웹 브라우저에서 읽고 실행되는 언어

보안성이 없음

html 내에 작성

대소문자의 구문 엄격

적용방법

* 내부 스크립트
* html 문서 내에 스크립트 정의 후 직접 작성
* 필요 위치에

<script type=”text/javascript>

스크립트 효과

</script>

* 외부 스크립트
* 스크립트 효과를 다른 문서에 작성하고 “이름.js”로 저장
* 효과를 적용할 문서에

<script type=”text/javascript src=”이름.js”>

다른 문장 작성 금지

</script>

ECMA script (ES6 or 7)

\*\*\* 자료형 \*\*\*

연산에 사용되는 값의 종류

* 상수 : 값이 정해져 있는 형식

1. 정수

음수, 소수를 제외한 양의 실수

8진수, 16진수 포함

* 실수

실제 사용 가능한 모든 수

* 문자형 상수 (string)

일반적인 텍스트 “텍스트” 사용\

* 불린(Boolean)

true(1) / false(0) 의 형식

* null

값이 없음

변수(variable)

변할 수 있는 ㄱ밧

값을 저장하는장소나 이름

변수 선언문

* var : 변수 재선언 & 변수 재할당 가능, 생략 가능

ex)var abc

* let : 재할당 O / 재선언 X
* const :재할당 X / 재선언 X (한번 정해진 값 고정)

선언과 동시에 값 입력가능

**\*\*\* 연산자 \*\*\***

| **산술 연산자** | **설명** |
| --- | --- |
| + | 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 더함. |
| - | 왼쪽 피연산자의 값에서 오른쪽 피연산자의 값을 뺌. |
| \* | 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 곱함. |
| / | 왼쪽 피연산자의 값을 오른쪽 피연산자의 값으로 나눔. |
| % | 왼쪽 피연산자의 값을 오른쪽 피연산자의 값으로 나눈 후, 그 나머지를 반환함. |

**예제**

*var* x = **10**, y = **4**;

document.write(x + y + "<br>"); *// 14*

document.write(x - y + "<br>"); *// 6*

document.write(x \* y + "<br>"); *// 40*

document.write(x / y + "<br>"); *// 2.5*

document.write(x % y);          *// 2*

#### 대입 연산자(assignment operator)

대입 연산자는 변수에 값을 대입할 때 사용하는 이항 연산자이며, 피연산자들의 결합 방향은 오른쪽에서 왼쪽입니다.

또한, 앞서 살펴본 산술 연산자와 결합한 다양한 복합 대입 연산자가 존재합니다.

| **대입 연산자** | **설명** |
| --- | --- |
| = | 왼쪽 피연산자에 오른쪽 피연산자의 값을 대입함. |
| += | 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 더한 후, 그 결괏값을 왼쪽 피연산자에 대입함. |
| -= | 왼쪽 피연산자의 값에서 오른쪽 피연산자의 값을 뺀 후, 그 결괏값을 왼쪽 피연산자에 대입함. |
| \*= | 왼쪽 피연산자의 값에 오른쪽 피연산자의 값을 곱한 후, 그 결괏값을 왼쪽 피연산자에 대입함. |
| /= | 왼쪽 피연산자의 값을 오른쪽 피연산자의 값으로 나눈 후, 그 결괏값을 왼쪽 피연산자에 대입함. |
| %= | 왼쪽 피연산자의 값을 오른쪽 피연산자의 값으로 나눈 후, 그 나머지를 왼쪽 피연산자에 대입함. |

##### 예제

*var* x = **10**, y = **10**, z = **10**;

x = x - **5**;

y -= **5**; *// y = y - 5 와 같은 표현임.*

z =- **5**; *// z = -5 와 같은 표현임.*

#### 증감 연산자(increment and decrement operator)

증감 연산자는 피연산자를 1씩 증가 혹은 감소시킬 때 사용하는 연산자입니다.

이 연산자는 피연산자가 단 하나뿐인 단항 연산자입니다.

증감 연산자는 해당 연산자가 피연산자의 어느 쪽에 위치하는가에 따라 연산의 순서 및 결과가 달라집니다.

| **증감 연산자** | **설명** |
| --- | --- |
| ++x | 먼저 피연산자의 값을 1 증가시킨 후에 해당 연산을 진행함. |
| x++ | 먼저 해당 연산을 수행하고 나서, 피연산자의 값을 1 증가시킴. |
| --x | 먼저 피연산자의 값을 1 감소시킨 후에 해당 연산을 진행함. |
| x-- | 먼저 해당 연산을 수행하고 나서, 피연산자의 값을 1 감소시킴. |

##### 예제

*var* x = **10**, y = **10**;

document.write((++x - **3**) + "<br>"); *// x의 값을 우선 1 증가시킨 후에 3을 뺌.*

document.write(x + "<br>");         *// 11*

document.write((y++ - **3**) + "<br>"); *// 먼저 y에서 3을 뺀 후에 y의 값을 1 증가시킴.*

document.write(y);                  *// 11*

#### 비교 연산자(comparison operator)

비교 연산자는 피연산자 사이의 상대적인 크기를 판단하여, 참(true)과 거짓(false)을 반환합니다.

비교 연산자는 모두 두 개의 피연산자를 가지는 이항 연산자이며, 피연산자들의 결합 방향은 왼쪽에서 오른쪽입니다.

| **비교 연산자** | **설명** |
| --- | --- |
| == | 왼쪽 피연산자와 오른쪽 피연산자의 값이 같으면 참을 반환함. |
| === | 왼쪽 피연산자와 오른쪽 피연산자의 값이 같고, 같은 타입이면 참을 반환함. |
| != | 왼쪽 피연산자와 오른쪽 피연산자의 값이 같지 않으면 참을 반환함. |
| !== | 왼쪽 피연산자와 오른쪽 피연산자의 값이 같지 않거나, 타입이 다르면 참을 반환함. |
| > | 왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 크면 참을 반환함. |
| >= | 왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 크거나 같으면 참을 반환함. |
| < | 왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 작으면 참을 반환함. |
| <= | 왼쪽 피연산자의 값이 오른쪽 피연산자의 값보다 작거나 같으면 참을 반환함. |

자바스크립트에서 비교 연산자는 피연산자의 타입에 따라 두 가지 기준으로 비교를 진행합니다.

1. 피연산자가 둘 다 숫자면, 해당 숫자를 서로 비교합니다.

2. 피연산자가 둘 다 문자열이면, 문자열의 첫 번째 문자부터 알파벳 순서대로 비교합니다.

##### 예제

*var* x = **3**, y = **5**;

*var* a = "abc", b = "bcd";

document.write((x > y) + "<br>");  *// y의 값이 x의 값보다 크므로 false*

document.write((a <= b) + "<br>"); *// 알파벳 순서상 'a'가 'b'보다 먼저 나오므로 'a'가 'b'보다 작음.*

document.write(x < a);             *// x의 값은 숫자이고 a의 값은 문자열이므로 비교할 수 없음.*

#### 논리 연산자(logical operator)

논리 연산자는 주어진 논리식을 판단하여, 참(true)과 거짓(false)을 반환합니다.

&& 연산자와 || 연산자는 두 개의 피연산자를 가지는 이항 연산자이며, 피연산자들의 결합 방향은 왼쪽에서 오른쪽입니다.

! 연산자는 피연산자가 단 하나뿐인 단항 연산자이며, 피연산자의 결합 방향은 오른쪽에서 왼쪽입니다.

| **논리 연산자** | **설명** |
| --- | --- |
| && | 논리식이 모두 참이면 참을 반환함. (논리 AND 연산) |
| || | 논리식 중에서 하나라도 참이면 참을 반환함. (논리 OR 연산) |
| ! | 논리식의 결과가 참이면 거짓을, 거짓이면 참을 반환함. (논리 NOT 연산) |

다음은 논리 연산자의 모든 동작의 결과를 보여주는 진리표(truth table)입니다.

| **A** | **B** | **A && B** | **A || B** | **!A** |
| --- | --- | --- | --- | --- |
| true | true | true | true | false |
| true | false | false | true | false |
| false | true | false | true | true |
| false | false | false | false | true |

##### 예제

*var* x = **true**, y = **false**;

document.write((x && y) + "<br>"); *// false (논리 AND 연산)*

document.write((x || y) + "<br>"); *// true  (논리 OR  연산)*

document.write(!x);                *// false (논리 NOT 연산)*

#### 비트 연산자(bitwise operator)

비트 연산자는 논리 연산자와 비슷하지만, 비트(bit) 단위로 논리 연산을 수행합니다.

또한, 비트 단위로 전체 비트를 왼쪽이나 오른쪽으로 이동시킬 때도 사용합니다.

| **비트 연산자** | **설명** |
| --- | --- |
| & | 대응되는 비트가 모두 1이면 1을 반환함. (비트 AND 연산) |
| | | 대응되는 비트 중에서 하나라도 1이면 1을 반환함. (비트 OR 연산) |
| ^ | 대응되는 비트가 서로 다르면 1을 반환함. (비트 XOR 연산) |
| ~ | 비트를 1이면 0으로, 0이면 1로 반전시킴. (비트 NOT 연산) |
| << | 지정한 수만큼 비트를 전부 왼쪽으로 이동시킴. (left shift 연산) |
| >> | 부호를 유지하면서 지정한 수만큼 비트를 전부 오른쪽으로 이동시킴. (right shift 연산) |
| >>> | 지정한 수만큼 비트를 전부 오른쪽으로 이동시키며, 새로운 비트는 전부 0이 됨. |

연결 연산자

+ : 연산자의 좌변과 우변의 값이 산술 연산이 불가능한 경우 두개의 값을 이어주는 연산자

조건 연산자

조건의 결과에 따라 다른 값 출력

(조건)? A : B | 조건이 참일 경우 : 조건이 거짓일 경우

\*\*\* 제어문 \*\*\*

해당언어를 효율적으로 사용할 수 있도록 하는 문장

반복문

1. 일정한 규칙을 가지고 있는 문장을 직접 작성하지 않고 여러 번 실행 시키는 문장

조건(배열, 숫자 범위) + 반복문

for

while

do while

* while

조간 + 반복문

while(조건)

조건이참인 동안 반복할 문장

\*\*\* 함수 \*\*\*

미리 약속되어 있는 명령어

1. 내장함수 – 만들어져 있는 명령어
2. 사용자 정의 함수 – 만들어서 사용하는 명령어

대화상자 함수

* alert(“문자열”) : 메시지 상자
* confirm(“문자열”) : 확인/취소 선택 상자
* prompt(“질문메시지”, “초기메세지”) : 사용자 입력상자

화살표 함수

* 뚱뚱한 화살표 (=> arrow fuction)를 사용하여 함수 선언
* ES5 일반 함수

fuction (매개변수) { }

* 화살표 함수

==> { 실행 할 문장 }

문법은 간단해지지만 공통된 규칙이나 일반적인 사용방법이 필요 할 수 있음

        document.querySelector("CSS선택자 동알한 값 입력").vlaue = "텍스트"

        //CSS선택자 동알한 값 입력 = #id, .class, element 값등등

getElementByid, querySelector 를 제외만 class, name, tagname 등은 지원 안되는 브라우저가 많음

\*\*\* DOM 코어 \*\*\*

HTML-DOM : HTML 요소들의 속성제어

DOM 코어 : 스크립트가 지원되는 모든 기기에서 사용

ex) object.getAttribute(“src”)

HTML-DOM : 웹 브라우저에서만 사용 ( 정의되지 않은 속성값 불러오기 가능)

ex) obj.

html5 부터는 html에서 변수 사용가능 = data / data-role

ex) data-NAME = “####” / ex) data-role- NAME = “####”

\*\*\* 제이쿼리 \*\*\*

1. css 셀렉터

html 내의 엘리먼트들을 손쉽게 표현 및 사용가능

1. 플러그인 아키텍쳐

이미 개발된 많은 플러그인을 쉽고 빠르게 사용

1. 메소드 체인

여러 개의 기능을 한줄에 나열하며 불필요한 코드 반복을 줄입

1. 크로스 브라우저

브라우저별 발생 이벤트를 각각 지정 해 줄 필요가 없음

* 제이쿼리 사용 전 준비

1. 라이브러리 가져오기
2. 문서가 준비 됐는지 확인하기

적용방법

<script type=”text/javascript” src=”제이쿼리 문서 경로”></script>

1. local import

* 파일을 다운로드 후 html 문서에 임포트
* js 파일을 항상 같이 이동
* 한국 서버에 있는 js파일을 다른 나라의 사용자가 접속하려 할 경우 속도 저하 발생 가능

1. CDN

* 특정 웹 사이트에서 제공하고 있는 파일을 링크
* 웹 사용 불가 시 스크립트 동작 안됨
* jquery.com / google.com / Microsoft.com
* compressed 버전 : 파일을 최소화 시키기 위해 불필요한 공백과 줄바꿈 생략

(일반적으로 사용)

* uncompressed 버전 : 코멘드등 포함 / 코드분석시 사용

selector($()함수)

jquery(“선택자”).메소드() – 기본적으로 따옴표로 선택 (문서를 나타내는 객체에 사용x window, document)

$(선택자).메소드()

$(누구한테).어떤효과를()

$(누구한테).어떤효과를(“”)

$(누구한테).어떤효과를({ })

$(누구한테).어떤효과를(function(){ })

$(누구한테).어떤효과를().다른효과({ }. 다른효과(function() { })

ready() 메소드

1. javascript 의 window.onload와 같은 기능이지만 이 기능보다 정확하게 동작
2. 스크립트가 먼저 작성되고 body 요소가 나중에 작성되기 때문에 스크립트 오류가 발생하는 것을 방지
3. 문서내에 한번만 선언
4. $(document).read(function(){ 실항할 문장 전체 } )