Hello,

KDT 웹 개발자 양성 프로젝트

5기!

with 

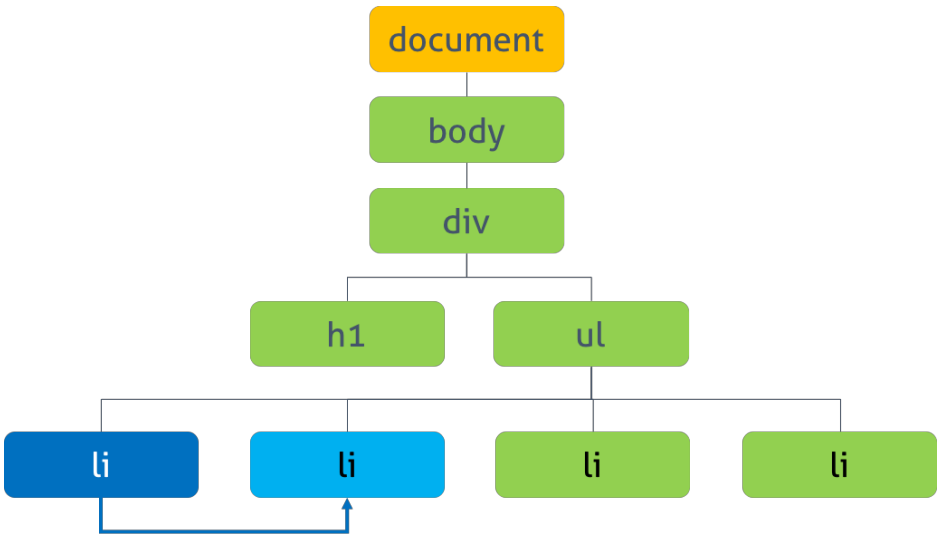
메소드 체이닝

```
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
```

DOM

(Document Object Model)



DOM API

Document Object Model, Application Programming Interface

querySelector

getElementById

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

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>

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);
```

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] i
▸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] i
  ▶0: li
  ▶1: li.list
  ▶2: li
  ▶3: li
    length: 4
  ▶[[Prototype]]: HTMLCollection
```

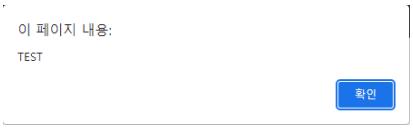
Onclick!

onclick

- 각각의 HTML 요소에 속성 값으로 JS 함수를 연결

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

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



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");
})
```

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
```

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!

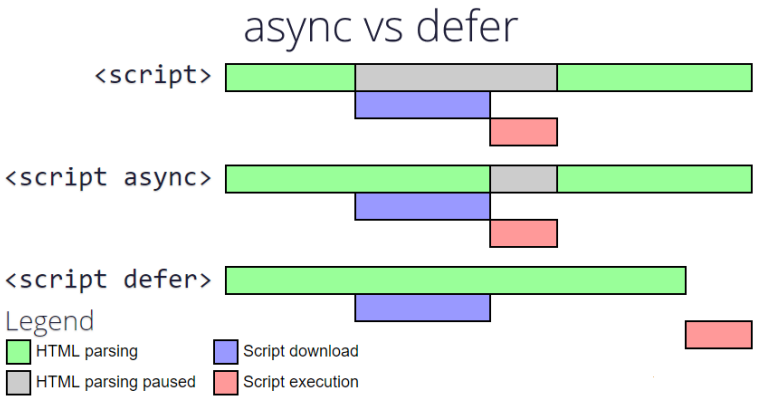
```html
<ul>
  <li class="list">1</li>
  <li>2</li>
  <li>3</li>
  <li>4</li>
</ul>
```

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

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

```
▶<li class="list">…</li>
```

Defer,

Async



TodoList

만들기!

HTML

```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>
```

CSS

```css
* {
  margin: 0;
}

ul {
  list-style: none;
  padding-left: 0;
}

.header {
  background-color: orange;
  height: 60px;
}

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

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

.input-part {
  padding: 30px 0px;
}

.list-part .todo-list {
  text-align: center;
}
```

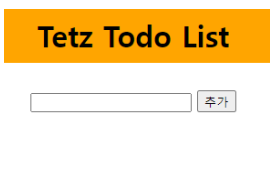
```js
const addBtn = document.querySelector(".input-btn");
const todoList = document.querySelector(".todo-list");
const inputTask = document.querySelector(".input-task");
```

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

JS

실습

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



실습

Tetz Todo List

내용을 입력하세요! 추가

☐DOM 마스터하기 삭제

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

실습

Tetz Todo List

추가

☑투두리스트 만들기 삭제

☐DOM 마스터하기 삭제

Tetz Todo List

추가

☐DOM 마스터하기 삭제

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

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);
```

취업 관련 정보 안내

공부용 사이트

블로그 운영

취업 관련 정보

https://github.com/jojoldu/junior-recruit-scheduler

공부용 사이트

https://poiemaweb.com/

블로그 운영

블로그 운영

- https://velog.io/
  - 깔끔 / 개발자 많음
- https://www.tistory.com/
  - 전통의 강자 / 코드로 커스터마이징 가능 / 수익 창출 가능
- https://medium.com/
  - 수익성이 좋음 / 영어 기반 서비스 / 팀 블로그에 좋다 합니다!?
- 그 외 아무 서비스나 환영

- https://blog.naver.com/xenosign

티스토리 (Tistory)	진입장벽 매우 낮음 커스마이징이 쉬움(괜찮은 템플릿이 많이 풀려있음) 수익/통계/구독 있음	코드블럭이 그저그럼 개발친화적이라기엔 조금 애매함(다양한 주제로 글쓰기 가능) 카카오가 브런치를 많이 밀어주면서 포지션이 더 애매해짐	리뷰, 여행, 개발
벨로그 (Velog)	프론트엔드 개발자 벨로퍼트님이 개발 편함 깔끔함 심플함 글 작성이 매우 직관적임 개발특화 조회수/통계/구독 기능 있음(최근 추가됨) 팀블로그로 좋음	마크다운으로만 작성 가능 수익은 추후 추가 예정 커스마이징 어려움 개인이 운영	개발
미디엄 (Medium)	해외에서 많이 사용하는듯 구독 수익 통계 기능 있음 피드백이 좋음 개발특화 팀블로그로 좋음	글쓰기 개수 제한 있음(결제하면 업그레이드 가능) 한글 폰트가 별로라고함 커스터마이징 어려움	개발

블로그 운영

- 기본 사항 → 주차 별 교육 내용 정리 & 회고 올리기!
  - 수업에서 느끼거나 배운 부분 정리
  - 스스로 개선해야할 부분 또는 목표 정리
- [포스코 x 코딩온] 을 제목에 붙여 주세요!

블로그 운영

- Best Case
  - TIL(Today I Learned) 작성 or 개념별로 정리해서 올리기
  - 주말에 주차 별 정리 내용 추가!
- Base Case
  - 주차 별 교육 내용 정리 & 회고 올리기!


지금 시작합니다

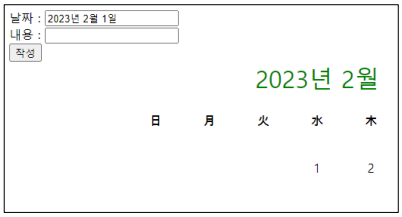

DOM

실습, 스케쥴 달력 만들기!

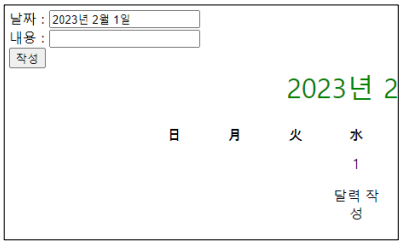
- 이번에는 스케쥴 달력을 만들어 봅시다! ☺

실습, 스케쥴 달력 만들기!

- 날짜 or 날짜 칸을 클릭하면 날짜 input 에 날짜가 입력 됩니다



- 그리고 내용을 입력한 다음 작성을 누르면 해당 날짜에 스케쥴이 추가 됩니다!



실습, 스케쥴 달력 만들기!

- 작성 버튼에는 writeSchedule() 이라는 함수가 걸려 있습니다! → 잘 구현해 주세요 ☺
- 스케쥴은 div 요소로 테이블의 td 요소에 추가하면 됩니다!
  - 그러면 안되지만...
- 이미 작성 된 스케쥴을 클릭하면 삭제가 되도록 만들어 주세요!
- 물론, 요일 또는 날짜를 클릭했을 때 삭제가 되면 안되겠죠!?

힌트

- 클릭 된 요소는 전역 let 변수로 놓고, e.target 을 활용하면 어떤 요소가 클릭이 되었는지 알 수 있습니다!
- 유의미한 클릭 이벤트는 달력 내부에서만 발생하겠죠?

```
// JS 구현
const calendar = document.querySelector("table");
const date = document.querySelector("#date");

// 클릭 된 요소를 저장하기 위한 전역 변수
let targetEl;

calendar.addEventListener("click", function (e) {});
```

힌트

- 어떤 태그가 클릭 되었는지를 확인하는 방법은!?
- tagName 이라는 API 를 사용하세요!
- 클릭 된, 태그의 이름이 "대문자 문자열"로 리턴 됩니다!

```
if (e.target.tagName === "P") {}
```


미래의 그날이 온다!
백투더퓨처
OCT 21 2015

JS 파일

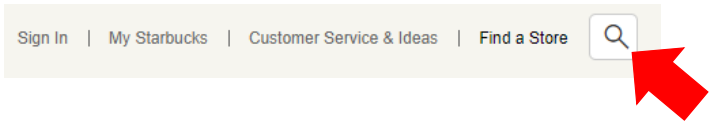
연결하기

Main.js 파일 연결하기!

- 안전하게 속성 값으로 defer 넣고 시작!

서브메뉴

Search

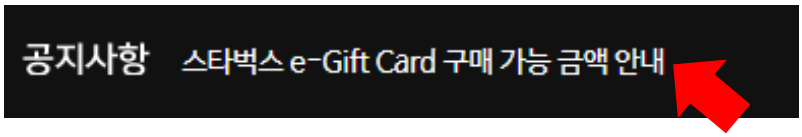


서브 메뉴, search 변경하기

- 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)

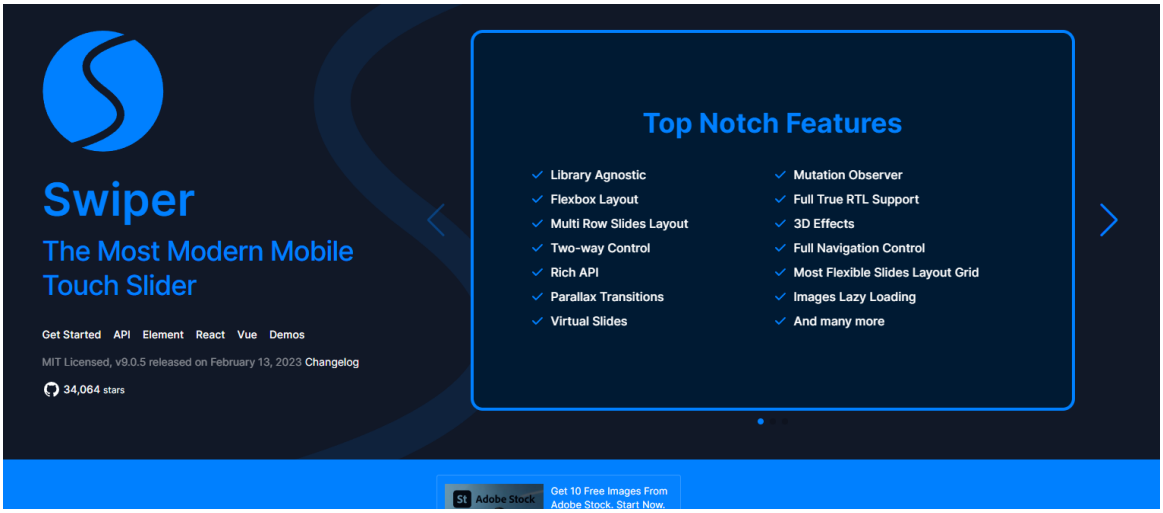
```
// SEARCH
const searchEl = document.querySelector(".search");
const searchInputEl = searchEl.querySelector("input");

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

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

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

공지 사항



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!



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??

```css
.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;
}
```

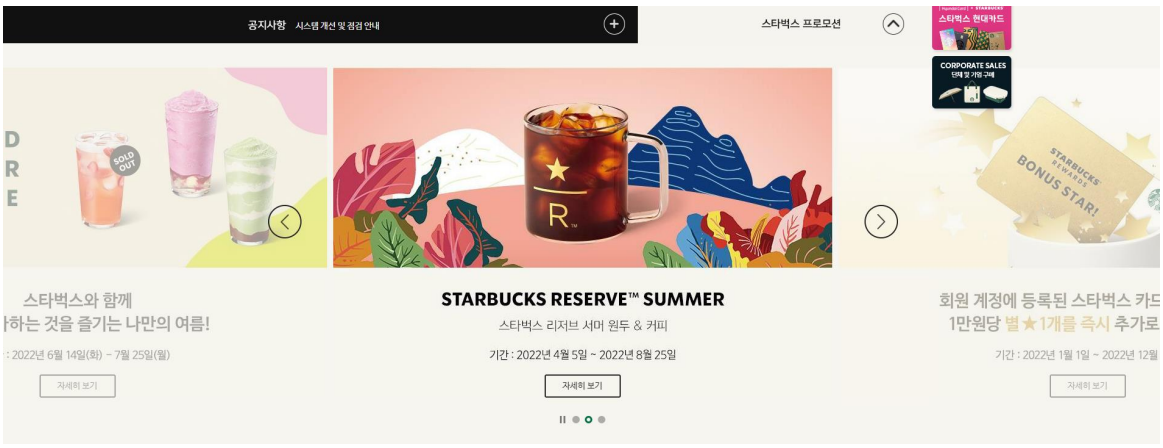
JS에 Swiper 생성자 함수 추가

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

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

프로모션

슬라이드



HTML 구성하기

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

```html
<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: calc(819px * 3 + 20px)

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);

```css
.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">
                        <img src="./images/promotion_1.jpg" alt="스타벅스와 함께 내가 좋아하는 것을 즐기는 나만의 여름" />
                        <a href="#" class="btn btn--black">자세히 보기</a>
                    </div>
                    <div class="swiper-slide">
                        <img src="./images/promotion_2.jpg" alt="스타벅스와 함께 내가 좋아하는 것을 즐기는 나만의 여름" />
                        <a href="#" class="btn btn--black">자세히 보기</a>
                    </div>
                    <div class="swiper-slide">
                        <img src="./images/promotion_3.jpg" alt="스타벅스와 함께 내가 좋아하는 것을 즐기는 나만의 여름" />
                        <a href="#" class="btn btn--black">자세히 보기</a>
                    </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 아이템 수
  spaceBetween: 10, // 아이템간 거리
  centeredSlides: true, // 슬라이드 센터 여부
  loop: true, // 루프 여브
  autoplay: {
    // 자동 재생, 변경 시간 설정
    delay: 1000,
    disableOnInteraction: false,
  },
});
```

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

```css
.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

```html
<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 여브
  autolay: {
    // 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

```css
.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

```css
.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!

```css
.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!

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

Autoplay?

- Pagination 에 재생, 멈춤 버튼을 만드는 것은 고통 스러우니 공지 사항의 + 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">check_circle</span>
</a>
```

```
function controlAutoPlay() {
  swiperPromotion.autoplay.start();
}
```

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
  - <a href="javascript:void(0);"> 사용

```
<a href="javascript:controlAutoPlay();"><span class="material-symbols-
outlined"> check_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

```HTML
<div class="inner__right">
            <h1>스타벅스 프로모션</h1>
            <a href="javascript:controlAutoPlay();" class="toggle-promotion">
              <span class="material-symbols-outlined"> check_circle </span>
            </a>
          </div>
```

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

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

```Javascript
// 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");
  }
});
```

Promotion Toggle 버튼 애니메이션!

- Promotion section 토글 버튼 애니메이션 추가하기!
- 보여 졌을 때 180deg 돌아가도록 처리하기!
- 보이는 상황에서는 show 클래스를 추가해서 180deg 돌리기!

Javascript

```javascript
const promotionEl = document.querySelector(".promotion");
const promotionToggleBtn = document.querySelector(".toggle-promotion");
promotionToggleBtn.addEventListener("click", function () {
  if (promotionEl.classList.contains("hide")) {
    promotionEl.classList.remove("hide");
    promotionToggleBtn.classList.add("show");
  } else {
    promotionEl.classList.add("hide");
    promotionToggleBtn.classList.remove("show");
  }
});
```

CSS

```css
.notice .inner__right .toggle-promotion {
  transition: 0.4s;
}
.notice .inner__right .toggle-promotion.show {
  transform: rotate(180deg);
}
```

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 :active 로 처리 했던 것으로 클래스로 변경 → :active → 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!
- active 로 처리해 두었던 애니메이션을 animte 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.


수고하셨습니다!