:: HTML ::

* 주석 : <!-- (주석 내용) -->
* < > : tag
* <meta> : <meta> 태그는 해당 문서에 대한 정보인 메타데이터를 정의할 때 사용

<meta /> 를 이용하여 닫는 효과

* <title> </title> : 웹페이지의 제목
* <body> </body> : body 안에 웹페이지에 나타낼 data들을 작성
* 렌더링 : 웹 브라우저(chrome, Edge 등)가 html을 이해하여 그려주는 작업
* Open with live server : http://127.0.0.1:5500/1\_hello\_world/hello.html

Vs code 🡪hello.html

127.0.0.1 (ip 주소) : vscode가 돌고 있는 자기 자신의 컴퓨터

5500 : data packet을 분간할 수 있는 port 5500번지

Chrome 브라우저가 127.0.0.1.5500이 vscode 안에 있으니까

그곳의 1\_hello\_world/hello.html을 가서 보여주라고 요청

* <h1> </h1> : head Tag. 웹 페이지에 제목을 넣을 때 사용할 수 있다.

숫자가 작을수록 큰 숫자 (1~6)

* anchor Tag :

<a href="https://www.google.co.kr"> 구글 </a>

* 구글을 클릭하면 해당 주소로 점프
* Anchor tag의 가장 중요한 속성 : href : hypertext reference
* Input Tag

<input type=”text”/>

* <button> (button에 쓸 문자열) </button>
* <ul> </ul> : unordered list (순서가 없는 list)

<ul>

<li>첫번째</li> <!--list item-->

<li>두번째</li>

<li>세번째</li>

</ul>

* <oi></oi> : ordered list (순서가 있는 list)

<ol> <!-- ordered list -->

<li>첫번째</li> <!--list item-->

<li>두번째</li>

<li>세번째</li>

</ol>

* <hr> : horizontal rule (수평 자) : 줄 하나를 그어준다’
* <img src="./grass.jpg" alt=""> : 현재 폴더의 grass.jpg 파일을 보여준다

TTS : test to speech : 시각장애인을 위해 글을 읽어주는 기술.

사진은 못 읽어주기 때문에 alt(alternative)에 읽을 글을 넣는다.

* Division tag :

Open with server를 한 후 ctrl + shift + i 를 누르면

Chrome inspector가 나옴 element에서 html코드와 division을 확인할 수 있음

<div class="divclass" id="first"> div class=”(class명==같은family이다)”

<h1>H1 첫 번째입니다.</h1>

</div>

<div class="divclass" id="second">

<h1>H1 두 번째입니다.</h1>

</div>

<div class="divclass" id="third">

<h1>H1 세 번째입니다.</h1>

</div>이렇게 division을 나누어서 각 division 별로 원하는 design을 할 수 있다

* Style

<style>

div#first div#(div id명)

{

background-color: ; (속성 : 속성값)

}

</style>

<style>

div.divclass{ (div.(class명))

background-color: beige;

}

</style>

:: CSS ::

Html의 head에 link로 .css 파일과 linking한다.

<link rel="stylesheet" href="./(css파일명)">

.css 파일에는

* Division style

div#first

{

background-color:lightsteelblue;

display: inline-block;

width: 300px; (숫자와 단위를 붙인다)

}

div#second

{

background-color: peachpuff;

display: inline-block;

width: 400px;

}

div#third

{

background-color:sandybrown;

display: inline-block;

width: 500px;

}

* Margin, border, padding

Margin : 나를 감싸고 있는 outer divison에서 나쪽으고 설정값만큼 margin을 둔다

Padding : 나를 기준으로 outer divison쪽으로 설정값만큼 margin을 둔다

Border : division에 박스를 친다

<style>

        div#outer {

display: inline-block;

            background-color: blueviolet;

            margin: 100px;

            border: 2px solid #ff0000; (Solid==실선)

        }

        div#inner{

display: inline-block;

            background-color: aqua;

            /\*margin: 50px;\*/

/\* margin: 50px 10px;\*/

/\*margin: 50px 10px 90px;\*/ /\*위 좌우 아래\*/ margin: 50px 10px 40px 100px; /\*상 우 하 좌 (시계방향)\*/

padding : 100px; (나를 기준으로 100px씩 몸집을 키운다)

        }

 </style>

:: JavaScript ::

* 문자열을 표현하는 3가지 방법

<script>

      // 1. 출력 ---> chrome -> 개발자도구 -> console

      //첫번째 큰따옴표 // console.log("이것은 자바스크립트입니다.");

      //두번째 작은따옴표// console.log('이것은 자바스크립트입니다.');

      //두번째 작은따옴표// 이렇게 하면 error. ()

      // console.log('이것은 자바스크립트입니다.

      // by backtick');

      //세번째 backtick // backtick이 다른 점 : 이렇게 줄바꿈한 그대로로 나타난다

            console.log(`이것은 자바스크립트입니다.

            by backtick`);

        </script>

* Typeof

console.log(typeof("hello"));  // string

console.log(typeof(4));  // number

console.log(typeof(4.3));  // number

4랑 4.3이랑 똑 같은 number형이다.

console.log('hello의 자료형은 '+typeof("hello"));  // string

       console.log('4의 자료형은 '+typeof(4));  // number

       console.log('4.3의 자료형은 '+typeof(4.3));  // number

+ : string을 연결해서 붙이는 연산자. String concatenation operator

console.log('string concat의 문자열 더하기 숫자 결과는 ' +('123'+5));

// 이 결과는 1235로 나온다. 하지만 내가 원했던 결과는 128이라면

       // 문자열을 숫자로 바꿔주는 함수 : parseInt(문자열) -- > 정수

       // parseInt('123') // '123' --> 123

      console.log('parseInt의 결과는 : ' +(parseInt('123')+5));

* 변수 (const, let)

const : 한 번 대입하면 변경 불가

// const 변수 error

const a=1;

a=2;

let : 일반 변수

let b=1;

console.log('b의 초기값:' +b);

b=2;

console.log('b의 변경값 : ' +b);

// 제곱

const e=2\*\*3; // 2의 3승

* QuerySelector

H1 태그가 하나인 경우

<h1>여기입니다.</h1>

        <script>

            const userMsg = prompt();  // 사용자 입력 : prompt()

            console.log('사용자가 입력한 메시지는 ' +userMsg); // debugging 용도

            document.querySelector('h1').textContent=userMsg;

            // document는 전체 코드,

            // querySelector('h1')은 6번째줄을 가리킴,

            // textContent는 여기입니다.를 가리킴. 그곳에 userMsg 변수를 출력한다.

        </script>

H1 태그가 여러 개인 경우(id를 사용하여 어디인지 지정한다)

<body>

        <h1 id="h11">여기입니다.</h1>

        <h1 id="h12">여기일까?.</h1>

        <h1 id="h13">요기인가?</h1>

        <script>

            const userMsg = prompt();

            console.log('사용자가 입력한 메시지는 ' +userMsg);

            document.querySelector('h1#h12').textContent=userMsg;

// h1태그의 id : h12에 출력하겠다.

            // document는 전체 코드,

            // querySelector('h1#h12') h1태그의 id가 h12인 곳을 가리킴

            // textContent는 여기입니다.를 가리킴. 그곳에 userMsg 변수를 출력한다.

        </script>

    </body>

// 사용자가 입력한 숫자가 짝수?홀수? 판별하여 h11에 출력하기

<body>

        <h1 id="h11">여기입니다.</h1>

        <h1 id="h12">여기일까?.</h1>

        <h1 id="h13">요기인가?</h1>

        <script>

            // 사용자가 입력한 숫자가 짝수?홀수? 판별하여 h11에 출력하기

            const userMsg=prompt();

            const userNumber=parseInt(userMsg); // 사용자 입력 문자열 ---> 정수

            const result=userNumber%2;

            let msg="";

            if(result==1) // 홀

            {

                msg=userMsg+"는 홀수입니다";

            }

            else // 짝

            {

                msg=userMsg+"는 짝수입니다";

            }

            document.querySelector('h1#h11').textContent=msg;

        </script>

    </body>

* QuerySelectorAll

const allh1=document.querySelectorAll('h1'); // h11, h12, h13이 모두 들어있다

       allh1.forEach((elem,index)=>{elem.textContent=userMsg;});

       // allh1에 있는 h11, h12, h13 에 대해서

      // elem : h11, index : 0 (한 번 썼으면 다음꺼)

      // elem : h12, index : 1 (한 번 썼으면 다음꺼)

      // elem : h13, index : 2

      // => 갖고 온다는 의미, 갖고 온 elem로 {무엇을 할건지}

* Boolean형

let k=true;

            console.log('k : ' +k)

            console.log('k type : '+typeof(k));

            k=false;

            console.log('k : '+k);

            console.log('k type : '+typeof(k));

* 비교연산자

console.log('1==1: ' +(1==1));   // true.

            console.log('\'1\'==1 : ' +('1'==1)); // 원래는 false인데,

                                               // javaScript 형변환을 하여 서로 같은 자료형으로 만든다

            console.log('\'1\' === 1 : '  +'1'===1); // === 연산자는 자료형까지 비교. 그래서 false.

            console.log('1!=1: ' +(1!=1));   // flase

            console.log('\'1\'!=1 : ' +('1'!=1)); // false

            console.log('\'1\' !== 1 : '  +'1'!==1); // true

* Undefined VS Null

let k;

            console.log('k is : ' +k); // undefined, 초기화가 되지 않았음

            let x=null;  // 초기화했지만 값이 없음

            console.log('x is : ',+x);

* If문

const score=parseInt(prompt());

            if(score >= 90){

                document.querySelector('h1#grade').textContent='A';

            }

            else if(score>=80){

                document.querySelector('h1#grade').textContent='B';

            }

            else

                document.querySelector('h1#grade').textContent='C';

* Switch 문

switch(score) {

                case 100:

                    document.querySelector('h1#grade').textContent='완벽 한 실력';

                    break;

                case 90:

                    document.querySelector('h1#grade').textContent='충분 한 실력';

                    break;

                default:

                    document.querySelector('h1#grade').textContent='나머 지 공부 필요';

            }

* 삼항연산자

// 50 --> pass, otherwise --> fail

            const msg= (score>=50) ? 'pass' : 'fail' ;

            document.querySelector('h1#grade').textContent=msg;

* For 반복문

const num=parseInt(prompt());

            let msg='';

            for(let i=0 ; i<num ; i++){

                msg+='hello';

            }

            document.querySelector('h1#grade').textContent=msg;

* While 반복문

const num=parseInt(prompt());

            let msg='';

            let i=0;

            while(i<num){

                msg+='hello ';

                i+=1;

            }

            document.querySelector('h1#grade').textContent=msg;

* Array (배열 element에 들어갈 수 있는 자료형이 다양하다)

const myArray=[1,2,'apple','banana',3.14,true];

                // 길이

            console.log('myArray length : '+myArray.length);

                // 내용

            console.log(myArray);

* Array in array .. in array

const myArray2=[1,[3.14,true,'banana',10],'apple'];

                // 길이

            console.log('myArray2 length : '+myArray2.length);

                // 내용

            console.log(myArray2);

* Array 출력

// array 출력

            const myArray=[1,2,'apple','banana',3.14,true];

            for(let i=0;i<myArray.length;i++) {

                console.log(myArray[i]);

            }

            // array in array 출력

            const myArray2=[1,[3.14,true,'banana',10],'apple'];

            for(let i=0;i<myArray2.length;i++) {

                console.log(myArray2[i]);

            }

            // array in array만 출력

            for(let i=0;i<myArray2[1].length;i++) {

                console.log(myArray2[1][i]);

            }

* 지우기. Pop(), shift()

const myArray=[1,2,'apple','banana',3.14,true];

            console.log('before : ' +myArray);

            let value = myArray.pop(); // pop하면 맨 뒤에 있는 true라는 값만 제거됨

            console.log('pop value : '+value);

            value=myArray.shift(); // shift하면 맨 앞에 있는 1만 제거됨

            console.log('shift value : '+value)

            console.log('after : '+myArray);

* 추가하기. Push()은 뒤에서, unshift는 앞에

const myArray=[1,2,'apple','banana',3.14,true];

            console.log('before : ' +myArray);

            myArray.push('orange'); // push하면 맨 뒤에 값 추가

            myArray.unshift(false); // unshift하면 맨 앞에 값 추가

            console.log('after : '+myArray);

* Splice

const myArray = [1, 2 ,'apple','banana',3.14,true,false,2];

            //중간에 있는 element 삭제

            myArray.splice(1,1) // 1번 인덱스부터 1개를 삭제

            myArray.splice(1,2); // 1번 인덱스부터 2개를 삭제. 2랑 'apple'

            myArray.splice(1); // 1번 인덱스부터 끝까지 삭제. 1만 남는다

console.log(myArray);

// 삭제 후 추가

const myArray = [1, 2 ,'apple','banana',3.14,true,false,2];

            myArray.splice(1,1,'orange');

            //  1번 인덱스에서 1번째꺼를 삭제 후 'orange'를 넣는다

            myArray.splice(1,1,'orange','mellon');

            //  1번 인덱스에서 1번째꺼를 삭제 후 'orange'와 'mellon'를 넣는다

            console.log(myArray);

// 단순 추가

myArray.splice(1,0,'orange'); // 1번 인덱스에서 안지우고 'orange'를 추가한다

                               // 1, 'orange', 2, 'apple' .. 이렇게 된다.

        // splice로 unshift와 같은 효과를 낼 수 있다.

        myArray.splice(0,0,'orange');

       myArray.unshift('orange');

       // splice로 push와 같은 효과를 낼 수 있다.

        myArray.splice(myArray.length,0,'orange');  // 맨 마지막에 orange 추가

         myArray.push('orange');

        // splice로 pop과 같은 효과를 낼 수 있다.

          myArray.splice(myArray.length-1,1);

          myArray.pop();

 console.log(myArray);

* Includes. 배열 안에 해당하는 값이 있는지 확인. True or False

console.log('Apple이 있냐? '+myArray.includes('apple'));

console.log('orange이 있냐? '+myArray.includes('orange'));

if(myArray.includes('apple')===true){

     console.log('apple 있음');

  }

else{

     console.lo('apple 없음');

    }

* indexOf. 해당하는 값이 어디있는지? 없으면 -1

// indexOf. 해당하는 값의 index가 어디있는지? 없으면 -1

console.log('apple이 있냐?' +myArray.indexOf('apple'));  // 2

console.log('orange이 있냐?' +myArray.indexOf('orange')); //-1

// indexOf('apple',fromindex) // fromindex부터 'apple'을 찾아라 // \*\*(fromindex --> 끝)

console.log('apple이 있냐?' +myArray.indexOf('apple',0));  // 2

console.log('apple이 있냐?'+myArray.indexOf('apple',2));  // 2

console.log('apple이 있냐?'+myArray.indexOf('apple',3));  // -1

  // lastIndexOf('apple',fromIndex) // 'apple'이 여러개가 있는데, 마지막 인덱스를 출력하라

\*\*\*(처음 <-- fromindex) \*\*\* myArray=[1,2 ,'apple','banana','apple',3.14,true,false,'apple',2,'apple'];

console.log('apple이 있냐?' +myArray.lastIndexOf('apple')); // 10

console.log('apple이 있냐?' +myArray.lastIndexOf('apple',8)); // 8

console.log('apple이 있냐? '+myArray.lastIndexOf('apple',1)); // -1

* 함수 만들기

(1)

<script>

            function sayHello(){

                console.log('안녕하세요');

            }

            sayHello();

        </script>

(2) : Arrow function , 가장 javScript다운 방법.

// k라는 변수에 () => {console.llg('안녕하세요2');} 라는 함수가 있다}

       const k = () => {console.log('안녕하세요2');}

       k();

const k2 = (name) => {console.log(name+'씨, 안녕하세요2');}

       k2('이형섭');

(3)

const v = function() {console.log('안녕하세요3');}

            const v2 = function(name) {console.log(name+'씨, 안녕하세요3');}

            v();

            v2('이형섭');

* 함수 return값

function sayHello(){

                console.log('안녕하세요');

                // return '바이!';

                // return ['바이!',10,false];

            }

// javascript에서 반환값은 1개이다.

            // 여러개 반환하고 싶으면 array 이용한다.

            console.log('sayHello의 반환값 : '+sayHello());  // 반환값이 없으면 undefined, == void

* Parameter VS Argument

function sayHelloWithParameter(name,age){

                console.log(age+'살, ' +name+'씨, 안녕하세요');

            }

            sayHelloWithParameter('찰스',20);

            sayHelloWithParameter('찰스',20,false);

            // 이렇게 해도 false는 짤리고 찰스,20만 가져간다

            sayHelloWithParameter('찰스');

            // 이렇게 해도 정상동작한다. name : 찰스, age : undefined

// 배열로 넘기기 C/C++과 달리 typechecking이 엄하지 않다.-->webBrowser

            function sayHelloWithParameter2(values){

                console.log(values[1]+'살, ' +values[0]+'씨, 안녕하세요');

            }

            sayHelloWithParameter2(['찰스',20]);

* Object literal

// object literal

// : 배열 ['찰스',20,true]에 index 번호를 지정할 수 있다

//        0      1    2 가 아니라 --> name, age, tf 이런 식으로..

 // const person =['찰스',20];  --> array

 const person ={ name:'찰스', age:20  }; // object literal

 // person의 name이라는 속성의 값이 찰스.

 // person의 age라는 속성의 값이 20.

console.log('person name : '+person.name);

console.log('person name : '+person['name']);

person.age=35;

console.log('person age : '+person.age);

console.log('keys : '+Object.keys(person));

// person이라는 object literal 안에 담긴 속성의 이름들을 출력

console.log('keys value : '+Object.values(person));

// person이라는 object literal 안에 담긴 값들을 출력

// object literal의 key들을 뽑아보기. foreach 사용

const showOL=(ol)=>{

        const keys=Object.keys(ol);

              keys.forEach((elem,idx)=>{

                    console.log(elem+' ---> '+ol[elem]);

                });

            };

      showOL(person);

// for문 사용(C언어식)

const showOL2=(ol)=>{

                const keys=Object.keys(ol);

                for(let i=0;i<keys.length;i++){

                    console.log(keys[i]+' ===> '+ol[keys[i]])

                }

            };

            showOL2(person);

* object literal에 값 뿐만 아니라 method(=function)도 추가.

person.sayHello = () => {console.log('Hello!!! ');};

person.sayHello2= (v) => {console.log(v+'! Hi!');};

person.sayHello();

person.sayHello2('이형섭');

 person.areYouOlderThan = (\_age) => {

         if(person.age>\_age) {

             return true;

             }

             else

                return false;

            };

            const check\_age=40;

console.log('Are you older than '+check\_age+'? '+person.areYouOlderThan(check\_age));

* object literal 안에 object literal 추가하기

const a={

     name:'찰스',

     detail: {

          name:'톰',

           age:10

      }

}

console.log('--->'+a.detail.name)

* object literal 비교

const a={name: '찰스', age:20};

const b={name: '로라', age:25};

const c={name: '찰스', age:20};

const d=a;

console.log('a===c : '+(a===c));

// 다르다. 왜? memory상 다른 곳에 저장되어있다

console.log('a===d : '+(a===d));

// 같다. 왜? d=a라는 연산이 a가 있는 곳에 d라는 이름이 붙는다.

// 그렇다면 const a=1; const b=1;

// a===b : true --> 왜? a가 1을 가리키고 있고, b가 1을 가리키고 있기 때문에 true이다.

* Object literal 내용 비교

const a2={name:'찰스', age:20};

const b2={name:'찰스', age:20};

const cmp\_ol=(\_x1,\_x2) => {

if((\_x1.name === \_x2.name) && (\_x1.age === \_x2.age))

   return true;

else

  return false;

}

console.log('com\_ol(a,c) : '+cmp\_ol(a,c));

console.log('com\_ol(a,c) : '+cmp\_ol(a,d));

* Object literal의 구조를 모를 때, 비교

// key들의 개수가 같은지?

// key들의 값이 같은지?

const cmp\_ol\_2=(\_x1,\_x2) => {

let \_x1\_keys=Object.keys(\_x1);

let \_x2\_keys=Object.keys(\_x2);

let isEqual=true;

if(\_x1\_keys.length!==\_x2\_keys.length)

isEqual=false;

\_x1\_keys=\_x1\_keys.sort();

\_x2\_keys=\_x2\_keys.sort();

\_x1\_keys.forEach( (elem,idx) => {

if(\_x1\_keys[idx]!==\_x2\_keys[idx])

      {

        // return false; 로 하면 안된다.

          // forEach기 때문에 전체함수에서 return하는 것이 아닌,

      // arrow function에서만 return하기 때문에 다음꺼를 계속 비교하기 때문이다.

        // 그래서 하나라도 다르면 isEqual 변수를 flag처럼 사용하여 false로 세팅.

        isEqual=false;

      }

if(\_x1[\_x1\_keys[idx]]!==\_x2[\_x2\_keys[idx]])

       {

            isEqual=false;

       }

}

);

return isEqual;

}

console.log('cmp\_ol\_2(a,b) : '+cmp\_ol\_2(a,b));

console.log('cmp\_ol\_2(a,c) : '+cmp\_ol\_2(a,c));

* Event & listener

Event : 사용자가 컴퓨터에 뭔가를 요청

Listener : 그 요청을 실행하여 사용자에게 결과를 반환해줄 함수

* Functional programming & higher order function

프로그램 작성시 function을 argument나 return값으로 돌려줄 수 있는 언어를 functional programming 언어라고 한다. 자바스크립트는 functional programming을 지원하는 언어이다.

numFunc이라는 함수는 n을 인수로 받고, 함수를 return한다.

위의 numFunc과 아래의 numFun은 똑 같은 higher order fucntion이다.

const numFunc= (n) => {

                return ()=> {

                };

            };

            const numFunc = (n) => () {

            };