**main.js**

'use strict'

/\*\*

\* 검색창 제어

\*/

// 검색창 요소(.search) 찾기.

const searchEl = document.querySelector('.search')

const searchInputEl = searchEl.querySelector('input')

// 검색창 요소를 클릭하면 실행.

searchEl.addEventListener('click', function () {

searchInputEl.focus()

})

// 검색창 요소 내부 실제 input 요소에 포커스되면 실행.

searchInputEl.addEventListener('focus', function () {

searchEl.classList.add('focused')

searchInputEl.setAttribute('placeholder', '통합검색')

})

// 검색창 요소 내부 실제 input 요소에서 포커스가 해제(블러)되면 실행.

searchInputEl.addEventListener('blur', function () {

searchEl.classList.remove('focused')

searchInputEl.setAttribute('placeholder', '')

})

/\*\*

\* 페이지 스크롤에 따른 요소 제어

\*/

// 페이지 스크롤에 영향을 받는 요소들을 검색!

const badgeEl = document.querySelector('header .badges')

const toTopEl = document.querySelector('#to-top')

// 페이지에 스크롤 이벤트를 추가!

// 스크롤이 지나치게 자주 발생하는 것을 조절(throttle, 일부러 부하를 줌)

window.addEventListener('scroll', \_.throttle(function () {

// 페이지 스크롤 위치가 500px이 넘으면.

if (window.scrollY > 500) {

// Badge 요소 숨기기!

gsap.to(badgeEl, .6, {

opacity: 0,

display: 'none'

})

// 상단으로 스크롤 버튼 보이기!

gsap.to(toTopEl, .2, {

x: 0

})

// 페이지 스크롤 위치가 500px이 넘지 않으면.

} else {

// Badge 요소 보이기!

gsap.to(badgeEl, .6, {

opacity: 1,

display: 'block'

})

// 상단으로 스크롤 버튼 숨기기!

gsap.to(toTopEl, .2, {

x: 100

})

}

}, 300))

// 상단으로 스크롤 버튼을 클릭하면,

toTopEl.addEventListener('click', function () {

// 페이지 위치를 최상단으로 부드럽게(0.7초 동안) 이동.

gsap.to(window, .7, {

scrollTo: 0

})

})

/\*\*

\* 순서대로 나타나는 기능

\*/

// 나타날 요소들(.fade-in) 찾기.

const fadeEls = document.querySelectorAll('.visual .fade-in')

// 나타날 요소들을 하나씩 반복해서 처리!

fadeEls.forEach(function (fadeEl, index) {

// 각 요소들을 순서대로(delay) 보여지게 함!

gsap.to(fadeEl, 1, {

delay: (index + 1) \* .7,

opacity: 1

})

})

/\*\*

\* 슬라이드 요소 관리

\*/

new Swiper('.notice-line .swiper-container', {

direction: 'vertical', // 수직 슬라이드

autoplay: true, // 자동 재생 여부

loop: true // 반복 재생 여부

})

new Swiper('.promotion .swiper-container', {

// direction: 'horizontal', // 수평 슬라이드

autoplay: { // 자동 재생 여부

delay: 5000 // 5초마다 슬라이드 바뀜

},

loop: true, // 반복 재생 여부

slidesPerView: 3, // 한 번에 보여줄 슬라이드 개수

spaceBetween: 10, // 슬라이드 사이 여백

centeredSlides: true, // 1번 슬라이드가 가운데 보이기

pagination: { // 페이지 번호 사용 여부

el: '.promotion .swiper-pagination', // 페이지 번호 요소 선택자

clickable: true // 사용자의 페이지 번호 요소 제어 가능 여부

},

navigation: { // 슬라이드 이전/다음 버튼 사용 여부

prevEl: '.promotion .swiper-prev', // 이전 버튼 선택자

nextEl: '.promotion .swiper-next' // 다음 버튼 선택자

}

})

new Swiper('.awards .swiper-container', {

// direction: 'horizontal', // 수평 슬라이드

autoplay: true, // 자동 재생 여부

loop: true, // 반복 재생 여부

spaceBetween: 30, // 슬라이드 사이 여백

slidesPerView: 5, // 한 번에 보여줄 슬라이드 개수

// slidesPerGroup: 5, // 한 번에 슬라이드 할 개수(전체 개수로 나뉘어야 함)

navigation: { // 슬라이드 이전/다음 버튼 사용 여부

prevEl: '.awards .swiper-prev', // 이전 버튼 선택자

nextEl: '.awards .swiper-next' // 다음 버튼 선택자

}

})

/\*\*

\* Promotion 슬라이드 토글 기능

\*/

// 슬라이드 영역 요소 검색!

const promotionEl = document.querySelector('.promotion')

// 슬라이드 영역를 토글하는 버튼 검색!

const promotionToggleBtn = document.querySelector('.toggle-promotion')

// 슬라이드 영역 숨김 여부 기본값!

let isHidePromotion = false

// 토글 버튼을 클릭하면,

promotionToggleBtn.addEventListener('click', function () {

// 슬라이드 영역 숨김 여부를 반댓값으로 할당!

isHidePromotion = !isHidePromotion

// 요소를 숨겨야 하면,

if (isHidePromotion) {

promotionEl.classList.add('hide')

// 요소가 보여야 하면,

} else {

promotionEl.classList.remove('hide')

}

})

/\*\*

\* 부유하는 요소 관리

\*/

// 범위 랜덤 함수(소수점 2자리까지)

function random(min, max) {

// `.toFixed()`를 통해 반환된 '문자 데이터'를,

// `parseFloat()`을 통해 소수점을 가지는 '숫자 데이터'로 변환

return parseFloat((Math.random() \* (max - min) + min).toFixed(2))

}

// 부유하는(떠 다니는) 요소를 만드는 함수

function floatingObject(selector, delay, size) {

gsap.to(

selector, // 선택자

random(1.5, 2.5), // 애니메이션 동작 시간

{

delay: random(0, delay), // 얼마나 늦게 애니메이션을 시작할 것인지 지연 시간을 설정.

y: size, // `transform: translateY(수치);`와 같음. 수직으로 얼마나 움직일지 설정.

repeat: -1, // 몇 번 반복하는지를 설정, `-1`은 무한 반복.

yoyo: true, // 한번 재생된 애니메이션을 다시 뒤로 재생.

ease: Power1.easeInOut // Easing 함수 적용.

}

)

}

floatingObject('.floating1', 1, 15)

floatingObject('.floating2', .5, 15)

floatingObject('.floating3', 1.5, 20)

/\*\*

\* 요소가 화면에 보여짐 여부에 따른 요소 관리

\*/

// 관리할 요소들 검색!

const spyEls = document.querySelectorAll('section.scroll-spy')

// 요소들 반복 처리!

spyEls.forEach(function (spyEl) {

new ScrollMagic

.Scene({ // 감시할 장면(Scene)을 추가

triggerElement: spyEl, // 보여짐 여부를 감시할 요소를 지정

triggerHook: .8 // 화면의 80% 지점에서 보여짐 여부 감시

})

.setClassToggle(spyEl, 'show') // 요소가 화면에 보이면 show 클래스 추가

.addTo(new ScrollMagic.Controller()) // 컨트롤러에 장면을 할당(필수!)

})

/\*\*

\* 올해가 몇 년도인지 계산

\*/

const thisYear = document.querySelector('.this-year')

thisYear.textContent = new Date().getFullYear()

**youtube.js**

// Youtube IFrame API를 비동기로 로드합니다.

var tag = document.createElement('script');

tag.src = "https://www.youtube.com/iframe\_api";

var firstScriptTag = document.getElementsByTagName('script')[0];

firstScriptTag.parentNode.insertBefore(tag, firstScriptTag);

// onYouTubePlayerAPIReady 함수 이름은,

// Youtube IFrame Player API에서 사용하는 이름이기 때문에,

// 다르게 지정하면 동작하지 않습니다!

// 그리고 함수는 전역(Global) 등록해야 합니다!

function onYouTubePlayerAPIReady() {

// <div id="player"></div>

new YT.Player('player', {

videoId: 'An6LvWQuj\_8', // 최초 재생할 유튜브 영상 ID

playerVars: {

autoplay: true, // 자동 재생 유무

loop: true, // 반복 재생 유무

playlist: 'An6LvWQuj\_8' // 반복 재생할 유튜브 영상 ID 목록

},

events: {

// 영상이 준비되었을 때,

onReady: function (event) {

event.target.mute() // 음소거!

}

}

})

}