpkg.com/swiper@8/swiper-bundle.min.js"></script>

<!-- main -->
<link rel="stylesheet" href="./css/main.css">
<script defer src="./js/main.js"></script>
```

HTML 코드 추가!



- Swiper 클래스 div 추가
- Swiper-wrapper div 자식으로 swiper-slide div 추가



```
<div class="inner">
  <div class="inner__left">
    <h1>공지사항</h1>
    <div class="swiper">
      <div class="swiper-wrapper">
        <div class="swiper-slide">My DT Pass 이용 안내</div>
        <div class="swiper-slide">My DT Pass 관련 서비스 점검 안내</div>
        <div class="swiper-slide">시스템 개선 및 점검 안내</div>
        <div class="swiper-slide">전자 영수증 서비스 서비스 점검 안내</div>
        <div class="swiper-slide">스타벅스 e-Gift Card 구매 가능 금액 안내</div>
      </div>
    </div>
    <a href="#"><span class="material-symbols-outlined">add_circle</span></a>
  </div>
  <div class="inner__right">
    <h1>스타벅스 프로모션</h1>
    <a href="#"><span class="material-symbols-outlined">expand_circle_down</span></a>
  </div>
</div>
```

CSS 수정



- Swiper 클래스의 CSS 수정
- 크기 값이 필요 → 높이 값 주기!
- 아이템 중앙 정렬!
 - 한 줄 텍스트는??

```
.notice .inner .inner__left .swiper {  
  position: absolute;  
  height: 62px;  
  left: 80px;  
  font-size: 14px;  
}  
  
.notice .inner .inner__left .swiper .swiper-  
wrapper .swiper-slide {  
  line-height: 62px;  
  height: 62px;  
}
```


JS에 Swiper 생성자 함수 추가



- 변수 하나를 만들고 Swiper 생성자 함수 추가
- 설정 값 입력
 - Direction: "vertical"
 - Autoplay: true
 - Loop: true

```
// SWIPER
// SWIPER NOTICE
const swiperNotice = new
Swiper(".notice .inner .inner__left .swiper", {
  direction: "vertical",
  loop: true,
  autoplay: true,
});
```





프로모션

슬라이드



공지사항 시스템 개선 및 점검 안내



스타벅스 프로모션



스타벅스와 함께
하는 것을 즐기는 나만의 여름!

기간 : 2022년 6월 14일(화) - 7월 25일(월)

자세히 보기

STARBUCKS RESERVE™ SUMMER

스타벅스 리저브 서머 원두 & 커피

기간 : 2022년 4월 5일 ~ 2022년 8월 25일

자세히 보기



회원 계정에 등록된 스타벅스 카드
1만원당 별★1개를 즉시 추가로

기간 : 2022년 1월 1일 ~ 2022년 12월

자세히 보기

HTML 구성하기



- Promotion 클래스 div 선언
- Swiper 사용을 위한 swiper 클래스 div 추가
- Swiper-wapper 추가
- Swiper-slide 추가

```
<div class="promotion">  
  <div class="swiper">  
    <div class="swiper-wrapper">  
      test  
    </div>  
  </div>  
</div>
```



CSS 선언해 놓기!



- `.notice .promotion {}`
- `.notice .promotion .swiper {}`
- `.notice .promotion .swiper .swiper-wrapper {}`
- `.notice .promotion .swiper .swiper-wrapper .swiper-slide {}`

Promotion / Swiper 크기 세팅



- 스타벅스 페이지를 참고
- Promotion
 - Height : 658px / background-color: #f6f5ef
- Swiper
 - Height : 553px / Width: $\text{calc}(819\text{px} * 3 + 20\text{px})$

중앙 정렬!



- `Text-align: center;` 는 항상 중앙을 가르킬까요?
- 화면 확대 축소에 따른 중앙 정렬 확인!
- Container 의 크기에 따른 중앙 정렬 → `left: 50% / translate(-50%, 0);`



```
.notice .promotion {  
  position: relative;  
  height: 658px;  
  background-color: #f6f5ef;  
}  
  
.notice .promotion .swiper {  
  width: calc(819px * 3 + 20px);  
  height: 553px;  
  background-color: orange;  
  text-align: center;  
  position: absolute;  
  left: 50%;  
  transform: translate(-50%, 0);  
  top: 40px;  
}
```

이미지 삽입하기!



- Swiper slide 에 이미지 삽입하기
- 자세히 보기 버튼(black) 추가!



```
<div class="promotion">
  <div class="swiper">
    <div class="swiper-wrapper">
      <div class="swiper-slide">
        
        <a href="#" class="btn btn--black">자세히 보기</a>
      </div>
      <div class="swiper-slide">
        
        <a href="#" class="btn btn--black">자세히 보기</a>
      </div>
      <div class="swiper-slide">
        
        <a href="#" class="btn btn--black">자세히 보기</a>
      </div>
    </div>
  </div>
</div>
```

Swiper 생성자 함수 추가!



- 새로운 swiperPromotion 변수에 Swiper 생성자 추가
- Direction / slidePerView / spaceBetween / centeredSlide / loop / autoplay 옵션 추가하기!



```
// SWIPER PROMOTION
const swiperPromotion = new Swiper(".promotion .swiper", {
  direction: "horizontal", // 기본 값
  slidesPerView: 3, // 한번에 보여줄
  spaceBetween: 10, // 아이템간 거리
  centeredSlides: true, // 슬라이드 센터 여부
  loop: true, // 루프 여부
  autoplay: {
    // 자동 재생, 변경 시간 설정
    delay: 5000,
  },
});
```

CSS 수정하기!



- Swiper 배경색 제거
- Swiper 동작을 확인하기
- 보여지는 슬라이드의 클래스명 확인(active)
- 양 옆 슬라이드 반투명 처리
- 자세히 보기 버튼 위치 및 크기 조절



```
.notice .promotion .swiper .swiper-wrapper {  
}  
  
.notice .promotion .swiper .swiper-wrapper .swiper-slide {  
  opacity: 0.5;  
  transition: 0.2s;  
}  
  
.notice .promotion .swiper .swiper-wrapper .swiper-slide-active {  
  opacity: 1;  
}  
  
.notice .promotion .swiper .swiper-wrapper .swiper-slide .btn {  
  width: 150px;  
  position: absolute;  
  left: 0;  
  right: 0;  
  bottom: 0;  
  margin: auto;  
}
```


페이지네이션 / 이동 버튼 추가하기



- Swiper 에서 제공하는 pagination / prev / next 버튼 추가하기

```
<div class="swiper-pagination"></div>  
<div class="swiper-button-prev"></div>  
<div class="swiper-button-next"></div>
```

- Swiper 생성자에 옵션 추가하기!



```
// SWIPER PROMOTION
const swiperPromotion = new Swiper(".promotion .swiper", {
  direction: "horizontal", // 기본 값
  slidesPerView: 3, // 한번에 보여줄
  spaceBetween: 10, // 아이템간 거리
  centeredSlides: true, // 슬라이드 센터 여부
  loop: true, // 루프 여부
  autoplay: {
    // 자동 재생, 변경 시간 설정
    delay: 5000,
    disableOnInteraction: false,
  },
  pagination: {
    el: ".promotion .swiper-pagination", // pagination을 할 엘리먼트 클래스 설정
    clickable: true, // 클릭 가능 여부 설정
  },
  navigation: {
    prevEl: ".promotion .swiper-button-prev", // 이전 버튼 클래스 설정
    nextEl: ".promotion .swiper-button-next", // 이후 버튼 클래스 설정
  },
});
```

페이지네이션 CSS 수정



- Pagination 동작 및 클래스 확인하기
- 각각 bullet 크기 설정
- 선택 효과 설정
 - 백그라운드 이미지 조절
 - 백그라운드 컬러 제거 → 투명하게 처리



```
.notice .promotion .swiper .swiper-pagination {  
}  
  
.notice .promotion .swiper-pagination .swiper-pagination-bullet {  
  width: 12px;  
  height: 12px;  
}  
  
.notice .promotion .swiper-pagination .swiper-pagination-bullet-active {  
  background-image: url("../images/promotion_on.png");  
  background-size: cover;  
  background-color: transparent;  
}
```



이동 버튼 CSS 수정

- 버튼 위치부터 확인 → `position: absolute; / top: 300px;`
- 버튼 스타일 수정
 - `Width / height : 55px;`
 - `Border: 2px solid #333;`
 - `Color: #333;`
 - `Border-radius: 50%;`
 - `Cursor: pointer;`
- 화살표 크기도 수정하기 → `after` 의 폰트 사이즈 수정



```
.notice .promotion .swiper-button-prev,  
.notice .promotion .swiper-button-next {  
  width: 55px;  
  height: 55px;  
  border: 2px solid #333;  
  color: #333;  
  border-radius: 50%;  
  position: absolute;  
  top: 300px;  
  cursor: pointer;  
  z-index: 1;  
}  
  
.notice .promotion .swiper-button-prev:after,  
.notice .promotion .swiper-button-next:after {  
  font-size: 25px;  
}
```



이동 버튼 위치 지정

- Position: absolute; 는 이미 지정된 상태!
- Left: 400px? / right: 400px?
- 반응형 확인하기!
- Left: 50% / translate(-550px, 0);
- Right: 50% / translate(550px, 0);
- Promotion 크기를 기반으로 배치하기 → 단 일정 크기 부터는 반응형 필요!



```
.notice .promotion .swiper-button-prev {  
  left: 50%;  
  transform: translate(-550px, 0);  
}  
  
.notice .promotion .swiper-button-next {  
  right: 50%;  
  transform: translate(550px, 0);  
}
```


Promotion 에 overflow 지정



- Swiper 의 크기로 인해 가로 스크롤 발생!
- Promotion 에 overflow: hidden; 설정으로 스크롤 없애기!

Autoplay?



- Pagination 에 재생, 멈춤 버튼을 만드는 것은 고통스러우니 공지사항의 + 버튼에 autoplay 멈춤, 재시작 기능을 넣어 봅시다!
- <a> 태그에 onclick 주기!
- Js 파일에 함수 생성
- Swiper의 autoplay 관련 함수
 - .autoplay.start() / .autoplay.stop()
- Autoplay.stop() 적용 → 어? 그런데 안먹네요?



```
<a href="" onclick="controlAutoPlay()">  
  <span class="material-symbols-outlined">add_circle</span>  
</a>
```

Autoplay?



- <a> 태그의 특성 상 href 로 인해 페이지를 새로 고침
- 이를 해결하기 위해 href="javascript:함수명();" 을 사용
- A 태그의 효과는 쓰고 싶지만, 페이지 새로 고침을 막으려면
 - 사용

```
<a href="javascript:stopAutoPlay();">  
  <span class="material-symbols-outlined">add_circle</span>  
</a>
```

Autoplay?



- swiperPromotion 의 값을 console.log 로 찍어보기
- swiperPromotion.autoplay.running 값에 따라 stop / start 여부 결정!




```
function controlAutoPlay() {  
  if (swiperPromotion.autoplay.running == true) {  
    swiperPromotion.autoplay.stop();  
  } else {  
    swiperPromotion.autoplay.start();  
  }  
}
```

Promotion Toggle



- Promotion section 토글 기능 추가!
- Toggle 용 버튼에 클래스 주기!
 - Javascript:void(0); 처리
- JS 에서 버튼에 addEventListener 추가
 - Promotion 에 hide 클래스 추가 / 삭제 처리
- CSS 에서 promotion 처리



```
<div class="inner_right">
  <h1>스타벅스 프로모션</h1>
  <a href="javascript:void(0);" class="toggle-promotion"><span
    class="material-symbols-outlined">expand_circle_down</span></a>
</div>
```

HTML

```
.notice .promotion {
  position: relative;
  height: 658px;
  background-color: #f6f5ef;
  overflow: hidden;
  transition: height 0.4s;
}

.notice .promotion.hide {
  height: 0px;
}
```

CSS



```
// Toggle Promotion
const promotionEl = document.querySelector(".promotion");
const promotionToggleBtn = document.querySelector(".toggle-promotion");

promotionToggleBtn.addEventListener("click", function () {
  if (promotionEl.classList.contains("hide")) {
    promotionEl.classList.remove("hide");
  } else {
    promotionEl.classList.add("hide");
  }
});
```

JavaScript

Promotion Toggle 버튼 애니메이션!



- Promotion section 토글 버튼 애니메이션 추가하기!
- 보여 졌을 때 180deg 돌아가도록 처리하기!
- JS 로만 수정 가즈아!



```
// Toggle Promotion
const promotionEl = document.querySelector(".promotion");
const promotionToggleBtn = document.querySelector(".toggle-promotion");
const promotionToggleIcon = document.querySelector(".toggle-promotion span");

promotionToggleBtn.addEventListener("click", function () {
  if (promotionEl.classList.contains("hide")) {
    promotionEl.classList.remove("hide");
  } else {
    promotionEl.classList.add("hide");
  }

  if (promotionToggleIcon.style.transform === "rotate(180deg)") {
    promotionToggleIcon.style.transform = "rotate(0deg)";
  } else {
    promotionToggleIcon.style.transform = "rotate(180deg)";
  }
});
```

JavaScript

```
.notice .inner__right .toggle-promotion span {
  transition: 0.4s;
}
```

CSS



스크롤에 따른 애니메이션 처리!



Scroll 발생을 체크

- 브라우저 레벨에서 발생하는 개념이므로
- document → window 사용!

```
// SCROLL
let scrollYpos;
window.addEventListener("scroll", function () {
  scrollYpos = window.scrollY;
  console.log(scrollYpos);
});
```

- 콘솔 로그에서 확인하기!



페이지가 로드되면 바로!

- Visual 의 애니메이션 재생 필요
- Visual 의 inner 클래스에 :hover 로 처리 했던 것으로 클래스로 변경 → :hover → animate
- 페이지가 로드 되면 바로 실행되는 함수
 - Window.onload() 사용

```
window.onload = () => {  
  const visualInner = document.querySelector(".visual .inner");  
  visualInner.classList.add("animate");  
};
```



엘살바도르 애니메이션!

- 스크롤 위치 찾기! → 300 정도가 적당해 보이네요!
- Hover 로 처리해 두었던 애니메이션을 animate 클래스로 변경!

```
if (scrollYpos > 300) {  
    const elsalvadorAnimate = document.querySelector(".elsalvador");  
    elsalvadorAnimate.classList.add("animate");  
}
```

JavaScript

```
.elsalvador.animate .inner .img_product {  
    transform: translate(0px, 0);  
    opacity: 1;  
    transition: 2.5s;  
}
```

CSS

실습, 나머지 애니메이션!



- 이디오피아
- Pick your favorite
- Magazine
- Find a store



**JUST
DID
IT.**

