

HTML



Markup Language
Content

CSS



Style sheet Language
Presentation

JS



Programming Language
Behavior

자바스크립트 기본 문법 2

함수

- 특정 작업을 수행하기 위해 설계된 코드 블록
- 자바스크립트에는 이미 여러 함수가 만들어져 있어서 사용할 수 있음
 - 예) alert()

- 함수 선언 (함수 정의)

- 함수가 어떤 명령을 처리해야 할지 미리 알려주는 것
- function 예약어를 사용하고, { } 안에 실행할 명령을 작성

```
function name(parameter1, parameter2, parameter3) {  
    // code to be executed  
}
```

```
function myFunction(a, b) {  
    return a * b;  
}
```

- 함수 호출 (함수 실행)

- 함수 이름을 사용해 함수 실행

```
함수명(전달할 변수들);
```

```
let rs = myFunction(4, 6); // 24
```

Scope

■ 전역 변수

- 자바스크립트 코드 내의 어느곳에서나 접근할 수 있는 변수
- 방법
 - 선언하지 않고 사용
 - var로 선언 시 함수 밖에서 선언된 변수
 - let으로 선언 시 블록 밖에서 선언된 변수

■ 지역 변수

- 블록 내에서만 사용할 수 있는 변수
- 방법
 - var: 함수내에서 선언시 선언된 함수 내에서만 유효
 - let: 블록내에서 선언시 선언된 블록 내에서만 유효

호이스팅(hoisting)

- 변수를 뒤에서 선언하지만, 마치 앞에서 미리 선언한 것처럼 인식
- 함수 실행문을 앞에 두고 함수 정의 부분을 뒤에 두더라도 앞으로 끌어올려 인식
- var로 선언된 변수는 호이스팅 가능, let으로 선언된 변수는 호이스팅 불가
- 그러나 var는 재선언이 가능한 변수로 실수로 변수를 잘못 조작할 확률이 높아지므로 권장하지 않음

```
<script>
  function displayNumber(){
    var x = 10;
    let a = 10;
    console.log(x);
    console.log(y);
    console.log(a);
    console.log(b);
    var y = 20;
    let b = 20;
  }
  displayNumber();
</script>
```

10

undefined

10

✖ ▶ Uncaught ReferenceError: Cannot access 'b' before initialization
at displayNumber (ex_hoist.html:16:25)
at ex_hoist.html:20:9

매개변수와 반환 값

```
function functionName(parameter1, parameter2, parameter3) {  
    // code to be executed  
    return value;  
}
```

■ 매개변수(parameter)

- 함수 정의부에 열거된 변수명. 함수의 입력으로 들어오는 변수들

■ 인수(arguments)

- 함수를 실행할 때 매개 변수 자리에 넘겨주는 값. 호출 시 실제 전달되는 값

■ 반환 값(return)

- 함수를 실행한 결과값을 함수를 호출한 쪽으로 넘겨줌

디폴트 매개변수

- 함수 정의부에 선언된 매개변수보다 적은 수의 인수가 전달되면, 전달값이 없는 매개변수에 undefined 가 설정

```
function myFunction(x, y) {  
  if (y === undefined) {  
    y = 2;  
  }  
}
```

- 디폴트 매개변수 값 설정

```
function multiple(a, b = 5, c = 10){ // b = 5, c = 10 으로 기본값 지정  
  return a * b + c;  
}  
var result1 = multiple(5, 10, 20); // a = 5, b = 10, c = 20  
var result2 = multiple(10, 20);    // a = 10, b = 20, c = 10  
var result3 = multiple(30);        // a = 30, b = 5, c = 10
```

가변길이 매개변수

```
function sum(...args) {  
  let sum = 0;  
  for (let arg of args) sum += arg;  
  return sum;  
}  
  
let x = sum(4, 9, 16, 25, 29, 100, 66, 77);
```

익명 함수

- 함수 이름이 없는 함수
- 함수 자체가 식이므로 함수를 변수에 할당할 수도 있고 다른 함수의 매개변수로 사용할 수도 있음
- 변수에 저장된 익명 함수는 함수 이름 대신 변수를 이용해 함수를 실행함

```
const x = function (a, b) {return a * b};  
let z = x(4, 3);
```


즉시 실행 함수(Self-Invoking Functions)

- 함수를 정의함과 동시에 실행하는 함수(한 번만 실행하는 함수)
 - 함수를 실행하는 순간 자바스크립트 해석기에서 함수를 해석함
- 식 형태로 선언하기 때문에 함수 선언 끝에 세미콜론(;) 붙임

```
(function () {  
    let x = "Hello!!"; // I will invoke myself  
})();
```

```
(function(a, b){  
    sum = a + b;  
})(100, 200));
```

화살표 함수

- ES6 이후 사용 => 표기를 이용해 함수 식을 작성하는 간단한 문법
- function 키워드 생략
- 문장이 한개밖에 없을 경우 return, 중괄호 생략 가능

```
// ES5  
var x = function(x, y) {  
    return x * y;  
}
```



```
const x = (x, y) => { return x * y };
```



```
const x = (x, y) => x * y;
```

콜백함수

■ 다른 함수에 인수로 전달되는 함수

```
function myDisplayer(some) {  
  document.getElementById("demo").innerHTML = some;  
}
```

```
function myCalculator(num1, num2, myCallback) {  
  let sum = num1 + num2;  
  myCallback(sum);  
}
```

```
myCalculator(5, 5, myDisplayer);
```

- 함수가 인수로 전달될 때 소괄호 사용안함

콜백함수 예

```
// Create an Array
const myNumbers = [4, 1, -20, -7, 5, 9, -6];

// Call removeNeg with a callback
const posNumbers = removeNeg(myNumbers, (x) => x >= 0);

// Display Result
document.getElementById("demo").innerHTML = posNumbers;

// Keep only positive numbers
function removeNeg(numbers, callback) {
  const myArray = [];
  for (const x of numbers) {
    if (callback(x)) {
      myArray.push(x);
    }
  }
  return myArray;
}
```

비동기 함수

- 다른 함수와 병렬로 실행되는 함수

- setTimeout()

- 시간 초과 시 실행될 콜백 함수 지정

```
setTimeout(myFunction, 3000);  
function myFunction() {  
    document.getElementById("demo").innerHTML = "I love You !!";  
}
```

- setInterval()

- 간격마다 실행될 콜백 함수 지정

```
setInterval(myFunction, 1000);  
function myFunction() {  
    let d = new Date();  
    document.getElementById("demo").innerHTML =  
        d.getHours() + ":" +  
        d.getMinutes() + ":" +  
        d.getSeconds();  
}
```

최근엔 비동기 프로그래밍을
위해 콜백대신 promise 사용

```
let myPromise = new Promise(function(myResolve, myReject) {  
  // "Producing Code" (May take some time)
```

```
  myResolve(); // when successful  
  myReject();  // when error  
});  
// "Consuming Code" (Must wait for a fulfilled Promise)  
myPromise.then(  
  function(value) { /* code if successful */ },  
  function(error) { /* code if some error */ }  
);
```

Promise Object

■ Syntax

```
function myDisplayer(some) {  
  document.getElementById("demo").innerHTML = some;  
}
```

```
let myPromise = new Promise(function(myResolve, myReject) {  
  let x = 0;  
  // The producing code (this may take some time)  
  if (x == 0) {  
    myResolve("OK");  
  } else {  
    myReject("Error");  
  }  
});
```

```
myPromise.then(  
  function(value) {myDisplayer(value);},  
  function(error) {myDisplayer(error);}  
);
```

Callback vs Promise

■ Example Using Callback

```
setTimeout(function() { myFunction("I love You !!!"); }, 3000);

function myFunction(value) {
  document.getElementById("demo").innerHTML = value;
}
```

■ Example Using Promise

```
let myPromise = new Promise(function(myResolve, myReject) {
  setTimeout(function() { myResolve("I love You !!"); }, 3000);
});

myPromise.then(function(value) {
  document.getElementById("demo").innerHTML = value;
});
```

async, await

```
function myFunction() {  
  return Promise.resolve("Hello");  
}
```

■ async

- Promise 객체를 반환하는 함수 작성하는 키워드

```
async function myFunction() {  
  return "Hello";  
}  
myFunction().then(  
  function(value) {myDisplayer(value);}  
);
```

■ await

- async 함수 내에서 사용

```
async function myDisplay() {  
  let myPromise = new Promise(function(resolve) {  
    resolve("I love You !!");  
  });  
  document.getElementById("demo").innerHTML = await myPromise;  
}  
myDisplay();
```

```
async function myDisplay() {  
  let myPromise = new Promise(function(resolve) {  
    setTimeout(function() {resolve("I love You !!");}, 3000);  
  });  
  document.getElementById("demo").innerHTML = await myPromise;  
}  
myDisplay();
```


객체

■ 객체(object)란

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

■ 자바스크립트 객체


- 자바스크립트 안에 미리 객체로 정의해 놓은 것
- 문서 객체 모델(DOM) : 문서 뿐만 아니라 웹 문서 안에 포함된 이미지·링크·텍스트 필드 등을 모두 별도의 객체로 관리
- 브라우저 관련 객체 : 웹 브라우저 정보를 객체로 관리
- 내장 객체 : 웹 프로그래밍에서 자주 사용하는 요소를 객체로 정의해 놓음.

■ 사용자 정의 객체

- 필요할 때마다 사용자가 직접 만드는 객체

객체(Objects)

■ 실세계에서의 자동차 객체 사례

Object	Properties	Methods
	car.name = Fiat car.model = 500 car.weight = 850kg car.color = white	car.start() car.drive() car.brake() car.stop()

```
const person = {  
  firstName: "John",  
  lastName: "Doe",  
  age: 50,  
  eyeColor: "blue"  
};
```

```
const car = {type:"Fiat", model:"500", color:"white"};
```

```
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

■ 객체 속성 접근 방법

`objectName.propertyName`

or

`objectName["propertyName"]`

객체 메소드(Objects Methods)

- 함수 정의를 포함하는 객체의 속성

```
const person = {  
  firstName: "John",  
  lastName: "Doe",  
  id: 5566,  
  fullName: function() {  
    return this.firstName + " " + this.lastName;  
  }  
};
```

- 객체 메소드 사용 방법

```
objectName.methodName()
```

```
name = person.fullName();
```

객체 출력

■ 반복문에서 객체 출력

```
const person = {  
  name: "John",  
  age: 30,  
  city: "New York"  
};  
  
let txt = "";  
for (let x in person) {  
  txt += person[x] + " ";  
};
```

■ Object.values()

- 객체를 배열로 반환

```
const person = {  
  name: "John",  
  age: 30,  
  city: "New York"  
};  
const myArray = Object.values(person);
```

■ JSON.stringify()

- 객체를 JSON 형식의 문자열로 변환

```
const person = {  
  name: "John",  
  age: 30,  
  city: "New York"  
};  
let myString = JSON.stringify(person);
```

배열 (Array)

- 배열 생성, 요소 접근

```
const cars = ["Saab", "Volvo", "BMW"];  
let car = cars[0];
```

- length : 배열의 길이 반환

```
let size = fruits.length;
```

- sort() : 정렬

- reverse() 배열의 요소를 역순으로 반환

- Array.forEach() 함수 사용

- 배열의 모든 요소에 주어진 함수 적용

```
const fruits =  
["Banana", "Orange", "Apple", "Mango"];  
let text = "<ul>";  
fruits.forEach(myFunction);  
text += "</ul>";  
function myFunction(value) {  
    text += "<li>" + value + "</li>";  
}
```

- map(): 배열 각 요소에 함수 적용 결과값을 새로운 배열로 반환
- filter(): 배열 각 요소에 함수 적용 결과값이 참인 요소들만 새로운 배열로 반환
- reduce(): 배열 각 요소에 함수 적용 → 하나의 결과값 반환

배열 객체 주요 메소드 (계속)

■ toString() : 문자열 반환

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
document.getElementById("demo").innerHTML = fruits.toString();
```

Banana,Orange,Apple,Mango

■ at()

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
let fruit = fruits[2];  
let fruit = fruits.at(2);
```

■ join() : 구분자와 함께 연결된 문자열 반환

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
document.getElementById("demo").innerHTML = fruits.join(" * ");
```

Banana * Orange * Apple * Mango

배열 객체 주요 메소드 (계속)

- `pop()` : 마지막 요소 제거/반환
- `push()`: 마지막 요소 추가

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
let fruit = fruits.pop();  
fruits.push("Kiwi");
```

- `shift()`: 첫번째 요소 제거/반환
- `unshift()`: 첫번째 요소 추가

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
let fruit = fruits.shift();
```

배열 객체 주요 메소드 (계속)

■ concat()

- 서로 다른 배열 2개를 합쳐서 새로운 배열을 만듦

```
const myGirls = ["Cecilie", "Lone"];  
const myBoys = ["Emil", "Tobias", "Linus"];  
const myChildren = myGirls.concat(myBoys);
```

■ splice(추가위치, 삭제되는 요소 수, 새롭게 추가될 요소들)

- 배열 중간에 새로운 요소를 추가하거나 삭제
- 삭제된 아이템들 반환

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
fruits.splice(2, 0, "Lemon", "Kiwi");
```

■ slice()

- 원본 배열은 그대로 두고, 주어진 인덱스 범위의 요소들로 구성된 새로운 배열을 반환

```
const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];  
const citrus = fruits.slice(1);  
const citrus2 = fruits.slice(1, 3);
```


■ 생성

```
const d = new Date();
```

```
const d = new Date("2022-03-25");
```

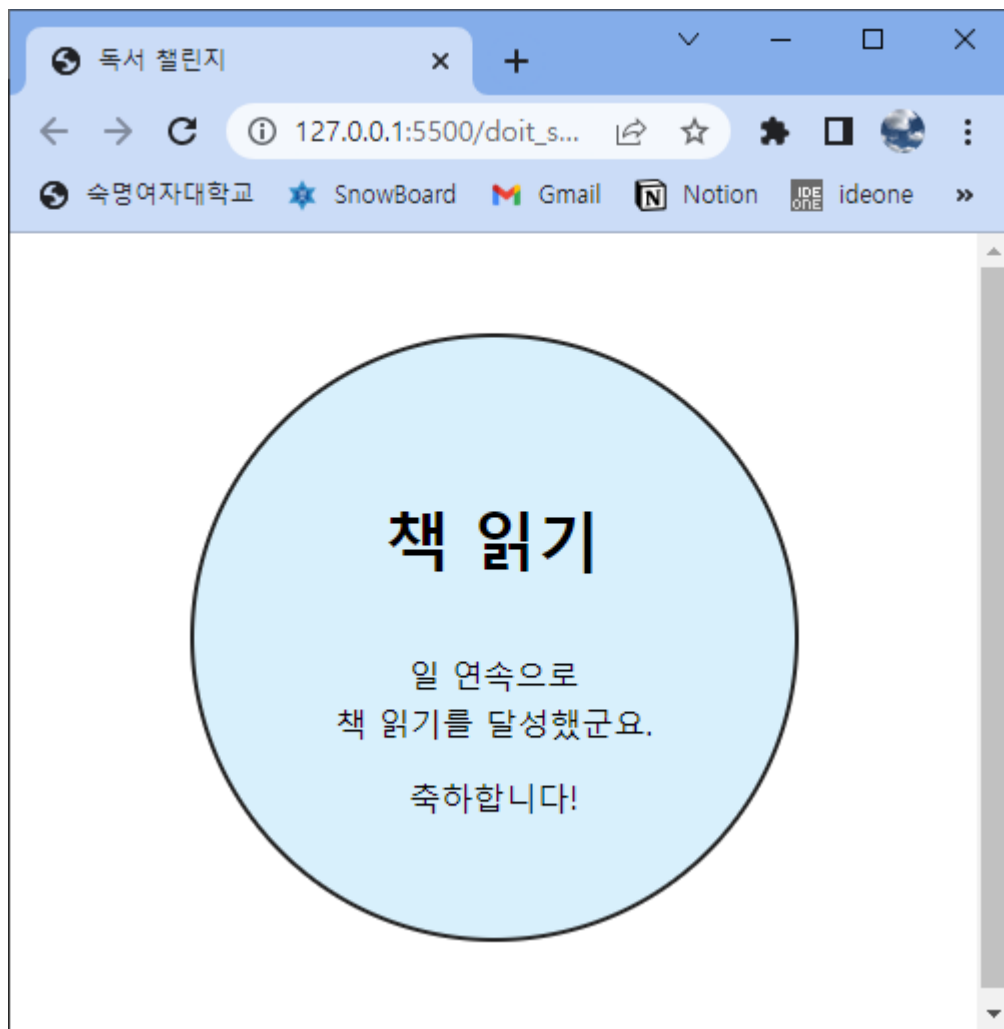
■ 날짜 출력 문자열 메소드

- toString() Sat Apr 20 2024 13:00:22 GMT+0900 (한국 표준시)
- toDateString() Sat Apr 20 2024
- toUTCString() Sat, 20 Apr 2024 04:00:47 GMT
- toISOString() 2024-04-20T04:01:30.984Z

Date 객체 주요 메소드

Method	Description	Method	Description
getFullYear()	Get year as a four digit number (yyyy)	setDate()	Set the day as a number (1-31)
getMonth()	Get month as a number (0-11)	setFullYear()	Set the year (optionally month and day)
getDate()	Get day as a number (1-31)	setHours()	Set the hour (0-23)
getDay()	Get weekday as a number (0-6)	setMilliseconds()	Set the milliseconds (0-999)
getHours()	Get hour (0-23)	setMinutes()	Set the minutes (0-59)
getMinutes()	Get minute (0-59)	setMonth()	Set the month (0-11)
getSeconds()	Get second (0-59)	setSeconds()	Set the seconds (0-59)
getMilliseconds()	Get millisecond (0-999)	setTime()	Set the time (milliseconds since January 1, 1970)
getTime()	Get time (milliseconds since January 1, 1970)		

날짜 계산 프로그램



```
<style>
  #container{
    margin:50px auto;
    width:300px;
    height:300px;
    border-radius:50%;
    border:2px double ■ #222;
    background-color:□ #d8f0fc;
    text-align: center;
  }
  h1 {
    margin-top:80px;
  }
  .accent {
    font-size:1.8em;
    font-weight:bold;
    color:■ red;
  }
</style>
</head>
<body>
  <div id="container">
    <h1>책 읽기</h1>
    <p><span class="accent" id="result"></span>
      일 연속으로 <br> 책 읽기를 달성했군요.</p>
    <p>축하합니다!</p>
  </div>
```

```

<div id="container">
  <h1>책 읽기</h1>
  <p><span class="accent" id="result"></span>일 연속으로 <br> 책 읽기를 달성했군요.</p>
  <p>축하합니다!</p>
</div>

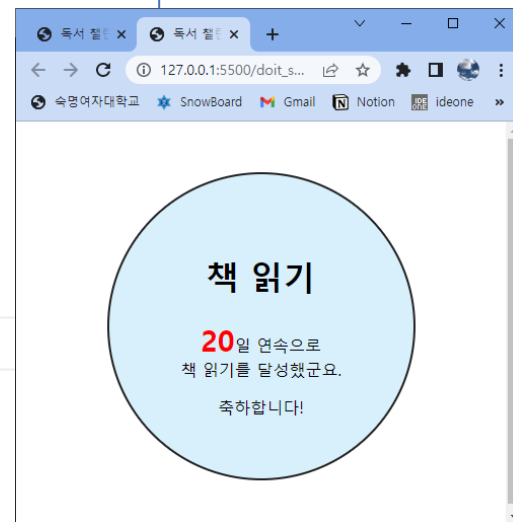
<script>
  var now = new Date();          // 오늘 날짜를 객체로 지정
  var firstDay = new Date("2023-04-01"); // 시작 날짜를 객체로 지정

  var toNow = now.getTime();      // 오늘까지 지난 시간(밀리 초)
  var toFirst = firstDay.getTime(); // 첫날까지 지난 시간(밀리 초)
  var passedTime = toNow - toFirst; // 첫날부터 오늘까지 지난 시간(밀리 초)

  passedTime = Math.round(passedTime/(1000*60*60*24)); // 밀리 초를 일 수로 계산하고 반올림

  document.querySelector('#result').innerText = passedTime;
</script>

```



Math 객체

■ Math 객체의 특징

- 수학 계산과 관련된 메서드가 많이 포함되어 있지만 수학식에서만 사용하는 것은 아님.
- 무작위 수가 필요하거나 반올림이 필요한 프로그램 등에서도 Math 객체의 메서드 사용함
- Math 객체는 인스턴스를 만들지 않고 프로퍼티와 메서드 사용

■ Math 객체의 프로퍼티

```
Math.E           // returns Euler's number
Math.PI          // returns PI
Math.SQRT2       // returns the square root of 2
Math.SQRT1_2     // returns the square root of 1/2
Math.LN2         // returns the natural logarithm
of 2
Math.LN10        // returns the natural logarithm
of 10
Math.LOG2E       // returns base 2 logarithm of E
Math.LOG10E      // returns base 10 logarithm of E
```

Math 객체 주요 메소드

- 사용 형식: *Math.method(number)*

Method	Description
<u>abs(x)</u>	Returns the absolute value of x
<u>acos(x)</u>	Returns the arccosine of x, in radians
<u>acosh(x)</u>	Returns the hyperbolic arccosine of x
<u>asin(x)</u>	Returns the arcsine of x, in radians
<u>asinh(x)</u>	Returns the hyperbolic arcsine of x
<u>atan(x)</u>	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
<u>atan2(y, x)</u>	Returns the arctangent of the quotient of its arguments
<u>atanh(x)</u>	Returns the hyperbolic arctangent of x
<u>cbrt(x)</u>	Returns the cubic root of x
<u>ceil(x)</u>	Returns x, rounded upwards to the nearest integer
<u>cos(x)</u>	Returns the cosine of x (x is in radians)
<u>cosh(x)</u>	Returns the hyperbolic cosine of x
<u>exp(x)</u>	Returns the value of E^x
<u>floor(x)</u>	Returns x, rounded downwards to the nearest integer

Math 객체 주요 메소드(계속)

<u>log(x)</u>	Returns the natural logarithm (base E) of x
<u>max(x, y, z, ..., n)</u>	Returns the number with the highest value
<u>min(x, y, z, ..., n)</u>	Returns the number with the lowest value
<u>pow(x, y)</u>	Returns the value of x to the power of y
<u>random()</u>	Returns a random number between 0 and 1
<u>round(x)</u>	Rounds x to the nearest integer
<u>sign(x)</u>	Returns if x is negative, null or positive (-1, 0, 1)
<u>sin(x)</u>	Returns the sine of x (x is in radians)
<u>sinh(x)</u>	Returns the hyperbolic sine of x
<u>sqrt(x)</u>	Returns the square root of x
<u>tan(x)</u>	Returns the tangent of an angle
<u>tanh(x)</u>	Returns the hyperbolic tangent of a number
<u>trunc(x)</u>	Returns the integer part of a number (x)

Math 객체 메소드 사용 예제

- random()

```
// Returns a random integer from 0 to 9:  
Math.floor(Math.random() * 10);
```

```
// Returns a random integer from 1 to 100:  
Math.floor(Math.random() * 100) + 1;
```

- min에서 max(포함) 사이의 난수 반환하는 함수 예

```
function getRndInteger(min, max) {  
    return Math.floor(Math.random() * (max - min + 1) ) + min;  
}
```


이벤트 당첨자 뽑기 프로그램

- `Math.random()` 활용
 - 0에서 1 사이(1포함 하지 않음)의 랜덤 숫자 출력
- 1~100 사이의 무작위 수 출력하기
 - `Math.random() * 100 + 1`
- 소수점 이하의 수는 버리고 출력하기
 - `Math.floor(Math.random() * 100 + 1)`

```

<body>
  <h1>당첨자 발표</h1>
  <script>
    var seed = prompt("전체 응모자 수 : ","");
    var picked = Math.floor((Math.random() * seed) + 1);

    document.write("전체 응모자 수 : " + seed + "명");
    document.write("<br>");
    document.write("당첨자 : " + picked + "번");
  </script>
</body>

```

