**자바스크립트**

**원시데이터형태(가장 초창기에 만들어진 타입)**

**(1) String – 문자데이터** “가” + ’가‘ + `가` = 가가가

**(2) Number – 숫자데이터** 1+2+3=6 (\* / - %가능) %(나머지연산-나머지값)

**(3) Boolean – 참 거짓 값 데이터 형태**

**- 비교를 통한 참거짓 (주의 : ! 아니다, == 같다. )**

**- 사실에 근거 참거짓 (숫자데이터 0, 빈문자인 문자데이터, 초기화안한변수)**

**- 논리연산자 &&** - and 좌우둘다 true라면 전체가 true

**- 논리연산자 ||** - or 좌우하나라도 true라면 전체가 true

**- 삼항연산자 – (조건?”참값”:”거짓값”)**

**- 수학관련도구 - Math객체함수 console.log(Math.round(값)) 반올림**

**복합/참조데이터**

**(4) array – 배열데이터 []**

**(5) object – 객체데이터 {key(변수로 사용):value – 프로퍼티, 메서드도 들어올 수 있음} 객체 = 사물**

**(6) function(){} – 함수데이터 (동작을 메서드라고 한다.)**

(기본형태) let 변수 = {key:value(=property), key:value, key:value}

(예시) let person = {name(key):”손나라”(value), age:33, gender:”여자”, married:false}

(부르는방법) alert(person1[”name”]), alert(person1변수.name키)

--------------------------------------------------------------------------------------------------------------

**typeof()** – 괄호 안의 데이터 형식 확인 도구 (문자 or 숫자)

**prompt()** – 질문 후 답변을 그 자리에 문자데이터로 남기는 도구

**Number()** – 진짜 숫자데이터로 변경 도구

**document.write()** – 그대로 필기해주는 도구

--------------------------------------------------------------------------------------------------------------

**상수** : 변하지 않는 수

**변수** : 변할 가능성이 있는 수

--------------------------------------------------------------------------------------------------------------

**•if조건문**

- if(조건){소스코드}

- if(조건){소스코드1}else{소스코드2}

조건이 참이라면 소스코드1을 해석하고, 거짓인 경우 소스코드2를 해석.

- if(조건){소스코드1}else if(조건2){소스코드2}else{소스코드3}

조건1이 참이라면 소스코드1을 해석, 거짓이면 조건2가 참이라면 소스코드2를 해석, 거짓이면 소스코드3을 해석.

--------------------------------------------------------------------------------------------------------------

**• while반복문 (조건이 참인동안 소스코드를 반복한다.)(**보안약함. 바깥쪽에서 접근가능함.)

- while(조건){반복소스코드}

- <ul><script> let n = 0; while(n<9) {n++; document.write(`<li>2x${n}=${n\*2}</li>`)} </script></ul>

**•while반복문의 중첩**

- while(조건){반복소스코드 while(조건){반복소스코드}}

- let a=0; while(a<3) {a++; let b = 0; while(b<2) {b++; document.write(`a의 값은 ${a}이고 b의 값은 ${b}이다.<br>`)}}

--------------------------------------------------------------------------------------------------------------

**• for반복문** (while보다 안전성이 더 좋음. 바깥쪽에서 접근 못하고, 안쪽에서만 접근 가능함.)

– for**(**반복변수;조건식;증감식**){**반복소스코드**}**

- for (let t = 0; t <= 2; t++){document.write(t+”<br>”)}

**• for반복문의 중첩**

- for**(**반복변수;조건식;증감식**){**반복소스코드 for**(**반복변수;조건식;증감식**){**반복소스코드**}}**

- for ( let a = 1; a <= 9; a++ ) { document.write(`<ul>`)

for (let b = 1; b <= 9; b++) {

document.write(`<li>${a}x${b}=${a\*b}</li>`) } document.write(`</ul>`)}

<증식> - j에 1을 더한 값을 적용한다.

J=j+1 / J+=1 / J++

<감식> - k의 1을 뺀 값을 적용한다.

K=k-1 / k-=1 / k—

--------------------------------------------------------------------------------------------------------------

**• 배열 – array**

- let 변수이름 = [원소0, 원소1, 원소2, [원소3-1, 원소3-2,,,], 원소4, 원소5]

- 조건식의 변수 뒤에 .length 붙여주면 변수개수만큼 배열개수 늘려줄 수 있음.

--------------------------------------------------------------------------------------------------------------

**• 함수 -function**

- function 함수명(재료){동작할 소스코드}

--------------

Function a ( ) {

alert(“hi”)

}

a( )

Function a( ) 함수의 정의부

{alert(“hi”)} a라는 함수가 실행될 때 동작하는 소스코드

a( ) 함수의 실행부

- ( ) 실행한다는 의미

-------------

함수의 매개변수(재료) 설정

function b (name) { alert (name) }

b(“손나라”)

b(“오나라”)

-------------

매개변수의 개수는 제한이 없다.

function c (num1, num2) {

alert (num1 + num2)

}

c ( 1, 2 )

let number = Math.round(32.1)

let name = prompt(“당신의 이름을 입력해주세요”)

let result = document.write(“test”)

alert(result)

--------------

function으로 만드는 함수는 끌어올려진다(호이스팅 가능)

a( );

function a( ){alert(“A함수 실행함”)}

변수로 만드는 함수는 끌어올려지지 않는다.(호이스팅 불가)

a( );

let a = function ( ){alert(“A함수 실행함”)}

---------

**화살표함수**

Function 대신 ( )=>{ } 화살표로 써줄 수 있다.

----------

**생성자함수**

----------

**브라우저객체**

window 최상위 객체(생략가능)

document window 객체 안에 문서컨텐츠를 표현

location 브라우저의 페이지정보를 표현

console 개발자 툴 안

navigator 브라우저의 정보를 표현

window 최상위객체

let window={document: {}, navigator: {}, console:{}, location:{} }

let window={document : {}, querySelector : function(){}, querySelectorAll : function(){}, h1Element : {}, ulElement : {}}

document / navigator / console / location 하위객체

**태그 선택하기 – document.querySelector(“css선택자”)**

let 변수명 = document.querySelector(“#a”)

변수명.style.color=”darkred”

변수명.style.스타일속성=”스타일값”

변수명.style.transform=”translateX(-500px)”

**태그들 선택하기 – document.querySelectorAll(“css선택자”)**

**태그의 속성명과 속성값을 설정 – setAttribute(“속성명”, ”속성값”)**

변수명.setAttribute(“id”,”at”) => 선택한 태그의 id속성에 at라는 값을 설정

**태그의 속성값을 리턴 – getAttribute(“속성명”)**

Console.log(변수명.getAttribute(“id”)) => 선택한 변수의 id속성에 저장된 속성값을 리턴

**태그의 컨텐츠 교체하기 – innerHTML**

변수명.innerHTML = “문서데이터”

**문서객체**

window.onload–문서를 끝까지 읽은 뒤 실행

**window.onload = function ( ) {실행할 소스코드}**

**클래스 추가하기 – classList.add(“”)**

지정한 클래스명.classList.add(“추가할클래스명”)

**클래스 제거하기 – classList.remove(“”)**

지정한 클래스명.classList.remove(“제거할클래스명”)

부모태그 선택 – 지정할 변수명 = **지정한 변수명.parentElement**

자식태그들 선택 – 지정할 변수명 = **지정한 변수명.children**

다음형제태그 선택 – 지정할 변수명 = **지정한 변수명.nextElementSibling**

\*연달아 쓸 수 있다.

이전형제태그 선택 – 지정할 변수명 = **지정한 변수명.previousElementSibling**

**이벤트 – click…**

**지정한 변수명.addEventListener(“이벤트형태”, 익명함수function(){} )**

-------

기능추가

1.html

2.완성된css

3.클릭했을 때 완성된 css (class를 추가하거나 삭제)

--------------------

**제이쿼리**

\*연결방법

**주의사항 : 제이쿼리 먼저 연결 후 자바스크립트를 연결해야한다.**

(1) 파일 다운로드 후 head태그 안에 상대경로로 연결(인터넷 불필요)

(2) 웹상의 절대경로를 그대로 복사해서 head태그 안에 연결(인터넷 연결필요)

**태그선택하기 $ ( “** 태그 or 태그명 **” )**

**스타일적용 . css ( “** 속성 **” , ”** 속성값 **” )**

**중복적용가능(체이닝기법) . css ( “** 속성 **” , ”** 속성값 **” ) . css ( “** 속성 **” , ”** 속성값 **” )**

**$ ( document ) . ready ( function ( ) { 소스코드 } )**

모든 문서를 다 읽은 후 제이쿼리 기능 실행

**attr ( “** 속성명 **” , “** 속성값 **” )**

속성을 제어한다. Html 내의 속성값 교체

**태그내용교체(문법불포)** .text(“ “)

**태그내용교체(문법포)** .html(“ “)

**클래스추가** .addClass(“클래스명”)

**클래스제거** .removeClass(“클래스명”)

**자식선택** .children()

**부모선택** .parent()

**다음형제선택** .next()

**이전형제선택** .prev()

**순번직접선택** .eq(0)

**클릭이벤트** .click()

**마우스올렸을 때** .mouseover ( function ( ) { } )

**마우스내렸을 때** .mouseout ( function ( ) { } )

**앞에 선택된 태그의 부모태그기준 순번을 리턴** .index()

**슬라이드무브** slideMove(변수명)

**선택한 태그가 갖는 기본기능을 억제함 (a태그 스크롤 자동업되는거 삭제해줄 때)**

$(“선택자”).click ( function ( e ) { e.preventDefault ( ) } )

**선택한 태그에 on이라는 클래스가 있다면 사용된 자리에 true를 리턴하고,**

**on이라는 클래스가 없다면 사용된 자리에 false를 리턴하는 함수**

hasClass(“on”)

**이벤트 버블링 : 신호를 위쪽 부모로 전달하지 않는다. (자식한테 적용)**

**return을 사용하면 함수는 그 즉시 종료가 된다.**

return false

**scroll**

**$ ( window ) . scroll (익명함수)** 스크롤바가 움직일 때마다 입력한 동작이 실행되도록 하는 함수

**$ ( window ) . scrollTop( )** 스크롤바가 위에서 얼만큼 내려와있는지 계산해서 수치(픽셀값)을 리턴

**animate**

animate ( { 변형할 스타일 및 속성 : 속성값 } , 시간 )

animate ( { 변형할 스타일 및 속성 : 속성값 , 변형할 스타일 및 속성 : 속성값 } , 시간 )

stop( ) 애니메이션 정지

clearQueue( ) 대기중인 애니메이션 삭제

delay( ) 지연

b1클래스의 탑값

($(".b1").offset().top)

  $("html,body")

  .animate({scrollTop:b2top},500)

**마우스 휠이벤트 – 개발자창에서 originalevent wheeldelta 키값으로 확인가능**

$("wrap>div").on("click", function () { })

$("wrap>div").on("wheel", function () { })

$(".wrap>div").on("wheel click", function () { })

파이어폭스에서는 wheel 이벤트명칭을 사용하지 않기 때문에, 아래와같이 사용한다.

파이어폭스에서는 originalevent -> detail 키값으로 확인가능

$(".wrap>div").on("wheel DOMMouseScroll", function (event) {

    let E = event.originalEvent

    let delta = 0;

    if (E.detail) {delta = E.detail \* -40}

    else {delta = E.wheelDelta}

    if (delta < 0) {

      if ($(this).next().length != 0)

{let posTop = $(this).next().offset().top

      $("html,body").stop().animate({ scrollTop: posTop }, 1000)}

} else {

      if ($(this).prev().length != 0)

{let posTop = $(this).prev().offset().top

      $("html,body").stop().animate({ scrollTop: posTop }, 1000)}

}

    return false

  })