

7강. 배열과 객체, 함수



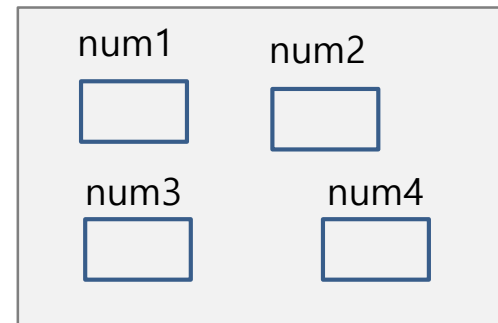
배열(객체)이란?

배열 사용의 필요성

- 정수 10개를 이용한 학생의 성적 프로그램을 만들때
10개의 변수를 선언

var num1, var num2, var num3... var num10;
정보가 흩어진 채 저장되고, 변수 이름이 많아
비효율적이고 관리하기 어렵다.

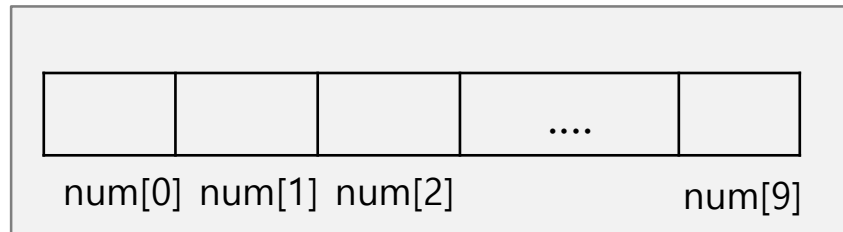
메모리



배열 사용의 장점

- 인덱스를 이용하여 순차(순서)적으로 관리할 수 있다 -> 효율적이다.

메모리



배열(객체)이란?

배열(Array)이란?

- 여러 개의 연속적인 값을 저장하고자 할 때 사용하는 자료형이다.

배열의 생성

방법1 : `let 배열 이름 = [값1, 값2, 값3...]`

방법2 : `let 배열 이름 = new Array(5)`

배열의 생성2

방법1 - Array 객체를 사용한 배열

Array 1

10 20 30 40 50

65, A

66, B

67, C

68, D

69, E

70, F

71, G

72, H

73, I

74, J

```
<h1>Array 1</h1>
```

```
<div id="result"></div>
```

```
<div id="result2"></div>
```

```
<script src="js/array1.js"></script>
```

배열의 생성2

방법1 - Array 객체를 사용한 배열

```
let result = document.getElementById("result");
let arr = new Array(5); //길이가 5인 배열 생성
console.log(arr.length); //5

for(let i=0; i<arr.length; i++){
  arr[i] = (i+1) * 10; //배열에 저장
  result.textContent += arr[i] + " ";
} //10, 20, 30, 40, 50
```

배열의 생성2

방법1 – Array 객체를 사용한 배열

```
let alphabet = new Array(26); //길이가 26인 배열 생성
let ch = 'A'; //A의 아스키코드값 65
ch = ch.charCodeAt(0); //65
console.log(ch); //65
console.log(String.fromCharCode(ch)); //A //아스키코드값 65

for(let i=0; i<alphabet.length; i++){
    alphabet[i] = ch; //65, 66, 67, ..., 90
    ch++;
} //A~Z까지 아스키코드값 저장

for(let i=0; i<alphabet.length; i++){
    // console.log(alphabet[i] + ", " + String.fromCharCode(alphabet[i]));
    result2.innerHTML += `${alphabet[i]}, ${String.fromCharCode(alphabet[i])}<br>`;
} //65, A 66, B 67, C ... 90, Z
```

배열의 생성

방법 2 – 배열 생성 및 초기화

```
//문자를 저장할 배열 선언
let result = document.getElementById("result");
let arr = ['사과', '배', '포도', '바나나'];
console.log(arr.length); //5

//배열 요소 출력
for(let i=0; i<arr.length; i++){
    console.log(arr[i]); //사과, 배, 포도, 바나나
    //result.innerHTML = result.innerHTML + arr[i];
    result.innerHTML += (arr[i] + " ");
}

//향상된 for문(배열 요소를 순차적으로 접근)
//for(변수 of 배열){}
for(let item of arr){ //let 생략 가능
    console.log(item);
}
```

배열의 생성

방법 2 – 배열 생성 및 초기화

```
//배열 요소 추가
arr[4] = '딸기'; //5번째 요소 추가

//배열 요소 추가2
arr.push('망고'); //마지막 요소로 추가
console.log(arr);
result.innerHTML += `<br>배열 요소 추가 후 : ${arr}`;

//배열 요소 삭제
arr[1] = undefined; //2번째 요소 삭제
console.log(arr); //['사과', undefined, '포도', '바나나', '딸기']
console.log(arr.length); //5

//배열 요소 삭제2
arr.splice(1, 1); //2번째 요소부터 1개 삭제
console.log(arr); //['사과', '포도', '바나나', '딸기']
console.log(arr.length); //4
result.innerHTML += `<br>배열 요소 삭제 후 : ${arr}`;
```


배열의 생성

방법 2 – 배열 생성 및 초기화

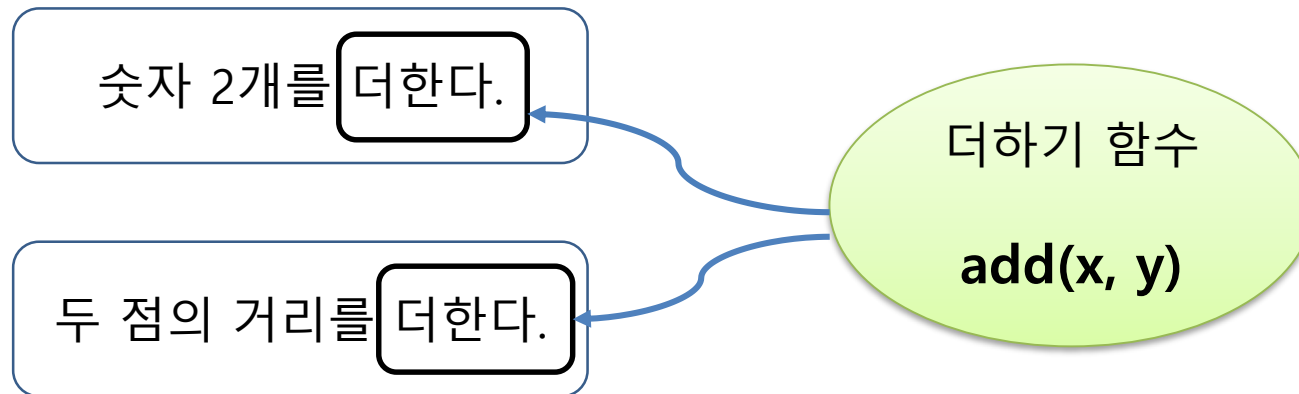
```
//배열 요소 수정
arr[2] = '키위'; //3번째 요소 수정
console.log(arr);
result.innerHTML += `<br>배열 요소 수정 후 : ${arr}`;

//배열 요소 수정2
arr.splice(2, 1, '수박'); //3번째 요소부터 1개를 '수박'으로 수정
console.log(arr);
result.innerHTML += `<br>배열 요소 수정2 후 : ${arr}`;
```

함수의 정의

- 함수(Function)

- 특정한 하나의 기능을 수행하는 일련의 코드
- 중복되는 기능은 함수로 구현하여 그 함수를 호출하여 사용한다.



함수 만들기

- 함수 (정의와 호출)

명시적 함수(이름있음)

```
function 함수이름(){  
    실행문;  
}
```



함수 호출: 함수이름();

함수의 유형

- 함수의 유형

반환값이 없는 경우

```
<h2>반환값이 없는 함수</h2>
<div id="result"></div>
<div id="result2"></div>

<h3>구구단 출력</h3>
<button onclick="gugudan(6)">확인</button>
<div class="display"></div>

<script src="js/func1.js"></script>
```

함수의 유형

- 함수의 유형

```
let result1 = document.getElementById("result");
let result2 = document.getElementById("result2");

//함수 정의
function sayHello(){
  | result.innerText = "안녕~";
  | ...
}

function sayHello2(name){
  | result2.innerText = `안녕~ ${name}`;
  |
}

//호출
sayHello();

sayHello2("영우");
sayHello2("Elsa")
```

함수의 유형

- 함수의 유형

```
//구구단
let display = document.querySelector(".display")
display.innerHTML = "<br>";

function gugudan(x){ //x는 매개변수
  for(let i = 1; i <= 9; i++){
    display.innerHTML += `${x} x ${i} = ${x*i}<br>`
  }
}
```

함수의 유형

- 함수의 유형

반환값이 있는 경우 – return 키워드 사용

반환값이 있는 함수

4의 제곱: 16

-10의 절대값: 10

응원 메시지

확인

Good Luck!!

```
<h2>반환값이 있는 함수</h2>
```

```
<p id="result"></p>
```

```
<p id="result2"></p>
```

```
<h3>응원 메시지</h3>
```

```
<button onclick="showMessage()">확인</button>
```

```
<div class="display"></div>
```

```
<script src="js/returnFunc.js"></script>
```

함수의 유형

- 함수의 유형

반환값이 있는 경우 – return 키워드 사용

```
//객체에 접근
let result = document.getElementById("result")
let result2 = document.getElementById("result2");
let display = document.querySelector(".display");

//제곱수 계산 함수
function square(x){
    return x * x;
}

//절대값 계산 함수
function myAbs(n){
    if(n < 0)
        return -n;
    else
        return n;
}
```


함수의 유형

- 함수의 유형

반환값이 있는 경우 – return 키워드 사용

```
//메시지 출력
function showMessage(){
  return display.innerHTML = "<em>Good Luck!!</em>";
}

//함수 호출
let num = square(4);
result.textContent = "4의 제곱: " + num;

let val = myAbs(-10);
result2.textContent = "-10의 절대값: " + val;
```

함수 만들기

- 익명 함수 만들기(정의와 호출)

```
let 변수이름 = function(){  
    실행문;  
}
```



함수 호출: 변수이름();

익명 함수

- 익명 함수

```
<h2>익명 함수</h2>
<div id="result"></div>
<div id="result2"></div>

<script src="js/func2.js"></script>
```

```
let greeting = function(){
  document.getElementById("result").innerText = "안녕하세요~";
}

let greeting2 = function(name){
  document.getElementById("result2").innerText = "안녕~ " + name;
}

greeting(); // 호출

greeting2("선화");
```

화살표 함수

● 화살표 함수 만들기(정의와 호출)

```
const 변수이름 = ()=>{  
    실행문;  
}
```



함수 호출: 변수이름();

- 매개변수 없는 경우: `() => { }`
- 매개변수 1개인 경우: `(x) => { }`
- `return`이 있는 함수:
`(x, y) => { return x + y }`

화살표 함수

- 화살표 함수

화살표 함수

안녕하세요~
안녕~ 상진

두 수의 합

계산

덧셈 결과: 30

```
<h2>화살표 함수</h2>
<div id="result"></div>
<div id="result2"></div>

<h3>두 수의 합</h3>
<button onclick="showAdd(10, 20)">계산</button>
<p class="total"></p>

<script src="js/arrow.js"></script>
```

화살표 함수

● 화살표 함수

```
let total = document.querySelector(".total");

const greeting = () => {
  document.getElementById("result").innerText = "안녕하세요~";
}

const greeting2 = (name) => {
  document.getElementById("result2").innerText = `안녕~ ${name}`;
}

//덧셈 함수
const add = (x, y) => {
  return x + y;
};

//add()를 호출
const showAdd = (a, b) => {
  total.textContent = `덧셈 결과: ${add(a, b)}`
}

//함수 호출
greeting();
greeting2("상진");
```

합살표 함수

- 짝수 / 홀수 판별

짝수/홀수 판별

정수 입력

255는(은) 홀수입니다.

```
<h2>짝수/홀수 판별</h2>
```

```
<label for="num">정수 입력 </label>
```

```
<input type="text" id="num" placeholder="예 : 1">
```

```
<button onclick="getResult()">확인</button>
```

```
<div id="result"></div>
```

```
<script src="js/even_odd.js"></script>
```

합살표 함수

- 짝수 / 홀수 판별

```
function getResult(){
  const result = document.getElementById("result");
  let num = document.getElementById("num").value; //입력된 값
  console.log(num);
  console.log(typeof(num)); //string

  if(num === "" || isNaN(num)){
    result.textContent = "유효한 숫자를 입력하세요";
  }else{
    num = parseInt(num); //숫자로 변환

    if(num % 2 == 0){
      result.textContent = `${num}는(은) 짝수입니다.`;
    } else {
      result.textContent = `${num}는(은) 홀수입니다.`;
    }
  }
}
```


전역 변수와 지역 변수

메모리 영역 - 프로세스가 운영체제로 할당받은 메모리 영역 구분



전역 변수와 지역 변수

● 지역 변수(local variable)의 유효 범위

- 생성 : 함수 또는 제어문의 코드블럭(중괄호{ }) 내부에서 생성됨 -지역, 매개변수
- 소멸 : **함수나 제어문의 영역{ }를 벗어났을때** 메모리(스택)에서 해제(삭제)

<h3>지역변수의 유효 범위</h3>

<script>

```
function oneUp(){  
    let x = 1; //x는 지역 변수  
    x = x + 1;  
    return x;  
}
```

scope_local.html

```
let num = oneUp(); //호출되면 x는 2를 반환하고 소멸함
```

```
document.write("num = " + num); //2
```

```
document.write("x = " + x); //오류
```

</script>

전역 변수와 지역 변수

● 전역 변수(global variable)의 유효 범위

- 생성 : 메인 영역에서 생성하며 전체에 영향을 미치며, 값을 공유한다.
- 소멸 : 프로그램이 종료될때 메모리(전역공간)에서 해제

<h3>전역 변수(Global letiable)의 유효 범위</h3>

<script>

let x = 1; //전역 변수

function oneUp(){

 x += 1;

 return x;

}

oneUp(); //1번 호출

document.write("x = " + x + "
");

oneUp(); //2번 호출

document.write("x = " + x + "
");

oneUp(); //3번 호출

document.write("x = " + x + "
");

</script>

scope_global.html

객체의 정의

● 객체(Object)란?

- 프로그램에서 인식할 수 있는 모든 대상
- 데이터를 저장하고 처리하는 기본 단위.

◎ 자바 스크립트 내장 객체

문서 객체 모델(DOM) : 문서 뿐만 아니라 웹 문서 안에 포함된 이미지, 링크, 텍스트 필드 등을 모두 별도의 객체로 관리

브라우저 객체 모델(BOM) : 웹 브라우저 정보를 객체로 관리

◎ 사용자 정의 객체

여러 가지 자료형을 포함하는 '**복합**' **자료형**을 직접 만들어 사용하는 것을 말한다.
객체 지향 언어에서는 클래스(class)라고도 하며 객체의 설계도, 틀 역할을 한다.

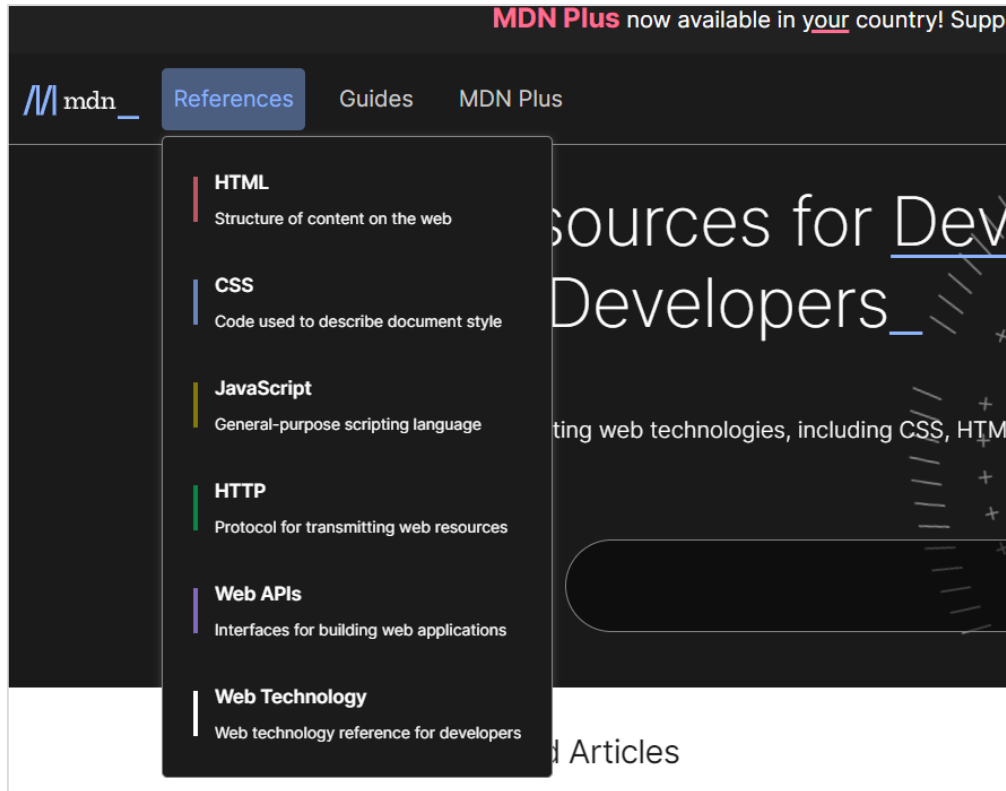
내장 객체 - Math

Math - 수학, 통계 관련 함수들을 내장하고 있음

함수	설명
abs(x)	숫자의 절대값을 반환합니다.
round(x)	숫자의 소수점 이하를 반올림합니다.(정수로 반환)
floor(x)	숫자의 소수점 이하를 버립니다.(정수로 반환)
pow(x, y)	거듭제곱수 구하기 - x의 y제곱을 반환합니다.
random()	0과 1사이의 무작위 수(난수)를 반환합니다.

내장 객체 - Math

MDN > Reference > JavaScript > Built-in objects > Math



▼ Built-in objects

[AggregateError](#)

[Array](#)

[ArrayBuffer](#)

[AsyncFunction](#)

[Atomics](#)

[BigInt](#)

[BigInt64Array](#)

[BigUint64Array](#)

[Boolean](#)

[DataView](#)

[Date](#)

내장 객체 - Math

MDN > Technology > JavaScript > Built-in objects > Math

`Math.pow(x, y)`

x의 y 제곱을 반환합니다.

`Math.random()`

0과 1 사이의 난수를 반환합니다.

`Math.round(x)`

숫자에서 가장 가까운 정수를 반환합니다.

```
Math.round( 20.49); // 20
```

```
Math.round( 20.5 ); // 21
```

```
Math.round( 42   ); // 42
```

```
Math.round(-20.5 ); // -20
```

```
Math.round(-20.51); // -21
```

```
Math.pow(7, 2);    // 49
```

```
Math.pow(7, 3);    // 343
```

```
Math.pow(2, 10);   // 1024
```

내장 객체 - Math

■ Console에서 실행하기

```
> Math.pi
```

```
< undefined
```

```
> Math.PI
```

```
< 3.141592653589793
```

```
> Math.round(2.78)
```

```
< 3
```

```
> Math.round(2.18)
```

```
< 2
```

```
> Math.ceil(2.18)
```

```
< 3
```

```
> Math.floor(5.4)
```

```
< 5
```

```
> Math.random()
```

```
< 0.7561141745190259
```

```
> Math.random() * 6 + 1
```

```
< 3.9185010260524082
```

```
> Math.floor(Math.random()*6) + 1
```

```
< 3
```

```
> Math.floor(Math.random()*6) + 1
```

```
< 6
```


내장 객체 - Math

- 절대값과 거듭 제곱함수 만들기

```
function myAbs(x){  
    if(x < 0)  
        return -x;  
    else  
        return x;  
}  
//x=-5, -(-5), 5  
//x=5, 5  
  
function myPow(x, y){ //x:밑, y:지수  
    var num = 1; //결과값  
    for(var i = 0; i < y; i++){  
        num *= x;  
    }  
    return num;  
}  
// i=0, 0<3, 1*2  
// i=1, 1<3, 2*2  
// i=2, 2<3, 4*2(2*2*2)
```

math-ex.html

```
//내장 객체와 비교하기  
document.write(Math.abs(-5) + "<br>");  
document.write(myAbs(-5)+ "<br>");  
  
document.write(Math.pow(2, 3)+ "<br>");  
document.write(myPow(2, 3)+ "<br>");
```

내장 객체 - Math

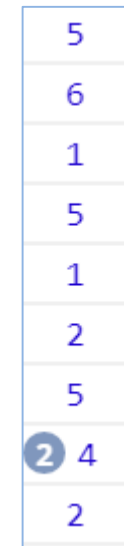
- 무작위수 만들기 - **Math.floor(Math.random())**

```
<script>
  //1~10 자연수중 무작위수
  var rand = Math.floor(Math.random()*10)+1;
  console.log(rand);

  //주사위 10번 던지기
  var dice = function(){
    return Math.floor(Math.random()*6)+1;
  }

  for(var i=1; i<=10; i++){
    console.log(dice());
  }
</script>
```

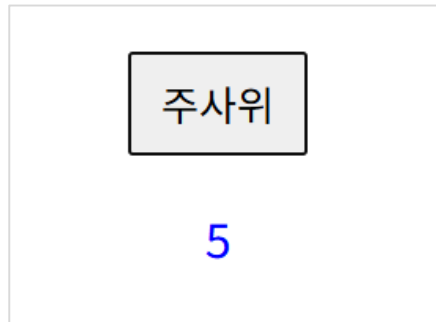
math-random.html



5
6
1
5
1
2
5
2
4
2

내장 객체 - Math

- 버튼 눌러 주사위 던지기



```
body{text-align: center;}
button{
  margin-top: 30px;
  padding: 10px 15px;
  font-size: 1.3rem;
}
p{font-size: 1.5rem; color: blue};
```

```
<button onclick="throwDice()">주사위</button>
<p id="demo"></p>

<script>
  function throwDice(){
    let dice = Math.floor(Math.random() * 6) + 1;
    const demo = document.querySelector("#demo");

    demo.textContent = dice;
  }
</script>
```

Math 객체 실습 문제

- ✓ 좋은 글귀를 배열에 저장해 두고 무작위로 출력하는 프로그램

성공하는 사람은 실패하는데 익숙한 사람이다.

```
// 좋은 글귀
var words = []

words[0] = "당신은 지금도 최고고, 이전에도 최고였으며 앞으로도 최고일 것이다";
words[1] = "성공하는 사람은 실패하는데 익숙한 사람이다.";
words[2] = "후회를 최대한 이용하라. 깊이 후회한다는 것은 새로운 삶을 산다는 것이다.";
words[3] = "가짜 친구는 소문을 믿고, 진짜 친구는 나를 믿는다.";
words[4] = "성공이라는 뜻을 박으려면 끈질김이라는 망치가 필요하다.";

var rand = Math.floor(Math.random()*words.length);

document.write(words[rand]);
```

Math 실습 문제

good-word.html

```
<style>
  body{
    text-align: center;
    margin-top: 40px;
    font-size: 2.5em;
    background-color: ■rgb(75, 1, 144);
    color: □#eee;
  }
</style>
<script src="js/word.js"></script>
```

내장 객체 - Date

Date 객체

- 날짜와 시간 정보를 다루는 객체
- **const date = new Date()**로 인스턴스를 만듦.

함수	설명
getFullYear()	연도를 4자리 숫자로 표시함
getMonth()	0부터 11사이의 숫자로 월을 표시함(0- 1월)
getDate()	1~31 사이의 숫자로 일을 표시함
getDay()	0~6 사이의 숫자로 요일을 표시함(0-일요일)
getHours()	0~23 사이의 숫자로 시를 표시함
getMinutes()	0~59 사이의 숫자로 분을 표시함
getSeconds()	0~59 사이의 숫자로 초를 표시함
getTime()	1970년 1월 1일 이후의 시간을 밀리 초로 표시함

내장 객체 - Date

Date 객체의 날짜/시간 함수

```
> var now = new Date()  
< undefined  
  
> now  
< Sun Jun 13 2021 18:47:13 GMT+0900 (대한민국 표준시)  
  
> now.toLocaleString()  
< "2021. 6. 13. 오후 6:47:13"  
  
> now.getFullYear()  
< 121  
  
> now.getFullYear()  
< 2021  
  
> now.getMonth()  
< 5  
  
> now.getDate()  
< 13  
  
> now.getDay()  
< 0
```

내장 객체 - Date

Date 객체의 날짜/시간 함수

날짜/시간 표시하기

현재 년도: 2025

현재 월: 9

현재 일: 30

현재 시: 7

현재 분: 50

현재 초: 30

현재 요일: 2

현재까지 시간(초): 1759186230.341

현재 날짜: 2025. 9. 30.

현재 시간: 오전 7:50:30

내장 객체 - Date

Date 객체의 날짜/시간 함수

```
<h2>날짜/시간 표시하기</h2>
<p id="demo"></p>

<script>
  const now = new Date();

  let demo = document.getElementById("demo");
  demo.innerHTML += `현재 년도: ${now.getFullYear()}<br>`;
  demo.innerHTML += `현재 월: ${now.getMonth() + 1}<br>`;
  demo.innerHTML += `현재 일: ${now.getDate()}<br>`;
  demo.innerHTML += `현재 시: ${now.getHours()}<br>`;
  demo.innerHTML += `현재 분: ${now.getMinutes()}<br>`;
  demo.innerHTML += `현재 초: ${now.getSeconds()}<br>`;
  demo.innerHTML += `현재 요일: ${now.getDay()}<br>`;
  demo.innerHTML += `현재까지 시간(초): ${now.getTime()/1000}<br>`;

  demo.innerHTML += `현재 날짜: ${now.toLocaleDateString()}<br>`;
  demo.innerHTML += `현재 시간: ${now.toLocaleTimeString()}<br>`;
</script>
```

날짜 계산 프로그램

날짜 계산 프로그램 만들기

지금까지 몇 일?

개강 이후 **17**일 지났습니다.

날짜 계산 프로그램

날짜 계산 프로그램 만들기

```
<div id="container">
  <h2>지금까지 몇 일?</h2>
  <p>개강 이후 <span id="day" class="accent"></span>일 지났습니다.</p>
  <p>축하합니다.</p>
</div>
```

passedtime.html

```
<script>
  let now = new Date(); //날짜 객체 생성
  let firstDay = new Date("2022-5-9");

  let passedTime = now.getTime() - firstDay.getTime(); //지난 시간 계산(밀리초)
  console.log(passedTime + "ms");

  passedTime = Math.round(passedTime/(24*60*60*1000));
  //밀리초(ms) ->초(s)로 환산후 일(day)로 환산.. 24시간 60분 60초 1000밀리초

  document.querySelector("#day").innerText = passedTime;
  //document.getElementById("day").innerHTML = passedTime;
</script>
```

날짜 계산 프로그램

날짜 계산 프로그램 만들기

```
#container{  
  width: 300px;  
  height: 300px;  
  margin: 50px auto;  
  border: 2px solid ■ #222;  
  border-radius: 50%;  
  background-color: □ aliceblue;  
  text-align: center;  
  position: relative;  
}  
h2{position: absolute; left: 60px; top: 60px;}  
p{position: absolute; left: 50px; top: 120px;}  
.accent{color: ■ red; font-size: 1.5rem; font-weight: bold;}
```

내장 객체 – Array

● indexOf() 함수 예제

127.0.0.1:5500 내용:

입력해주세요

127.0.0.1:5500 내용:

안녕하세요

단어를 검색하면 대답하는 프로그램

- '안녕'이 있으면 '안녕하세요' 출력
- 시간을 검색하면 '현재 시간' 출력
- '잘있어' 또는 '잘가'를 검색하면
"안녕히 가세요"를 출력하고 프로그램
이 종료됨
- 찾는 단어가 없으면 "모르는 단어입니다" 출력

내장 객체 - Array

● indexOf() 함수 예제

```
while(true){
    let input = prompt("단어를 입력해 주세요");

    if(input.indexOf("안녕") >= 0){
        alert("안녕하세요");
    }
    else if(input.indexOf("시") >= 0 || input.indexOf("분") >= 0){
        const now = new Date();
        let hours = now.getHours();
        let minutes = now.getMinutes();
        alert("현재 시각은 " + hours + "시 " + minutes + "분 입니다.");
    }
    else if(input.indexOf("잘 가") >= 0 || input.indexOf("잘 있어") >= 0){
        alert("안녕히 가세요.");
        break;
    }
    else{
        alert("모르는 단어입니다.");
    }
}
```

indexof2.html

내장 객체 - window

- window 객체 - setInterval() 함수
 - setInterval(in milliseconds) : 초가 설정되고 계속 반복함.

```
<h3>3초 후에 알림창이 뜨고, 1초에 한번 계속 창이 뜹니다.</h3>
<script>
    setInterval( //실행 함수
    | function(){
    |     alert("안녕하세요~")
    | }, 3000
    );

    /*
    setInterval(winHello, 3000); //일반함수 정의

    function winHello(){
    |     alert("Hello~ ");
    | }*/
</script>
```

디지털 시계

- 시계 표시하기

디지털 시계

오전 7:33:58

```
<div class="content">  
  <h2>디지털 시계</h2>  
  <p id="display"></p>  
</div>  
  
<script src="js/watch.js"></script>
```


디지털 시계

- 시계 표시하기

```
const display = document.getElementById("display");

//1초 간격으로 myWatch 호출
/*const watch = setInterval(myWatch, 1000);

function myWatch(){
  const date = new Date();
  let time = date.toLocaleTimeString();
  display.textContent = time;
}*/

//실행 함수
setInterval(() => {
  const date = new Date();
  let time = date.toLocaleTimeString();
  display.textContent = time;
},1000);
```

내장 객체 - window

- window 객체 - setTimeout() 함수
 - setTimeout() : 초가 1번 설정되고 종료함

```
<h3>3초 후에 알림창이 뜨고, 확인 하면 종료합니다.</h3>
<script>
  setTimeout(
    function(){
      alert("환영합니다.")
    }, 3000
  );
</script>
```

내장 객체 - window

- window 객체 - setTimeout() 함수



```
<h2>클릭 버튼을 누른후 찻잔을 클릭하세요.</h2>
<button onclick="welcome()">클릭</button>
<div>
  
</div>

<script src="js/settime.js"></script>
```

내장 객체 - window

- window 객체 - setTimeout() 함수

```
function welcome(){  
  setTimeout(() => {  
    alert("환영합니다.")  
  }, 1000);  
}  
  
function changePic(){  
  let img = document.getElementById("pic");  
  
  setTimeout(() => {  
    img.src = "images/cup-2.png";  
  }, 1000);  
}
```

settime.js

인사말 반복하기

- 인사말 반복하기



```
// 2초마다 인사말 변경
let message = ["안녕~", "잘 지내니?", "좋은 하루!", ];
let msgIdx = 0;
const greet = document.getElementById("greet");

// 2초마다 인사말 변경
setInterval(() => {
  greet.textContent = message[msgIdx];
  msgIdx = (msgIdx + 1) % message.length; // 순환
}, 2000);
```

객체의 정의

● 사용자 정의 객체

- 여러가지 자료형을 포함하는 '복합' 자료형을 말한다.
- 객체는 속성과 메서드로 구성되어 있다.

속성(property)

나이 : 26
이름 : 이정후
결혼유무 : false

메서드(function)

야구를 한다.

사람



```
let 객체이름 = {
```

```
  속성 이름 : 값,  
  속성 이름 : 값,  
  함수 이름 : function(){...}  
}
```

콜론(:)사용

coma(,)사용

키(key) : 값(value) 형태

객체의 정의 및 사용

- 사용자 정의 객체(object)

Person 객체

이름: 이정후
나이: 26
결혼여부: false
이정후이(가) 야구를 합니다.

```
<h3>Person 객체</h3>
<div id="demo"></div>

<script>
  let person = {
    name: "이정후",
    age: 26,
    isMerried: false,

    play() {
      return `${this.name}이(가) 야구를 합니다.`;
    }
  };
};
```

객체의 정의 및 사용

- 사용자 정의 객체(object)

```
console.log(person);
console.log(person.name);

const demo = document.getElementById("demo");
demo.innerHTML = `
  이름: ${person.name} <br>
  나이: ${person.age} <br>
  결혼여부: ${person.isMerried ? "기혼" : "미혼"} <br>
  ${person.play()} <br>
`;

</script>
```


객체의 정의 및 사용

- 여러 명의 객체(object)

Person 객체

이정후

나이: 26

결혼여부: 미혼

이정후이(가) 야구를 합니다.

김연아

나이: 34

결혼여부: 기혼

김연아이(가) 피겨 스케이팅을 합니다.

손흥민

나이: 32

결혼여부: 미혼

손흥민이(가) 축구를 합니다.

객체의 정의 및 사용

- 여러 명의 객체(object)

```
// 여러 명의 person 객체를 배열로 저장
let people = [
  {
    name: "이정후",
    age: 26,
    isMarried: false,
    play() {
      return `${this.name}이(가) 야구를 합니다.`;
    }
  },
  {
    name: "김연아",
    age: 34,
    isMarried: true,
    play() {
      return `${this.name}이(가) 피겨 스케이팅을 합니다.`;
    }
  },
]
```

객체의 정의 및 사용

- 여러 명의 객체(object)

```
{
  name: "손흥민",
  age: 32,
  isMarried: false,
  play() {
    return `${this.name}이(가) 축구를 합니다.`;
  }
}

];

const demo = document.getElementById("demo");

// 배열 반복 출력
people.forEach(person => {
  demo.innerHTML += `
    <h4>${person.name}</h4>
    나이: ${person.age} <br>
    결혼여부: ${person.isMarried ? "기혼" : "미혼"} <br>
    ${person.play()}
  </div>
  `;
});
```

클래스(class)

- 클래스(class)란?

- 객체(object)를 생성하기 위한 템플릿(틀)이다.
- 구성: 생성자(constructor), 멤버 변수, 메서드
- 멤버 변수 초기화 **this** 키워드 사용

속성(property)

name, age

생성자

constructor(name, age)

메서드(function)

sayHello()

클래스(class)

- 클래스 정의와 사용

```
// 클래스 정의
class Person {
  constructor(name, age) {    // 생성자 (객체 초기화)
    this.name = name;
    this.age = age;
  }

  // 메서드
  sayHello() {
    return `안녕하세요, 저는 ${this.name}이고 나이는 ${this.age}살 입니다.`;
  }
}
```

```
//클래스 사용
let p1 = new Person("우상혁", 29); // 인스턴스 생성 (person 객체)
console.log(p1.sayHello());

let p2 = new Person("신유빈", 21);
console.log(p2.sayHello());
```

클래스(class)

- Person 클래스

```
class Person {  
  constructor(name, age, isMerried) {  
    this.name = name;  
    this.age = age;  
    this.isMerried = isMerried;  
  }  
  
  play() {  
    return `${this.name}이(가) 야구를 합니다.`;  
  }  
}
```

클래스(class)

- Person 클래스

```
// person 객체 생성
let person = new Person("이정후", 26, false);

const demo = document.getElementById("demo");
demo.innerHTML = `
  이름: ${person.name} <br>
  나이: ${person.age} <br>
  결혼여부: ${person.isMerried ? "기혼" : "미혼"} <br>
  ${person.play()} <br>
`;
```