[변수](#e2q7ia4313n) / 연산자 / [선택자](#qzxsw3w9w9l1) / [CSS변경](#h2l23mk2jkqc) / 배열 / [클래스](#lvsmghv1sizo) / [이벤트](#77fyw53e56z6)

|  |
| --- |
| **01. Javascript 기초** |

**변수**

**값을 저장하는 그릇**

**변수타입**

|  |
| --- |
| // number (정수, 실수) var x = 100; var y = -100; var z = 3.14;   // string (문자열) var message = "Hello, world";   // boolean (true, false) var isSuccess = true; var isFail = false;   // null, undefined var obj = "asdf"; obj = null;   // object (객체형) var person = {}; var person = {age:21, name:"jobs"};   // object (배열형) var numbers = []; var numbers = [1, 2, 3];   // function (함수형) var square = function(x) { return x \* x; }; |

**식별자 - 변수명, 함수명과 같은 이름을 붙일 때 사용하는 단어, 네이밍 규칙**

|  |
| --- |
| ① 키워드와 공백 문자를 사용할 수 없음  ② 숫자로 시작할 수 없음  ③ 특수문자는 \_과 $만 허용    break는 예약어로 사용불가  예약어 리스트 확인하기 <https://www.w3schools.com/js/js_reserved.asp> |

**주석**

|  |  |
| --- | --- |
| **HTML 태그 주석** | **JavaScript 주석** |
| <!-- --> | 한 문장 : //주석문,  여러 문장 : /\* 주석문 \*/ |

**자료형의 종류**

|  |
| --- |
| • 프로그래밍 언어로 조작할 수 있는 값의 유형  • 문자열(String), 숫자(Number), 불린 (Boolean), 널(null), 언 디파인드(undefined), 객체(Object) |

**사칙 및 연산자**

|  |
| --- |
|  |

**- 비교 연산자**

|  |  |
| --- | --- |
| **연산자** | **설명** |
| > | 좌변이 우변보다 큼 |
| >= | 좌변이 우변보다 크거나 같음 |
| < | 좌변이 우변보다 작음 |
| <= | 좌변이 우변보다 작거나 같음 |
| == | 좌변이 우변이 같음 |
| != | 좌변이 우변이 다름 |

**- 논리 연산자**

|  |  |
| --- | --- |
| **연산자** | **설명** |
| ! | 논리 부정 (참은 거짓으로, 거짓은 참으로) |
| && | 논리곱 ( 좌우변 모두 참일 때 참) |
| || | 논리합(좌우변 둘 중 하나라도 참일 때 참) |

**복합 대입 연산자**

|  |
| --- |
|  |

|  |
| --- |
|  |

**증감 연산자**

|  |
| --- |
|  |

**증간연산자의 종류**

|  |
| --- |
|  |

**자동 자료형 변환 시 우선 순위**

|  |
| --- |
|  |

**[ 해보기 ]**

|  |  |
| --- | --- |
| //문자열로 자동 자료형 변환  console.log(12 + 34);  console.log('12' + 34);  console.log(12 + '34');   //숫자로 자동 자료형 변환  console.log('12' / 2);  console.log(12 \* '2');  console.log('12' - '2'); |  |

**연산자**

|  |
| --- |
| // 산술 연산자 // +, -, \*, /, % // ++, -- var x = 5; var y = 8; console.log(x + y); console.log(x - y); console.log(x \* y); console.log(x / y); console.log(x % y);   console.log(x++); console.log(x); console.log(++x);   // 관계 연산자 // >, >=, <, <=, ==, != console.log(10 > 5); // true console.log(5 >= 5); // true console.log(5 == 5); // true console.log(10 != 5); // true   // 논리 연산자 // &&, ||, ! console.log((10 > 5) && (9 > 5)); // true console.log(!true); // false console.log(!false); // true |

**함수**

|  |
| --- |
| function test() {  console.log("테스트입니다."); }   function sum(math, english, science) {  return (math + english + science) / 3; }   ============================================   function 함수명(전달인자1, 전달인자2, ... , 전달인자n){  ...  코드  ...  return 반환값; }   var 함수명 = function(전달인자1, 전달인자2, ... , 전달인자n){  ...  코드  ...  return 반환값; |

**if 조건문**

|  |
| --- |
|  |

**else**

|  |
| --- |
| if (조건) {   //조건이 true 일 때 실행할 문장   } else {   //조건이 false일 때 실행할 문장 } |

**else if**

|  |
| --- |
| if (조건1) {   //조건1이 true일 때 실행할 문장  } else if (조건2) {   //조건1이 false이고 조건2가 true일 때 실행할 문장 } else if (조건3) {   //조건1과 조건2가 false이고 조건3이 true일 때 실행할 문장 } else {   //조건1, 조건2, 조건3이 false일 때 실행할 문장  } |

**switch 조건문**

|  |
| --- |
|  |

**[ 해보기 ]**

|  |
| --- |
| var number = prompt('숫자를 입력하세요', ''); number = Number(number);  switch (number % 2) {  case 0 :  alert('짝수입니다.');  break;  case 1:  alert('홀수입니다.');  break;  default:  alert('숫자가 아닙니다.');  break; } |

**배열**

|  |
| --- |
| var array1 = [ 1, 2, 3, 4, 5 ]; array2 = [ 12, 'string', false, function () { }, array1 ]; |

**배열의 개수**

|  |
| --- |
| array1.length |

**배열에서 값 선택하기 -** 순번을 통해 선택한다. 0번부터 시작한다.

|  |
| --- |
| array1[0] |

**배열의 끝에 값 추가하기**

|  |
| --- |
| array1.push(6) |

**배열의 끝값 제거하기**

|  |
| --- |
| array1.pop() |

**배열의 앞에 값 추가하기**

|  |
| --- |
| A.unshift(b); // A배열의 앞에 b값을 추가한다. |

**배열의 앞값 제거하기**

|  |
| --- |
| A.shift(); // A배열의 맨앞값을 제거한다. |

**배열과 배열을 하나로 합치기**

|  |
| --- |
| A.concat(B); //a배열의 값과 b배열의 값을 합친다. |

**indexOf // 특정 문자열의 인덱스값 알아내기**

|  |
| --- |
| A.indexOf('b'); // a변수에서 b라는 문자열의 인덱스 번호를 확인 //일치하는 문자열이 없으면 -1 출력 |

**시간을 한번만 재고 그 시간이 지나면 할일**

|  |
| --- |
| setTimeout(할일, 시간); // 할일 = function(){ } // 시간은 밀리세컨드 1000 = 1초  setTimeout 멈추기 setTimeout(멈출 대상의 이름);  var timer = setTimerout(.....); clearTimeout(timer); |

**일정 시간마다 할일**

|  |
| --- |
| setInterval(할일, 시간); // 할일 = function(){ } // 시간은 밀리세컨드 1000 = 1초  setInterval 멈추기 clearInterval(멈출 대상의 이름);  var timer = setInterval(.....); clearInterval(timer); |

**자주 사용하는 Math.메서드**

|  |  |  |
| --- | --- | --- |
| **이름** | **설명** | **예시** |
| abs(num) | 매개변수 num의 값을 절대값으로 반환 | var b = Math.abs(-7.25); // b변수에 값은 7.25 |
| floor(num) | 매개변수 num을 정수로 내림 | var c = Math.floor(7.85); // c변수에 값은 7 |
| ceil(num) | 매개변수 num을 정수로 올림 | var c = Math.ceil(7.25); // c변수에 값은 8 |
| round(num) | 매개변수 num을 반올림 | var d = Math.round(7.25); // d변수에 값은 7 var e = Math.round(7.65); // e변수에 값은 8 |

**for 반복문**

|  |
| --- |
| for(시작; 조건; 증감){  반복할 일  } |

**구구단 3단 출력**

|  |
| --- |
| for(var i = 1; i< 9; i++){  document.write('3x' + i + "=" + (i\*3)); } |

|  |
| --- |
| **02. Javascript 선택자** |

**DOM 이란?**

- **Document Object Model**의 약어로 DOM이라고도 함

- 문서에 접근하기 위한 표준으로, W3C (World Wide Web Consortium)에서 정의

- HTML DOM은 HTML 구성요소들을 획득, 변경, 추가, 삭제하기 위한 표준

- HTML 문서를 브라우저에서 로드 시 각 구성요소들을 객체화하여 객체 트리 구조를 나타냄

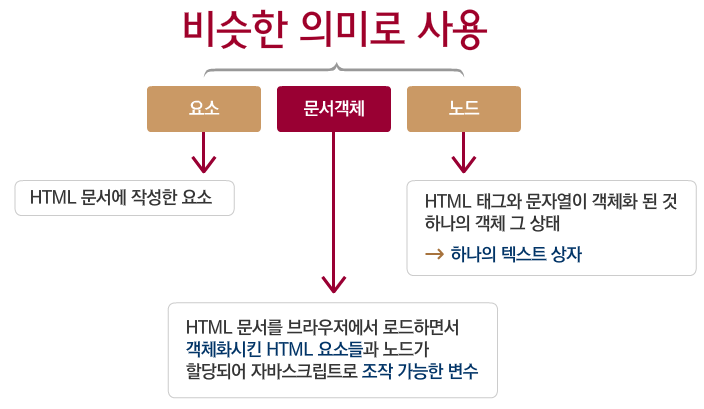
**[HTML]**

|  |
| --- |
| <html>  <head>  <title> My title </title>  </head>  <body>  <h1>My text</h1>  <a href="http://www.abc.com/">Good</a>  </body> </html> |

**[HTML 문서 객체 모델 ]**

|  |
| --- |
|  |

**요소, 문서 객체, 노드**



**1) document 객체**

- HTML 문서와 관련있는 객체

- window 객체의 한 부분으로써 **window.document** 속성을 통해 접근

- window는 생략 가능

- 좁은 의미의 문서 객체 모델 : document 객체와 관련된 객체의 집합

- 로드된 문서에 단 하나의 document 객체 존재

- HTML 문서 객체 접근의 시발점

**2) 문자열 출력**

**- write() 메서드**

• 문서에 JavaScript 코드나 HTML 표현을 작성

• 입력된 매개 변수를 순서대로 문서에 덧붙여 출력

• HTML 요소를 사용하여 문자열의 형식 지정 가능

**- writeln() 메서드**

• write() 메서드와 동일한 기능을 제공하나, 문자열 출력 후 줄 바꿈을 함

**[** **자바스크립트에서 요소 선택하기 ]**

|  |  |
| --- | --- |
| **JAVASCRIPT** | **console** |
| document.getElementById("myDIV") | 단일요소 |
| document.getElementsByTagName("div") | 배열 |
| document.getElementsByClassName("myDiv") | 배열 |
| document.querySelector('#myStyle') document.querySelector('.myStyle') | 단일요소 |
| document.querySelectorAll('.myStyle') | 배열 |

**[ 자바스크립트에서 CSS 수정하기 ]**

|  |
| --- |
| **JAVASCRIPT** |
| document.getElementById("myDIV").style.color='red' |
| document.getElementById("myDIV").style.display='none' |

**[ 배열의 형식으로 저장되는 요소들(클래스명, 태그명)의 개수 ]**

|  |
| --- |
| **paragraphs.length;** |

**[ 배열의 형식으로 저장되는 요소들(클래스명, 태그명)의 첫번째 선택하기 ]**

|  |
| --- |
| **myDiv[0] paragraphs[0]** |

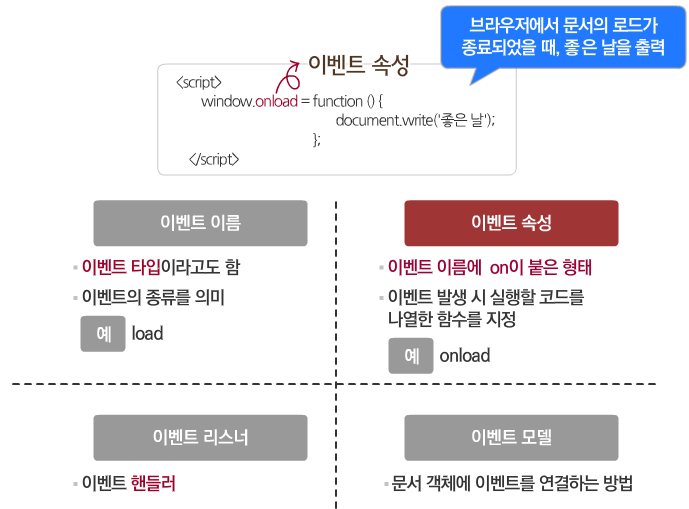
**[ 배열의 형식으로 저장되는 요소들(클래스명, 태그명) 모두의 색상 변경하기 ]**

|  |
| --- |
| for (var i=0; i<paragraphs.length; i++){  paragraphs[i].style.color = 'green';  } |

**[ 배열의 형식으로 저장되는 요소들(클래스명, 태그명)을 클릭하면 할일 ]**

|  |
| --- |
| for (var i=0; i< $targetLink.length; i++){  targetLink[i].addEventListener('click',function(){   }); } |

|  |
| --- |
| **03.** **Javascript 이벤트** |



**[ 자바스크립트 클릭 이벤트 ]**

|  |
| --- |
| **JAVASCRIPT** |
| document.addEventListener('DOMContentLoaded', function(){  //DOM의 컨텐츠 즉(BODY)의 내용이 모두 로드 되면 할일  }); |

**[ 자바스크립트 클릭 이벤트 ]**

|  |
| --- |
| **JAVASCRIPT** |
| document.getElementById("myDIV").addEventListener('click',function(){  }); |

**[ 기본문법 ]**

|  |
| --- |
| //연결하기 선택자.addEventListener('이벤트종류', 할일);  //해제하기 선택자.removeEventListener('이벤트종류', 할일); |

**[ script.js ]**

|  |
| --- |
| var submitBtn = document.getElementById('submit'),  container = document.querySelector('.container');   submitBtn.addEventListener('click', function(){  console.log('Hey, I\'ve been clicked!'); }); |

**Event 종류**

|  |  |
| --- | --- |
| **이벤트 명** | **설명** |
| **click** | 클릭시 발생 |
| **change** | input 요소의 변동이 있을 때 발생 |
| **scroll** | 스크롤이 발생 |
| focus | 포커스를 얻었을 때 발생  (링크, input등의 요소에) |
| keydown | 키를 눌렀을 때 발생 |
| keyup | 키에서 손을 땟을 때 발생 |
| load | 로드가 완료 되었을 때 발생  (body, image등 선택자가 로드 되었을 때) |
| mousedown | 마우스를 클릭 했을 때 발생 |
| mouseup | 마우스에서 손을 땟을 때 발생 |
| **mouseover** | 마우스가 특정 객체 위로 올려졌을 때 발생 |
| **mouseout** | 마우스가 특정 객체 밖으로 나갔을 때 발생 |
| **mouseenter** | 마우스가 특정 객체 안에 들어왔을 때 |
| **mouseleave** | 마우스가 특정 객체에서 떠났을 때 |
| select | option 태그 등에서 선택을 했을 때 발생 |

|  |
| --- |
| **04. Javascript - class 관련** |

|  |  |
| --- | --- |
| **JAVASCRIPT** | **Description** |
| document.getElementById('myDIV').className ='class1'; | 클래스명 입력(교체) |
| document.getElementById('myDIV').classList.add('class1'); | 클래스명 추가 (기존 클래스명 유지) |
| document.getElementById('myDIV').classList.add('class1', 'class2'); | 클래스명 추가(다중) |
| document.getElementById('myDIV').classList.remove('class1'); | 클래스명 제거 |
| document.getElementById('myDIV').classList.remove('class1', 'class2'); | 클래스명 제거(다중) |
| document.getElementById("myDIV").classList.toggle("class1"); | 클래스명 추가/제거 동시에 |
| document.getElementById('myDIV').classList.contains('class1'); | 클래스명을 포함하는지 여부 (조건문에 활용) |
| document.getElementById('myDIV').classList.contains('class1', 'class2'); | 클래스명을 포함하는지 여부 |

|  |
| --- |
| **05. Javascript - attribute 관련** |

|  |  |
| --- | --- |
| **JAVASCRIPT** | **Description** |
| document.getElementById('myDIV').getAttribute('id'); | id라는 속성의 값 가져오기(반환하기) |
| document.getElementById('myDIV').setAttribute('id', 'new value'); | id라는 속성의 값을 new value로 교체하기 |
| document.getElementById('myDIV').hasAttribute('id') | id라는 속성의 값이 있는지 없는지 여부 (true, false)  조건문 |

**06. 유사배열을 배열로 변환하기**

유사배열을 배열로 변환하는 이유는 forEach와같은 배열에서만 사용하는 메서드를 쓰기 위해서.

Array.from()를 사용하는 방법

|  |
| --- |
| Array.from(document.querySelectorAll('#list > .item')) |

Array.slice()를 사용하는 방법

|  |
| --- |
| Array.slice(document.querySelectorAll('#list > .item')) |

스프레드 연산자를 사용하는 방법

|  |
| --- |
| [...document.querySelectorAll('#list > .item')] |