

Hello,

KDT 웹 개발자 양성 프로젝트

5기!

with





TodoList

만들기!

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

```
function deleteTask(t) {
  t.parentNode.remove();
}
```

```
addBtn.addEventListener("click", function () {
  if (inputTask.value === "") {
    inputTask.setAttribute("placeholder", "내용을 입력하세요!");
  } else {
    const addLi = document.createElement("li");
    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";
      }
    });
    const deleteBtn = document.createElement("input");
    deleteBtn.setAttribute("type", "button");
    deleteBtn.setAttribute("value", "삭제");
    deleteBtn.setAttribute("onclick", "deleteTask(this);");
    addLi.append(checkBtn);
    addLi.append(inputTask.value);
    addLi.append(deleteBtn);
    todoList.appendChild(addLi);
  }
  inputTask.value = "";
});
```



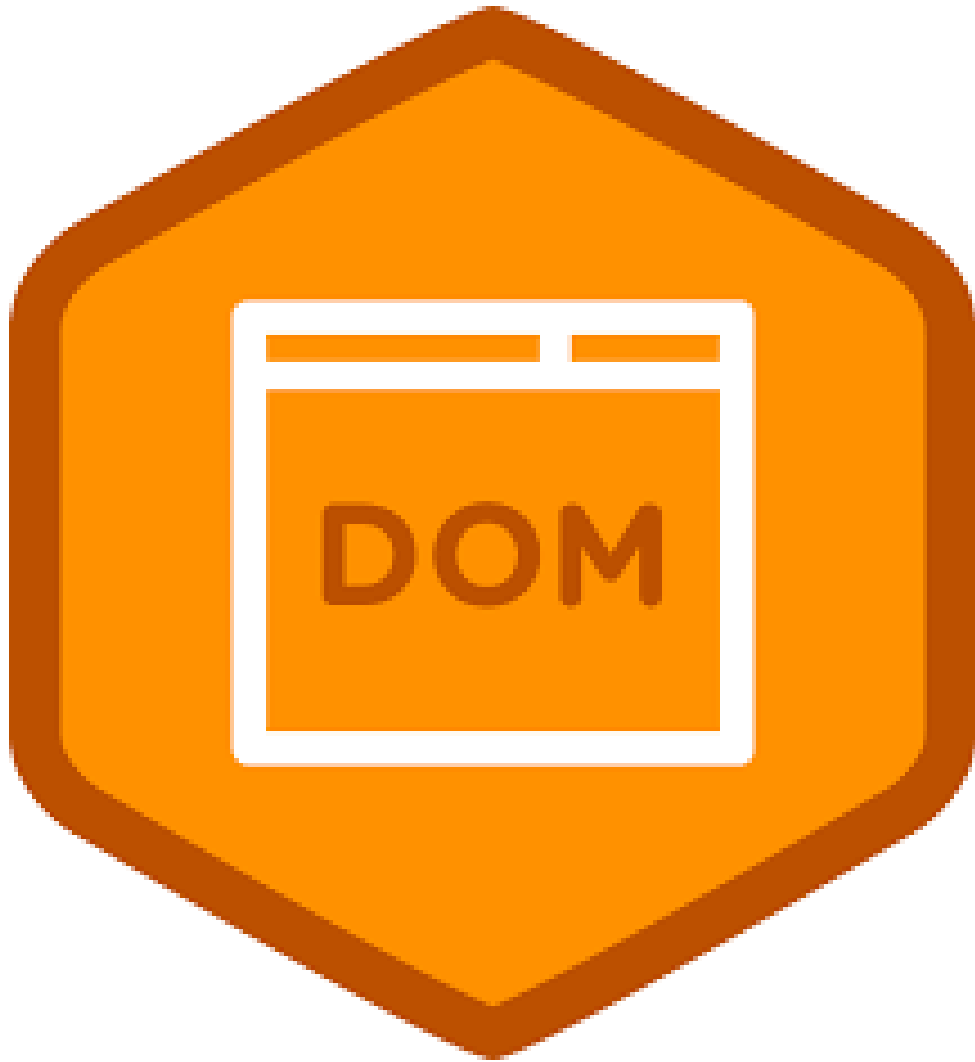
알려드립니다!!



취업 관련 정보 안내

공부용 사이트

블로그 운영



실습, 스케줄 달력 만들기!



- 이번에는 스케줄 달력을 만들어 봅시다! 😊

날짜 :

내용 :

작성

2023년 2월

日	月	火	水	木	金	土
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				



실습, 스케줄 달력 만들기!

- 날짜 or 날짜 칸을 클릭하면 날짜 input 에 날짜가 입력 됩니다
- 그리고 내용을 입력한 다음 작성을 누르면 해당 날짜에 스케줄이 추가 됩니다!

날짜 : 2023년 2월 1일
내용 :

2023년 2월

日	月	火	水	木
			1	2

날짜 : 2023년 2월 1일
내용 :

2023년 2월

日	月	火	水
			1
			달력 작성



```
const calendar = document.querySelector("table");
const date = document.querySelector("#date");
let targetEl;
```

```
calendar.addEventListener("click", function (e) {
  if (e.target.tagName === "P") {
    date.value = `2023년 2월 ${e.target.textContent}일`;
    targetEl = e.target.parentNode;
  } else if (e.target.tagName === "DIV") {
    e.target.remove();
  } else if (e.target.tagName === "TD") {
    date.value = `2023년 2월 ${e.target.children[0].textContent}일`;
    targetEl = e.target;
  }
});
```

```
function writeSchedule() {
  let content = document.querySelector("#content");
  let addEl = document.createElement("div");

  addEl.innerText = content.value;
  targetEl.append(addEl);
  content.value = "";
}
```


미래의 그날이 온다!



TM & © 1985 & 2015 Universal Studios. A Universal Picture © 1985 & 2015 Universal Studios.







JS





JS 파일 연결하기

Main.js 파일 연결하기!



- 안전하게 속성 값으로 defer 넣고 시작!

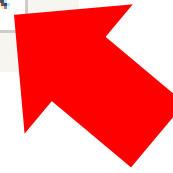


서브메뉴

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", "");
});
```



지금 시작합니다





	ACTIVE Class	:FOCUS	활성 여부
페이지 로딩시	X	X	X
돋보기 클릭	O	O	O
돋보기 다시 클릭	X	X	X
외부 클릭	O	X	X



외부 클릭 인 상태일 때, 돋보기를 누르면 focus 만 보내는 방식으로 해결!



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

// 외부 요소를 클릭하여 포커스 아웃 된 경우를 처리하기 위한 전역 변수
let isNotSearchClick = false;

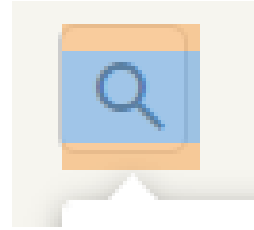
// 돋보기 아이콘이 아닌 외부 요소를 클릭하였는지를 확인 하기 위하여
// 문서에 이벤트 리스너 걸기
document.addEventListener("click", function (e) {
  // 요소를 클릭한게 아니라면 --> 외부를 클릭하여 포커스 아웃이 되었다면, 전역 변수를 true;
  if (!e.target.classList.contains("material-symbols-outlined")) {
    isNotSearchClick = true;
    // 요소를 클릭했다면 --> 전역 변수를 false;
  } else {
    isNotSearchClick = false;
  }
});
```



```
searchIconEl.addEventListener("click", function () {  
    // 클릭이 한번도 안된 상태라서 active 클래스가 없다면?  
    // active 클래스 부여 및 focus 보내기, placeholder 지정  
    if (!searchEl.classList.contains("active")) {  
        searchEl.classList.add("active");  
        searchInputEl.focus();  
        searchInputEl.setAttribute("placeholder", "통합 검색");  
        // 외부 요소 클릭으로 포커스가 아웃 된 상태에서, 다시 돋보기를 누르면  
        // active 클래스 부여 및 focus 보내기, placeholder 지정  
    } else if (isNotSearchClick) {  
        searchInputEl.focus();  
        searchInputEl.setAttribute("placeholder", "통합 검색");  
        isNotSearchClick = false;  
        // active 클래스와, focus 가 모두 있는 상태에서 클릭을 하면 active 클래스를 제거하여  
        // 토글 효과  
    } else {  
        searchEl.classList.remove("active");  
    }  
});  
  
// 포커스 아웃이 되면 일단 placeholder 초기화  
searchInputEl.addEventListener("blur", function (e) {  
    searchInputEl.setAttribute("placeholder", "");  
});
```



```
header .inner .sub-menu .search > .material-  
symbols-outlined {  
  position: absolute;  
  text-align: center;  
  width: 36px;  
  margin: auto;  
  right: -3px;  
  top: 0;  
  bottom: 0;  
  height: 24px;  
  cursor: pointer;  
}
```



돋보기 말고 input 이 클릭 되면 해당 로직이 작동하지 않으므로, 작은 input 을 가리기 위해
돋보기를 키웠습니다 😊😊😊😊😊





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 아이템 수
  spaceBetween: 10, // 아이템간 거리
  centeredSlides: true, // 슬라이드 센터 여부
  loop: true, // 루프 여부
  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



```
.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!

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

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">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
 - 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



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

HTML

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

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



```
// 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!
- 보이는 상황에서는 show 클래스를 추가해서 180deg 돌리기!



```
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");
  }
});
```

Javascript

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

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!



비주얼 애니메이션 적용하기!

- 시간 순서대로 이미지를 띄우는 애니메이션 적용하기!
- 각 아이템별로 0.4s 시간 주기 + delay 적용

```
.visual .inner.animate .item-1.image {  
  transition: 0.4s;  
  opacity: 1;  
  transition-delay: 0.4s;  
}
```



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 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





코딩

테스트!



최근 모든 기업들이 코테를 봅니다!

- 규모가 큰 기업일 수록 코테에 가중치를 줍니다!
- 코테를 하면
 - 일단 JS 를 친숙하게 다룰 수 있습니다!
 - 알고리즘과 친숙해 집니다! (쓰는 알고리즘은 거기서 거기!)
 - 높은 랭크를 자소서에 첨부하면 분명히 도움이 됩니다!
- 그리고 코딩에 대한 근본적 자신감이 올라 갑니다!
- 그러니 한번 시작 해보시죠!!



회원 가입부터 가시죠!

- 다양한 방법으로 회원 가입을 하시면 됩니다!
- https://programmers.co.kr/account/sign_in?referer=https%3A%2F%2Fprogrammers.co.kr%2F

다른 계정으로 로그인 하기



Facebook



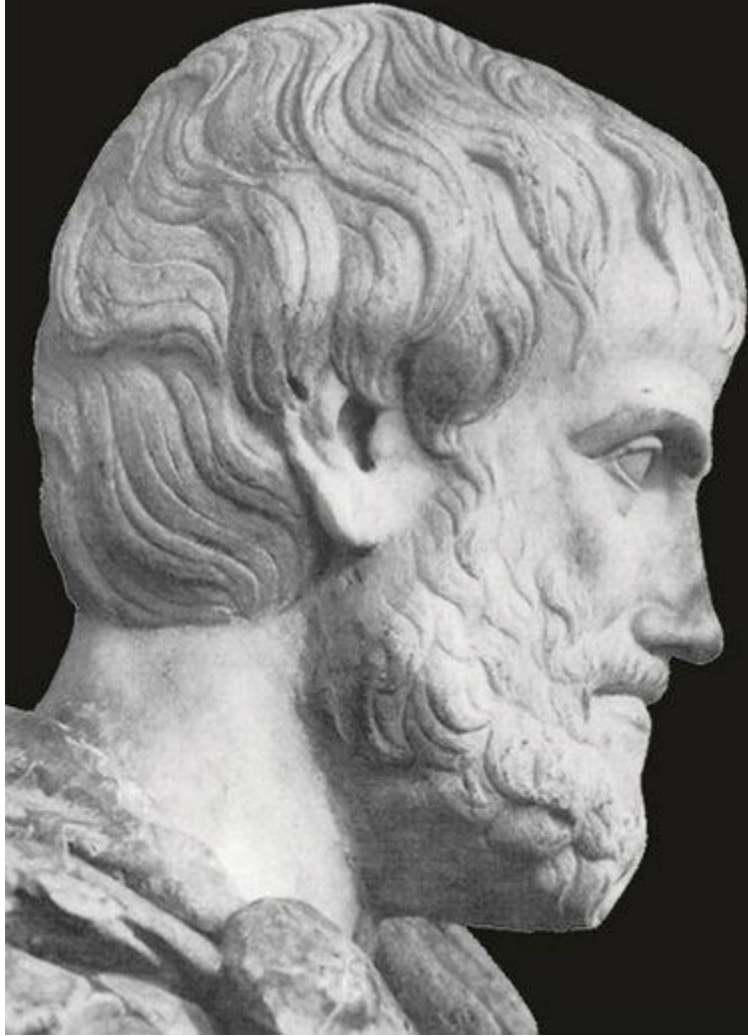
Github



Google



Kakao



Well Begun
is Half Done.

-Aristotle

프로그래머스 익히기!



- <https://school.programmers.co.kr/learn/courses/30/lessons/12937>
- <https://school.programmers.co.kr/learn/courses/30/lessons/120813?language=javascript>



문자 관련

JS 기능들!

문자열 길이, `.length`



- 문자열 `.length()`;

```
const str = "Hello, world";  
  
console.log(str.length);  
  
// 12
```

문자열 위치 찾기, indexOf



- 문자를 찾을 문자열.indexOf("찾을 문자열");

```
const str = "Hello, world";  
  
console.log(str.indexOf("world"));  
  
// 7
```

문자열 자르기, slice



- 문자열.slice(시작위치, 종료위치);

```
const str = "Hello, world";  
  
console.log(str.slice(0, 5));  
  
//Hello
```

문자열 바꾸기, **replace**



- 문자열.**replace**("찾을 문자" , "바꿀 문자");

```
const str = "Hello, world";  
  
console.log(str.replace("world", "뽀로로"));  
  
// Hello, 뽀로로
```


문자열 반복하기, repeat



- 문자열.repeat(반복 횟수);

```
let str = "우영";  
  
let longStr = str.repeat(10) + "우";  
  
console.log(longStr);
```

앞 뒤 공백 문자 제거하기, trim



- 문자열.trim();

```
const str = "    Hello, world    ";  
  
console.log(str.trim());  
  
// Hello, world
```

수박수박수박수박



- <https://school.programmers.co.kr/learn/courses/30/lessons/12922>



숫자 + 수학 관련

JS 기능들!

수학 함수들



```
console.log("abs: ", Math.abs(-999));      // 절대값

console.log("min: ", Math.min(10, 33));     // 작은 값
console.log("min: ", Math.min(10, 33, 6, 2, -1)); // 작은 값

console.log("max: ", Math.max(10, 33));     // 큰 값
console.log("max: ", Math.max(10, 33, 2, 1, 77)); // 큰 값

console.log("ceil: ", Math.ceil(3.14));     // 올림

console.log("floor: ", Math.floor(3.14));    // 버림

console.log("round: ", Math.round(3.6));    // 반올림
console.log("round: ", Math.round(3.4));    // 반올림

console.log("random: ", Math.random());     // 랜덤
```

실습



- 주어진 배열에서 가장 큰 수와 작은 수의 찾아서 소수점을 버림 처리
- 해당 수 절대값의 평균을 구하기

```
let nums = [-1.23, 13, 14.52, -33.53, 30];
```

- 0 ~ 100 까지의 숫자 중에서 랜덤한 정수가 나오도록 만들기

약수의 합



- <https://school.programmers.co.kr/learn/courses/30/lessons/12928>



배열 관련

기능들!



배열, `arr.map`

- 기존 배열에 특정 작업을 한 배열을 리턴하는 메소드
- `arr.map(function(배열의값, 인덱스, 원본배열) {});`

```
const fruits = ["사과", "파인애플", "수박", "포도", "아로니아"];

let obj = fruits.map(function (item, index) {
  return {
    id: index,
    name: item,
  };
});

console.log(obj);
```

화살표 함수!



```
const fruits = ["사과", "파인애플", "수박", "포도", "아원지"];

let obj = fruits.map((item, index) => {
  return {
    id: index,
    name: item,
  }
})

console.log(obj);
```

- 요런 식으로 줄이기가 가능합니다!

화살표 함수!



// 1단계

```
const numbers = [1, 2, 3, 4, 5, 6];  
numbers.map(function (item) {  
  return item * 2;  
});
```

// 2단계

```
const numbers = [1, 2, 3, 4, 5, 6];  
numbers.map((item) => {  
  return item * 2;  
});
```

// 3단계

```
const numbers = [1, 2, 3, 4, 5, 6];  
numbers.map((item) => item * 2);
```

실습



- 1 부터 100 까지의 숫자가 들어 있는 배열을 for 문으로 만들기
- Map 메소드를 이용하여 해당 배열의 합산을 구하기!

문자열도 사실 배열이다!



- `apple = ["a", "p", "p", "l", "e"];`

```
let string = "apple";

for (let i = 0; i < string.length; i++) {
  console.log(string[i]);
}

for (letter of string) {
  console.log(letter);
}

for (index in string) {
  console.log(letter);
}
```



배열 함수, **filter**

- 조건에 부합하는 배열 요소만을 반환
- **arr.filter**(function (매개변수) { return 조건});

```
let numbers = [1, 2, 3, 4, 5, 6];  
let arr;  
  
arr = numbers.filter((num) => num > 3);  
  
console.log(arr);  
  
const words = ['spray', 'limit', 'elite', 'exuberant',  
  'destruction', 'present'];  
  
const result = words.filter(word => word.length > 6);  
  
console.log(result);
```

배열 함수, **includes**



- 해당 배열에 지정한 요소가 있는지 확인하는 메소드
- **arr.includes(요소);**

```
let numbers = [1, 2, 3, 4, 5, 6];

console.log(numbers.includes(3));
console.log(numbers.includes(7));

const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];

console.log(words.includes('elite'));
console.log(words.includes('pororo'));
```

실습



```
let fruits1 = ["사과", "딸기", "파인애플", "수박", "참외", "오렌지", "자두", "망고"];  
let fruits2 = ["수박", "사과", "참외", "오렌지", "파인애플", "망고"];
```

- 두 배열에서 동일한 요소만을 가지는 배열 **same** 만들기
- 두 배열에서 서로 다른 요소만을 가지는 배열 **diff** 만들기



배열 함수, **find** / **findIndex**

- **find** : 배열에 특정 값이 있는지 찾고 반환
- **findIndex** : 배열에 특정 값이 있는지 찾고 위치를 반환

```
const fruits = ["Apple", "Banana", "Cherry"];

const result1 = fruits.find(item => {
  return /^A/.test(item);
})

const result2 = fruits.findIndex(item => {
  return /^C/.test(item);
})

console.log(result1);
console.log(result2);
```

배열, `arr.reduce`



- Reduce 는 가장 많은 기능을 할 수 있습니다!
- `arr.reduce`(function(누산기, 배열의값, 인덱스, 원본배열), 누산기 초기 값);



```
const arr = [1, 2, 3, 4, 5];  
const result = arr.reduce((acc, item, index) => {  
  return (acc += item);  
}, 0);  
console.log(result); // 15
```

```
const arr2 = [1, 2, 3, 4, 5];  
const result2 = arr2.reduce((acc, item, idx) => {  
  return (acc += item);  
}, 10);  
console.log(result2); // 25
```



```
const numbers = [2, -5, -123, 59, -5480, 24, 0, -69, 349, 3];
const result = numbers.reduce(
  (acc, item, index) => {
    if (item < 0) {
      // 처리할 현재 요소가 음수일 경우
      acc[0]++;
    } else if (item > 0) {
      // 처리할 현재 요소가 양수일 경우
      acc[1]++;
    }
    return acc;
  },
  [0, 0]
);
console.log(result); // [4, 5]
```

idx	cur	acc
0	2	[0, 1]
1	-5	[1, 1]
2	-123	[2, 1]
3	59	[2, 2]
4	-5480	[3, 2]
5	24	[3, 3]
6	0	[3, 3]
7	-69	[4, 3]
8	349	[4, 4]
9	3	[4, 5]

실습



- 1 부터 100 까지의 숫자가 들어 있는 배열을 for 문으로 만들기
- Reduce 메소드를 이용하여 해당 배열의 합산을 구하기! 단, reduce 메소드의 누산기를 이용하기

평균 구하기!



- <https://school.programmers.co.kr/learn/courses/30/lessons/12944>



Object

객체!

Object.assign()



- 객체의 병합에 사용되는 메소드

```
const obj1 = { a: 1, b: 2 };  
const obj2 = { b: 3, c: 4 };  
  
const returnObj = Object.assign(obj1, obj2);  
  
console.log(obj1);  
console.log(returnObj);
```

```
▶ {a: 1, b: 3, c: 4}  
▶ {a: 1, b: 3, c: 4}
```

- 기존에 선언된 객체에 사용하지 않고 가상의 Object 라는 것을 불러와서 사용 → 정적 메소드!



구조 분해 할당

(비구조화 할당)

구조 분해 할당



- 객체 또는 배열의 각각의 값을 분해하여 변수에 넣어 사용하는 방법

```
const user = {  
  id: 1,  
  name: "tetz",  
  email: "xenosign@naver.com",  
};  
  
const { id, name, email, address } = user;  
// 기본값 설정 const { id, name, email, address = "KOREA" } = user;  
// 특정 변수에 넣기 const { id, name: tetz, email, address = "KOREA" } = user;  
  
console.log(id);  
console.log(name);  
console.log(email);  
console.log(address);  
  
const fruits = ["사과", "딸기", "망고", "수박"];  
const [a, b, c, d] = fruits;  
console.log(a, b, c, d);
```



전개 연산자

전개 연산자



- 배열의 값을 , 단위로 구분 하여 전개시켜주는 연산자

```
const fruits = ["사과", "바나나", "수박"];
console.log(fruits);
console.log(...fruits);
// console.log("사과", "바나나", "수박");

function conLog(a, b, c) {
  console.log(a, b, c);
}

conLog(fruits[0], fruits[1], fruits[2]);
conLog(...fruits);
```



나머지 연산자(매개 변수일 때)

- 매개 변수에 너무 많은 값이 들어올 때 나머지 연산자를 사용하여 한꺼번에 처리 가능

```
const fruits = ["사과", "바나나", "수박", "망고",  
"딸기"];
```

```
function conLog(a, b, ...c) {  
  console.log(a, b, c);  
}
```

```
conLog(...fruits);
```

```
const fruits = ["사과", "바나나", "수박", "망고", "딸기"];
```

```
function conLog(...rest) {  
  rest.forEach((element) => {  
    console.log(element);  
  });  
}
```

```
conLog(...fruits);
```



문자열을 사실 진짜 배열로!

```
let string = "apple";  
let strToArr = [...string];  
console.log(strToArr);  
  
let strToArr2 = str.split("");  
console.log(strToArr2);
```

- 전개 연산자(...) 사용으로 "apple" 을 "a", "p", "p", "l", "e" 로 변환
- 그것을 [] 안에 넣어서 배열로 변환!

번호 가리기!



- <https://school.programmers.co.kr/learn/courses/30/lessons/12948>



자릿수 더하기



- <https://school.programmers.co.kr/learn/courses/30/lessons/12931>

모의 교사



- <https://school.programmers.co.kr/learn/courses/30/lessons/42840>

