자바스크립트(JavaScript)란?

자바스크립트(JavaScript)는 객체(object) 기반의 스크립트 언어입니다.
HTML로는 웹의 내용을 작성하고,
CSS로는 웹을 디자인하며,
자바스크립트로는 웹의 동작을 구현할 수 있습니다.

자바스크립트는 주로 웹 브라우저에서 사용되나, Node.js와 같은 프레임워크를 사용하면 서버 측 프로그래밍에서도 사용할 수 있습니다.
현재 컴퓨터나 스마트폰 등에 포함된 대부분의 웹 브라우저에는 자바스크립트 인터프리터가 내장되어 있습니다.

자바스크립트의 특징

1) 자바스크립트는 객체 기반의 스크립트 언어입니다.
2) 자바스크립트는 동적이며, 타입을 명시할 필요가 없는 인터프리터 언어입니다.
3) 자바스크립트는 객체 지향형 프로그래밍과 함수형 프로그래밍을 모두 표현할 수 있습니다.

객체, 속성, 메소드의 개념 이해

자바스크립트는 객체 지향적 언어입니다. 생활주변에서 볼 수 있는 사람, 동물, 자전거, 꽃 등 모든 사물이 객체입니다.

자바스크립트에서 객체(object)란 웹브라우저를 포함한 웹 문서의 모든 구성요소를 말합니다. 즉, 웹브라우저의 상태표시줄, 스크롤바, 웹문서 자체, 레이어, 하이퍼링크, 이미지, 폼버튼 등등 대부분의 구성요소를 객체라고 합니다.
객체들은 또 다른 하위 객체들을 가지고 있을 수 있습니다. 이런 이유로 자바스크립트의 객체구조를 계층적구조라고 말할 수 있습니다.

모든 객체는 속성(Property)을 가집니다.
window 객체는 frames(프레임), name(윈도우 이름), location(위치) 등의 속성을 가지며 이미지 객체는 가로크기, 세로크기등의 속성을 가집니다.

document.title="연습"; (객체.속성="속성값")

객체와 속성, 객체와 메소드를 연결 할 때에는 (.) 으로 연결합니다

메소드 이해하기

메소드(method)는 객체의 동작과 관련된 것입니다.
「토끼」라는 객체를 예를 들면,「빛깔이 희다」,「앞다리가 짧다」 등은 속성에 관한 사항이고,「뛰어간다」,「풀을 뜯어 먹는다」 등의 행동은 메소드에 해당합니다.
자바스크립트에서 open() 메소드는 팝업 윈도우를 열어주고, write() 메소드는 문자열을 출력하며 alert() 메소드는 경고창을 열어 줍니다.

HTML 요소에 쓰기 – innerHTML.
HTML 출력 – document.write().
alert box(경고창) - window.alert().
browser console에 쓰기 – console.log().

- 기본 명령어(메소드)

1) document.write()

document.write() 메소드(특별한 행동을 하는 자바스크립트 함수; 동사에 해당)에 들어가는 문자열에는 태그도 사용이 가능합니다.
한가지 명심할 것은 문자열은 반드시 따옴표 사이에 들러싸여 있어야 되고, 따옴표 안에 또 따옴표를 넣어야 될 경우에는 반드시 홑따옴표(' ')를 사용해야 됩니다. 그렇지 않으면 브라우저는 에러를 발생시키게 됩니다.
따옴표로 둘러싸여진 문자열은 아무리 길이가 길어도 절대 엔터키를 쳐서 줄바꿈은 하지 않도록 합니다.
HTML 문서가 로드된 후 document.write()를 사용하면 기존 HTML이 모두 삭제됩니다.
document.write() 메서드는 테스트용으로만 사용해야 합니다.

2) alert()

window.alert() 메서드는 경고 상자를 사용하여 데이터를 표시할 수 있습니다.
window키워드는 생략 가능합니다. 예) alert();
alert함수는 메시지와 OK 버튼만을 가진 다이얼로그 박스를 보여주는 함수로 사용자의 요구를 받을 필요가 없는 메시지와 경우에 사용합니다.

3) confirm()

confirm함수는 메시지와 OK/Cancel버튼을 포함한 다이얼로그 박스를 보여주는 함수로, 사용자로부터 응답을 듣고 싶을 때 사용합니다. 사용자가 OK버튼을 누르면 true를, Cancel 버튼을 누르면 false를 반환합니다.

4) prompt()

prompt 함수는 메시지와 입력 필드를 가진 다이얼로그 박스를 보여주는 함수로 사용자로부터 숫자나 문자열을 입력받고자 할 때 사용합니다.

스크립트는 <body>, 또는 <head>HTML 페이지의 섹션 또는 둘 다에 배치할 수 있습니다.

5) console.log()

디버깅을 위해 console.log()브라우저에서 메서드를 호출하여 데이터를 표시할 수 있습니다.

6) window.print()

window.print()브라우저에서 메서드를 호출하여 현재 창의 내용을 인쇄할 수 있습니다.

JavaScript 파일의 파일 확장자는 .js 입니다.
외부 스크립트를 사용하려면 <script> 태그의 src(source) 속성에 스크립트 파일 이름을 입력합니다.

HTML ID요소에 액세스하기 위해 JavaScript는 document.getElementById(id)메서드를 사용할 수 있습니다.

```
<!DOCTYPE html>
<html>
<head>
  <script>
      function myFunction() {
        document.getElementById("demo").innerHTML = "Paragraph changed.";
  }
</script>
</head>
<body>
  <h2>Demo JavaScript in Head</h2>
  <p id="demo">A Paragraph</p>
  <button type="button" onclick="myFunction()">Try it</button>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
  <h2>Demo JavaScript in Body</h2>
  <p id="demo">A Paragraph</p>
  <button type="button" onclick="myFunction()">Try it</button>
  <script>
  function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph changed.";
  }
  </script>
</body>
</html>
```

세미콜론은 JavaScript 문을 구분합니다.
각 실행 가능한 문의 끝에 세미콜론을 추가합니다.

JavaScript는 여러 공백을 무시합니다. 스크립트에 공백을 추가하여 가독성을 높일 수 있습니다.

```
let person = "Hege";
let person="Hege";
```

좋은 방법은 연산자(= + - * /) 주위에 공백을 넣는 것입니다.

```
let x = y + z;
```

최고의 가독성을 위해 프로그래머는 종종 80자보다 긴 코드 라인을 피하고 싶어합니다.
JavaScript 문이 한 줄에 맞지 않는 경우 가장 좋은 위치는 연산자 다음입니다.
예)

```
document.getElementById("demo").innerHTML =
    "Hello Dolly!";
```

JavaScript 문은 중괄호 {...} 안에 있는 코드 블록으로 그룹화할 수 있습니다.
코드 블록의 목적은 함께 실행할 명령문을 정의하는 것입니다.
블록으로 함께 그룹화된 명령문을 찾을 수 있는 곳은 JavaScript 함수입니다.

```
function myFunction() {
  document.getElementById("demo1").innerHTML = "Hello Dolly!";
  document.getElementById("demo2").innerHTML = "How are you?";
}
```

자바스크립트 키워드
JavaScript 문은 수행할 JavaScript 작업을 식별 하는 키워드로 시작하는 경우가 많습니다.
예약어 참조에는 모든 JavaScript 키워드가 나열되어 있습니다.
JavaScript 키워드는 예약어입니다. 예약어는 변수 이름으로 사용할 수 없습니다.

Keyword	설명
var	변수 선언
let	블록 변수 선언
const	블록 상수 선언
if	조건에서 실행할 명령문 블록을 표시합니다.
switch	다른 경우에 실행할 명령문 블록을 표시합니다.
for	루프에서 실행할 명령문 블록을 표시합니다.
function	함수 선언
return	기능 종료
try	명령문 블록에 대한 오류 처리를 구현합니다.

JavaScript Syntax

JavaScript Values(값)

JavaScript 구문은 두 가지 유형의 값을 정의합니다.
Fixed values(고정 값) - Literals
Variable values(변수 값) - Variables

JavaScript Literals
고정 값에 대한 가장 중요한 두 가지 구문 규칙은 다음과 같습니다.

1. 숫자는 소수를 포함하거나 포함하지 않고 작성됩니다.
2. 문자열은 큰따옴표나 작은따옴표로 묶인 텍스트입니다.

JavaScript Variables

프로그래밍 언어에서 변수는 데이터 값을 저장 하는 데 사용 됩니다.
JavaScript는 키워드 var, let, const를 사용하여 변수를 선언 합니다.
var키워드는 1995년부터 2015년까지 모든 JavaScript 코드에 사용되었습니다.
let및 키워드 const는 2015년에 JavaScript에 추가되었습니다.
이전 브라우저에서 코드를 실행하려면 var를 사용

JavaScript Let

let으로 정의된 변수는 재선언할 수 없습니다.
let으로 정의된 변수는 사용 전에 선언되어야 합니다.
let으로 정의된 변수에는 블록 범위가 있습니다.
({ } 블록 내부에 선언된 변수는 블록 외부에서 액세스할 수 없습니다.)
let으로 정의된 변수는 실수로 변수를 다시 선언할 수 없습니다.
예)

```
let x = "John Doe";
let x = 0;
// SyntaxError: 'x' has already been declared

{
  let x = 2;
}
// x can NOT be used here
```

{ } 블록 내부에 선언된 변수는 블록 외부에서 액세스할 수 있습니다.

let또는 const.로 선언된 변수는 다시 선언할 수 없습니다

```
let carName = "Volvo";
let carName;
```

==>이것은 작동하지 않습니다.

JavaScript var

프로그램의 모든 위치에서 JavaScript 변수를 다시 선언하는 var은 허용됩니다.
선언된 JavaScript 변수를 다시 선언하면 var값이 손실되지 않습니다.

```
var carName = "Volvo";
var carName;
```

변수 carName는 다음 명령문을 실행한 후에도 여전히 "Volvo" 값을 가집니다.

JavaScript Const

const로 정의된 변수는 재선언할 수 없습니다.
const로 정의된 변수는 재할당할 수 없습니다.
const로 정의된 변수에는 블록 범위가 있습니다.
JavaScript const변수는 선언될 때 값을 할당해야 합니다.

`const PI = 3.14159265359;`	O
`const PI;` `PI = 3.14159265359;`	X

```
const price1 = 5;
const price2 = 6;
```
두 개의 변수 price1및 price2 는 const키워드로 선언됩니다.
이들은 상수 값이며 변경할 수 없습니다.
다른 범위 또는 다른 블록에서 const를 사용하여 변수를 다시 선언하는 것은 허용됩니다.

```
        const x = 2;          // Allowed
        // Here x is 2
        {
          const x = 3;    // Allowed
        // Here x is 3
        }

        {
          const x = 4;    // Allowed
        // Here x is 4
        }
```

등호(=) 는 변수에 값을 할당 하는데 사용됩니다 .

x라는 변수에 값 6이 할당됩니다.

```
let x;
x = 6;
```

자바스크립트 주석
JavaScript 주석은 JavaScript 코드를 설명하고 더 읽기 쉽게 만드는 데 사용할 수 있습니다.
한 줄 주석 - 한 줄 주석은 // 로 시작합니다.
여러 줄 주석 – 여러 줄 주석은 /* 로 시작 하고 */ 로 끝납니다.

이중 슬래시 //또는 /*및 사이의 코드는 주석*/ 으로 처리됩니다 .
주석은 무시되며 실행되지 않습니다.

자바스크립트 식별자/이름

모든 JavaScript 변수는 고유한 이름으로 식별 되어야 합니다. 이러한 고유한 이름을 식별자라고 합니다 .
식별자는 JavaScript 이름입니다.
식별자는 변수, 키워드, 함수의 이름을 지정하는 데 사용됩니다.
한번에 1개의 데이터만 저장가능
새로운 데이터가 입력되면 기존의 값은 '삭제'
예약어(reserved word)는 자바스크립트에서 특별한 용도를 가진 키워드를 의미하며, 변수명으로 사용할 수 없다.

JavaScript 이름은 다음으로 시작해야 합니다.

1. 문자(AZ 또는 az)
2. 달러 기호($)
3. 또는 밑줄(_)
4. 후속 문자는 문자, 숫자, 밑줄 또는 달러 기호일 수 있습니다.
5. 숫자는 이름의 첫 문자로 사용할 수 없습니다.
6. JavaScript는 대소문자를 구분합니다

모든 JavaScript 식별자는 대소문자를 구분 합니다.

변수 lastName및 lastname는 두 가지 다른 변수입니다.
하이픈(-)은 JavaScript에서 허용되지 않습니다. 뺄셈을 위해 예약되어 있습니다.

식별자는 짧은 이름(예: x 및 y)이거나 더 설명적인 이름(연령, 합계, totalVolume)일 수 있습니다.

변수, 함수, 객체 등을 선언할 때는 변수명, 함수명, 객체명과 같이 이름이 필요합니다. 이러한 이름에 일정한 규칙을 부여하면 코드에 대한 가독성을 높일 수 있습니다.

표기법	설명
camelCase(카멜)표기법	첫 번째 단어의 첫 문자는 소문자, 두 번째 단어 이후부터는 첫 문자만 대문자로 표시한다. ->userAge, createElement()
Pascal(파스칼)표기법	각 단어의 첫 글자를 대문자로 표시한다. -> UserAge, SpeedDirection()
underscore(언더스코어)	각 단어를 언더바(_)로 이어준다. -->user_age, time_process()

변수명과 함수명은 Camel표기법으로, 객체는 Pascal 표기법으로 일정한 규칙을 만들어 사용하고, 변수는 명사, 함수는 동사로 표현하는 것이 좋습니다.

- 변수에 저장 가능한 데이터의 종류
  1) 문자형 데이터(String)
     큰따옴표 또는 작은따옴표로 둘러싸인 데이터로 문자

var userName = "하민지";
var num = "1000";

2). 숫자형 데이터(Number)
var num = 100000;

3) 논리형 데이터(Boolean)
참, 거짓으로 정의되는 데이터
var result = true;
var result = false;

4) 널형 데이터(Null) 값이 없음
var result = null;

JavaScript 연산자의 유형

다양한 유형의 JavaScript 연산자가 있습니다.

산술 연산자
할당 연산자
비교 연산자
논리 연산자
조건 연산자
유형 연산자

JavaScript 산술 연산자
산술 연산자 는 숫자에 대한 산술을 수행하는 데 사용됩니다.

Operator	Description
+	더하기
-	빼기
*	곱하기
**	지수
/	나누기
%	나머지
++	Increment(증가)
--	Decrement(감소)

JavaScript 할당 연산자

할당 연산자는 JavaScript 변수에 값을 할당합니다.

더하기 할당 연산자 ( ) 는 +=변수에 값을 추가합니다.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

JavaScript 문자열 추가

+연산자를 사용하여 문자열을 추가(연결)할 수도 있습니다.

```
let text1 = "John";
let text2 = "Doe";
let text3 = text1 + " " + text2;
```

+=할당 연산자를 사용하여 문자열을 추가(연결)할 수도 있습니다.

```
let text1 = "What a very ";
text1 += "nice day";
```

두 개의 숫자를 더하면 합계가 반환되지만 숫자와 문자열을 더하면 문자열이 반환됩니다.

JavaScript 비교 연산자

Operator	Description
==	equal to(동일)
===	equal value and equal type(동일한 값 및 동일한 유형)
!=	not equal(같지 않다)
!==	not equal value or not equal type(같지 않은 값 또는 같지 않은 유형)
>	greater than(~보다 큰)
<	less than(미만)
>=	greater than or equal to(크거나 같음)
<=	less than or equal to(이하)
?	ternary operator(삼항 연산자)

JavaScript 논리 연산자

Operator	Description
&&	logical and(논리곱)
||	logical or(논리합)
!	logical not(논리부정)

JavaScript 유형 연산자

Operator	Description
typeof	Returns the type of a variable (변수의 유형을 반환합니다.)
instanceof	Returns true if an object is an instance of an object type (객체가 객체 유형의 인스턴스인 경우 true를 반환합니다.)

문자열을 숫자로 변환
전역 메서드 Number()는 변수(또는 값)를 숫자로 변환합니다.

숫자 문자열(예: "3.14")은 숫자(예: 3.14)로 변환됩니다.
Number("3.14")
빈 문자열(예: "")은 0으로 변환됩니다.
Number(" ")
숫자가 아닌 문자열(예: "John")은 NaN(Not a Number)로 변환됩니다.
Number("John")

Method	Description
Number()	인수에서 변환된 숫자를 반환합니다.
parseFloat()	문자열을 구문 분석하고 부동 소수점 숫자를 반환합니다.
parseInt()	문자열을 구문 분석하고 정수를 반환합니다.

단항 + 연산자
단항 + 연산자를 사용하여 다음을 수행할 수 있습니다. 변수를 숫자로 변환

```
let y = "5";      // y is a string
let x = + y;      // x is a number
document.getElementById("demo").innerHTML = typeof y + "<br>" + typeof x;
```

```
let y = "John";   // y is a string
let x = + y;      // x is a number (NaN)
etElementById("demo").innerHTML = typeof y + "<br>" + typeof x;
```

만약 변수는 변환 할 수 없으며 여전히 숫자가되지만 값 (숫자가 아님)이됩니다.NaN

숫자를 문자열로 변환-->String()

전역 메서드는 숫자를 문자열로 변환할 수 있습니다.
모든 유형의 숫자, 리터럴, 변수 또는 표현식에 사용할 수 있습니다.let x = 123;

```
document.getElementById("demo").innerHTML =
  String(x) + "<br>" +
  String(123) + "<br>" +
  String(100 + 23);
```

toString() 메서드는 숫자를 문자열로 변환합니다.

```
<p id="demo"></p>
let x = 123;
document.getElementById("demo").innerHTML =
  x.toString() + "<br>" +
(123).toString() + "<br>" +
(100 + 23).toString();
```

Method	Description
toExponential()	지수 표기법을 사용하여 반올림되고 작성된 숫자가 포함된 문자열을 반환합니다
toFixed()	숫자를 반올림하고 지정된 소수점 이하 자릿수로 쓴 문자열을 반환합니다.
toPrecision()	지정된 길이로 쓰여진 숫자가 포함된 문자열을 반환합니다.

Date methods

Method	Description
getDate()	날짜를 숫자로 가져오기(1-31)
getDay()	요일을 숫자(0-6)로 가져오기
getFullYear()	네 자리 연도(yyyy) 가져오기
getHours()	시간 가져오기(0-23)
getMilliseconds()	밀리초 가져오기(0-999)
getMinutes()	분 가져오기(0-59)
getMonth()	월 가져오기(0-11)
getSeconds()	초 가져오기(0-59)
getTime()	시간 가져오기(1970년 1월 1일 이후 밀리초)

부울을 숫자로 변환

```
Number(false)     // returns 0
Number(true)      // returns 1
```

부울을 문자열로 변환

```
String(false)     // returns "false"
String(true)      // returns "true"
```

자동 형식 변환

자바 스크립트가 "잘못된"데이터 유형에서 작동하려고하면 값을 "올바른" 형식으로 변환합니다.

```
5 + null    // returns 5         because null is converted to 0
"5" + null  // returns "5null"   because null is converted to "null"
"5" + 2     // returns "52"      because 2 is converted to "2"
"5" - 2     // returns 3         because "5" is converted to 5
"5" * "2"   // returns 10        because "5" and "2" are converted to 5 and 2
```

search() 메서드는 표현식을 사용하여 일치 항목을 검색하고 일치 위치를 반환합니다.
대소문자 구분하지 않는다.

```
let text = "Visit W3Schools!";
let n = text.search("W3Schools");
document.getElementById("demo").innerHTML = n;
```

replace() 메서드는 패턴이 교체된 수정된 문자열을 반환합니다.

```
<button onclick="myFunction()">Try it</button>
<p id="demo">Please visit Microsoft!</p>
<script>
function myFunction() {
  let text = document.getElementById("demo").innerHTML;
  document.getElementById("demo").innerHTML =
  text.replace("Microsoft","W3Schools");
}
</script>
```

JavaScript 연산자 우선 순위
연산자 우선 순위는 산술 식에서 연산이 수행되는 순서를 설명합니다.

곱셈(*)과 나눗셈(/)은 덧셈(+)과 뺄셈(-)보다 우선 순위가 높습니다.
우선 순위가 같은 작업(* 및 / 등)은 왼쪽에서 오른쪽으로 계산됩니다.
let x = 100 + 50 * 3;-->450
let x = 100 / 50 * 3;--> 6

JavaScript Errors

The try statement 실행할 코드 블록을 정의합니다.
The catch statement 오류를 처리하는 코드 블록을 정의합니다.
The finally statement 결과에 관계없이 실행할 코드 블록을 정의합니다.
The throw statement 사용자 지정 오류를 정의합니다.

```
<script>
function myFunction() {
  const message = document.getElementById("p01");
  message.innerHTML = "";
  let x = document.getElementById("demo").value;
  try {
    if(x == "")  throw "is empty";
    if(isNaN(x)) throw "is not a number";
    x = Number(x);
    if(x > 10)   throw "is too high";
    if(x < 5)  throw "is too low";
  }
  catch(err) {
    message.innerHTML = "Input " + err;
  }
  finally {
    document.getElementById("demo").value = "";
  }
}
</script>

<p>Please input a number between 5 and 10:</p>

<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>
<p id="p01"></p>
```