자바스크립트기초

변수 , 자료형 ,연산자, 반복문 ,조건문 ,배열, 함수

1. 외부 스크립트 <script src="js/basic.js" defer></script>

JavaScript ->브라우저 객체 windows -> 동적으로 구조 수정, 스크립트 언어

자바스크립트 API -> 브라우저에서 사용자에게 제공되어 지는 객체, 함수 등(미리 만들어진)

console

-> 브라우저에서 제공 객체, 스크립트 실행 체크, DOM(화면구성하는 객체) 에 접근 할 수있다, 스크립트 테스트

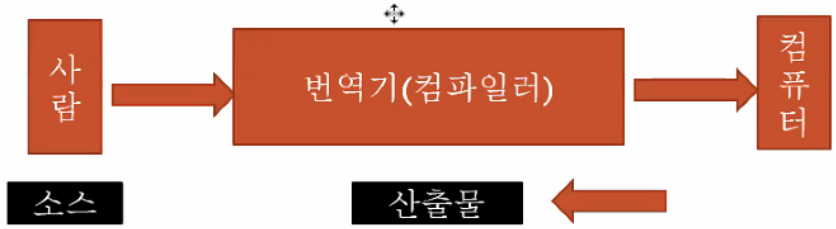
-> 스크립트 오류 체크

자바스크립트 실행 -> 새로고침 -> 동기방식

window.onload=function(){실행문}; -> body 안에 요소가 브라우저에 로드 된 이후 {} 실행

defer -> 외부 스크립트 적용 -> body 안에 요소가 브라우저에 로드된 이후 {} 실행

프로래밍 언어(개발) ->프로그램을 만드는 언어(컴퓨터 대화)




1. 컴파일언어: 컴퓨터 언어로 번역을 통해 실행 (C,java,,)

**영어 번역서(코드 전체를 번역해서 실행) -> 완전히 번역해서 출판

2.비컴파일언어(인터프리터 언어): 컴파일(번역)이 없이 실행 되는 언어.

하나의 실행문을 그 때 그 때 실행 -> 통역사

** 자바스크립트(스크립트언어)

컴퓨터

1.기본 구성 요소

하드웨어 -> 모니터 ,키보드 ,ram ,저장장치 -> 눈에 보이고 만져지는 것

소프트 웨어 - >하드 웨어 조정, 시스템 운영 -> 만져지지않음

-> 시스템 소프트 웨어 -> 컴퓨터 기본 운영체제, 기본 설정을 제어하는 소프트웨어

-> 응용 소프트 웨어 -> 시스템 소프트 웨어(기본)를 제외한 컴퓨터에 추가 소프트 웨어

-> 포토샵, 메모장,웹브라우저, 메모장

2.프로그래밍 언어 -> 하드웨어를 조정 하거나 소프트웨어를 만드는 언어 -> 프로그램을 만드는 언어

1) 컴파일 언어 -> 번역기, 소스 코드 모두를 산출물로 번역 ,한번에 전체 번역

ex) C+, C, java

컴퓨터 언어로 번역을 통해 실행

**영어 번역서(코드 전체를 번역해서 실행) -> 완전히 번역해서 출판

2) 비컴파일 언어 - > 번역기(통역사), 한줄씩 번역

ex) Javascript->jquery,react,vue,angluer,,jsp,swift, python

자바스크립트

1. 스크립언어 (비컴파일언어) -> 인터프린터(브라우저에 내장)
2. 이벤트 중심 언어 (마우스, 키보드등의 효과를 이용해서)
3. 객체 기반/ 지향적 언어 -> 웹브라우저 -> 자바스크립트 객체
4. 함수 기반의 언어
5. HTML 문서 어디에서든 사용 할 수 있다.

**head 사이에 위치 할때는 document 객체를 선택 시에는 반드시

window, onload() 함수안에 실행 시켜야 된다.

6. <script></script> 안에 자바스크립트 코드 작성

<script type="text/javascript'></script> -> 지금 사용HTML5<script></script>

<script src="js/"></script> -> 외부 스크립트 연결

변수: 하나의 data를 저장 할 수 있는 공간

저장공간/저장되어있는 값

data -> 숫자,문자열,객체 ,배열,null

1)변수 선언 -> 컴퓨터 메모리에 변수 공간을할당 ,초기화 data를 변수에 설정

-> 대부분의 프로그래밍 언어는 같은 코드 블록에서 한번만 초기화 할 수 있다.

2) 메모리 할당 - 컴퓨터 저장 공간 (기억공간 == 메모리 )할당

3) 변수 초기화 ->데이터

1-1) var -> ES6 이전

-> 호이스팅 적용

-> 함수 레벨 스코프

*변수 선언 키워드 ,함수 스코프

*엄격하지 못하다

*중복선언가능

-> 최근 사용 빈도 줄어듬

var btn1;// 변수선언

btn1 = window.document.getElementbyid("btn1");//변수 btn1객체 "?" 를 초기화했다

var btn1=window.document.getElementbyid("btn1");

1-2) let

-> ES6

-> 호이스팅 적용 안됨 -> 블록 레벨스코프(블록안에서 만 유효)

*변수 선언 키워드

**var에 비해 엄격 , 같은 이름 선언 불가능

변수명; // 변수 선언, let은 한번만 선언 할 수 있다.(특정 함수 영역)

let num1;//변수 num1 선언// > 실행 공간에서 변수 num1을 사용 할 수 있다,

num1=100;//변수 num초기화

1-3) const

-> ES6에 추가

-> 상수(변하지 않는 값)

-> 변수 선언 키워드

-> 호이스팅 적용 안됨

-> 실행영역에 한번만 선언

-> 한번 초기화 된 값을 변경 할 수 없다

-> 반드시 초기화를 해야된다

const num4=100;

**하나의 자료만 표시할 변수에 사용

데이터 (자료==data)종류 -> 변수에 저장 할 수 있는 자료(data)

data type(데이터 타입) -> 자료형

1.리터럴 -> 변하지 않는 데이터 하나

1) Primitive type(기본) - 변경불가

. Number -> 숫자 / 정수 또는 실수형 숫자 ex)42 혹은 3.14159

. String -> 문자열 ex) “안녕”

. Boolean -> true,False

. Null ->null,비어있다 / 어떤 값이 의도적으로 비어 있음을 표현

. undefined-> 선언을 했는데 초기화가 안됨 / 선언만 되고 값이 저장되어 있지 않은 최상위 속성

. Symbol-> ES5 ECMA2015 / 인스턴트가 고유하고 불변인 테이터형

.aroow

2.객체형(object) ->주소값으로 저장

array배열 -> 데이터를 순서대로 그룹화

object 객체-> 내장객체, 사용자 정의 객체

class-> 객체를 만들기 위한 설계도

함수 -> 실행코드를 미 저장

typeof 변수명;-> 변수의 타입을 알 수 있는 함수

변수명 작성규칙

1. 숫자로 시작하지 않는다->var 1aa;let0aa;
2. _$이외 특수문자를 사용하지 않는다 ->var %d;var a@a;
3. 공백을 허용하지 않는다 -> var a b;
4. 키워드 (예약어)를 사용하지 않는다(이미 사용되기로 약속된 문자열)

5. 한글 사용을 권하지 않는다
6. 대소문자를 구별한다
7. 소문자로 시작하기를 권한다

** 대부분의 프로그램  언어는 ; 으로 명령문을 종료한다.

자바스크립트는 ; 를 사용해도 되고 안해도 된다.

실습 시 명령문 (문장)의 끝을 ;를 반드시 붙여준다

3. function

**연산 -> 데이터들을 이용해서 프로그램을 처리하는 과정, 명령문을 처리하는 과정

연산자 (Operator) -> 연산기호

5+6 -> 피연산자(Operand), 항

산술 ->  +  - * /    %(나머지)

숫자 + 문자 -> 문자 + 문자

괄호안의 연산을 먼저처리한다

짝수 ->

비교 -> >,>=,<.<= -> 결과값 , true, false -> 조건문과 같이 많이 사용 한다

대입(할당) 연산자, 복합대입연산자

=" 오른쪽의 연산을 결과를 왼쪽 변수에 대입", 연산시 마지막 처리

논리

&& -> 조건이 모두 true일 때만 결과 값이 true

and, 논리곱

조건 1 && 조건2 결과 값

true && true true

true && false false

false && true false

false %% false false

|| -> 조건이 하나이상 true면 결과 값 true

or, 논리합

조건1 || 조건 2  결과 값

true || true  true

true || false  false

false || true  false

false || false  false

.항등 연산자 -> return 값 , 결과 값 true, false

== “같으냐?” 타입의 제외

=== “같으냐? 타입 포함

!= “ 같지 않지?” 타입은 제외

!==”같지 않지?”타입 포함

2)단항 연산자

증감

3)삼항 연산자

삼항?:

1) 단항 연산자 -> 반복문과 같이 많이 시용(항이 하나)

++ -> 1증가

i=0;

i++;// 선처리 후 증가

++i;// 선증가 후 처리

```
'use strict';
//단항 연산자
{
    let i=0;

    console.log(i++);// 선처리 후증가
    console.log(i);
    console.log(i++);// 선처리 후증가
    console.log(i);
    console.log(++i);//선증가 후처리
    console.log("=========");

    let j=10;
    // j의 값을 콘솔에 실행 시키고 J값을 1증가 시킨다.
    console.log(j++);// 선처리 후증가 10
    console.log(j);  // 11
      // j의 값을 1증가 시키고 콘솔에 실행 시킨다.
    console.log(++j);  //선증가 후처리


}
```

-- -> 1감소

i--;// 선처리 후 증가

--i;// 선증가 후 처리

2) 이항 연산자( 항이 두개)

.산술연산자 +.-.*,/,%

.비교연산 >,>=,<,<=

.대입(할당) 연산자 ,복합대입(할당) 연산자

+=,-=,*=,/=,%=

.논리 연산자

&&(and) | | (or)

3) 삼항연산자 -> 하나 밖에 없다.(항이 세개)

?:

조건식? 조건 true:조건false

```
let i=10;
 console.log(i<10?1:0);

if(조건){
                //조건true
}
if(조건){
                //조건true아닐때
}
```

변수 + 문자열

num+'+' -> 10+

'$(num)+' -> 10+

input:submit

-> 폼의 사용자 입력값들을 action=" joinOk.do"서버 페이지로 post 방법

자바스크립트 적용시

input:button

->이벤트를 적용해서 함수를 호출

-> 회원가입 처리 (실행)

1.joinOk  버튼 클릭

2.userId 유효성 검사 false -> 다시 입력, alert(경고창)

3.userPw 유효성 검사 false -> 다시 입력, alert(경고창)

4.joinForm -> 전체 폼을 실행 joinForm.submit();// 서버에 전송

1.찾아서 -> DOM(Document)-> 브라우저 화면 구현하는 객체

2.이벤트 -> click, mouse down

3.구현 -> 함수 -> 미리 사용하기 위해서 만들어 놓은 명령문 (처리 ,실행)

function 함수명(){명령문;}

let userId=document.getElementById('userId');// id 가  userid

userId.value;//form 안의 Input요소에 사용

userId.value.length;//input 요소의 값의 길이

Form -> 사용자의 입력 정보를 서버에 전송

함수 -> 미리 어떤 목적으로 만들어 놓은 명령문 (처리문) 구조화 시킨 타입

1. 내장 함수 -> 브라우저에 이미 만들어진 함수 ->

   setinterval(함수, 시간); // -> 함수를 시간(1000 -> 1초) 후에 반복 실행

2. 사용자 정의 함수

```
function 함수명()
        //처리문 ,실행 명령문

함수명();// -> 함수 호출(call)
```

ECMA 스크립트 -> 자바스크립트를 표준화 시킨 스크립트

- ECMA 스크립트 주요 버전별 특징

ES2009 (ES5)

1. 배열의 forEach,map,filter,reduce,some,every 메서드 추가
2. object에 대한 getter ,setter 메서드 추가
3. 자바스크립트 strict 모드 지원(이전 버전보다 엄격한 문법 검사)"use strict";
4. JSON 지원 - Ajax XML 대체

ES2015 (ES6)

1. let,const키워드추가 -> 이전에는 var 사용

2. block scope를 가진 let과 const(상수) 키워드를 추가
3. arrow 문법 지원 -> 함수처리의 효율성 (자바의 람다와 유사)
4. arrow 문법은 코드가 간소화, this를 바인딩 하지 않는다.