day01 -

2023년 2월 6일 월요일 오전 10:42

- 자바스크립트(Javascript)
  1. 퍼즐 조각처럼 코드 형태로 HTML 페이지에 내장된다.
  2. 컴파일 과정 없이 브라우저 내부의 자바스크립트 처리기(인터프리터)에 의해 바로 실행된다.
     ※ 개발이 발전됨에 따라 컴파일 과정이 가능해졌으며, Node.js로 서버 환경을 구축한다.

```
[index.html]      [웹 브라우저]
<script>
  자바스크립트 코드     인터프리터 -> 자바스크립트 실행
</script>
```

- 웹 페이지에서 자바스크립트의 역할
  - 웹 페이지는 3가지(HTML, CSS, JS) 코드가 결합되어 작성된다.
  - 자바스크립트는 사용자의 입력을 처리하거나 웹 애플리케이션을 작성하는 등
    웹 페이지의 동적 제어에 사용된다.

  - 사용자의 입력 및 연산
    HTML 폼은 입력 창만 제공하고,
    키, 마우스의 입력과 연산은 오직 자바스크립트로만 처리가 가능하다.

  - 웹 페이지 내용 및 모양의 동적 제어
    HTML 태그의 속성이나 콘텐츠, CSS 속성 값을 변경하여
    웹 페이지에 동적인 변화를 일으키는 데에 활용된다.

  - 브라우저 제어
    브라우저 윈도우의 크기나 모양 변경, 새 윈도우나 탭 열기, 다른 웹 사이트 접속,
    브라우저의 히스토리 제어 등 브라우저의 작동을 제어하는 데 활용된다.

  - 웹 서버와의 통신(Ajax)
    웹 페이지가 웹 서버와 데이터를 주고 받을 때 활용한다.

  - 웹 애플리케이션 작성(API)
    자바스크립트 언어로 활용할 수 있는 많은 API를 제공하므로,
    웹 브라우저에서 실행되는 다양한 웹 애플리케이션을 개발할 수 있다.

- JS 환경 구축

  Node.js 설치
  https://nodejs.org/en/download/releases/

==========

- 자바스크립트를 작성할 수 있는 위치
  1. HTML 태그의 이벤트 리스너 속성에 작성

- HTML 태그에는 마우스가 클릭되거나 키보드의 키가 입력되는 등의 이벤트가 발생할 때 처리하는 코드를 등록하는 리스너 속성이 있다.

2. <script></script> 태그 내에 작성

<head></head>, <body></body>, <body> 태그 밖 등 어디든 들어갈 수 있다.

3. 자바스크립트 파일에 작성

자바스크립트 코드를 확정자가 .js인 별도의 파일로 저장하고,

<script src=".js경로"> </script>

4. URL 부분에 작성

<a> 태그의 href 속성 등에도 자바스크립트 코드를 작성할 수 있다.

"javascript:"라는 키워드를 작성해야 href 속성에 JS 코드를 작성할 수 있다.

➢ 자바스크립트로 HTML 요소 출력

- 자바스크립트 코드로 HTML 요소를 웹 페이지에 직접 삽입하여 브라우저 윈도우에 출력되게 할 수 있다. 이 때 document.write() 혹은 document.writeln()을 사용한다.

	backtick(`):	여러 줄을 하나의 값으로 인식 시키거나 따옴표의 종류가 많아질 때 사용한다.
➢	with(객체명){};	중괄호 안에 모든 필드 앞에 객체명에 작성한 객체가 붙는다.
	ES6	템플릿 리터럴

➢ 자바스크립트 다이얼로그

- 사용자에게 메시지를 출력하거나, 입력을 받을 수 있는 3가지 다이얼로그가 있다.

-

프롬프트 다이얼로그	prompt("메시지", "디폴트값");	사용자가 입력한 문자열 값을 리턴하지만 아무 값도 입력하지 않으면 ""을 리턴, 취소나 강제로 닫으면 null을 리턴한다. "디폴트 값"은 생략이 가능하다.
확인 다이얼로그	confirm("메시지");	확인/취소 버튼을 가진 다이얼로그를 출력한다. 확인 true, 취소 혹은 강제로 닫을 시 false를 리턴한다.
경고 다이얼로그	alert("메시지");	

➢ 데이터 타입과 변수

1. 자바스크립트 식별자(이름)
   i. identifier(식별자)란 자바스크립트 개발자가 변수, 상수, 함수에 붙이는 이름이다.
   ii. 식별자를 만들 때 다음 규칙을 준수해야 한다.
      1) 첫 번째 문자: 알파벳, 언더바, $문자만 사용가능
      2) 두 번째 문자: 알파벳, 언더바, 0-9, $문자만 사용 가능

3) 대소문자 구분: data와 dAta는 다른 식별자이다.
4) 키워드(예약어) 사용 불가

2. 문장 구분
   i. 세미콜론으로 문장과 문장을 구분한다.
   ii. 한 줄에 한 문장만 있는 경우 세미콜론을 생략할 수 있다.

i = i+1	o
j = j+1;	o
k = k+1; m= m+1;	o
n = n+1 p = p+1	X

3. 주석
   i.

한 줄 주석	//
범위 주석	/* */

4. 데이터 타입
   i.

숫자 타입	number	
논리 타입	bool	
문자열 타입	string	
객체 레퍼런스 타입	object	Object, Array, Math, Date, …
undefined		타입이 정해지지 않은 것을 의미한다.
null		값이 정해지지 않은 것을 의미한다.

5. 변수
   i. -var 키워드로 선언한다,

```
var score;
var year, month, day;
var address="남양주";
```

   ii. var 키워드 없이 변수 선언할 수 있다.

```
age=20;
```

※ 대혼란

```
var age = 20;
var age = 10;
 age = 5;
```

모두 선언 또는 수정이다.

6. 지역 변수와 전역 변수
   i. 변수의 사용 범위(scope)에 따라서 전역 변수(global)과 지역 변수(local)로 나뉜다.
   ii. var로 선언 시, scope는 반드시 함수의 영역만 판단한다.

전역 변수	함수 밖에서 선언되거나 함수 안에서 var라는 키워드 없이 선언
지역 변수	함수 안에서 var 키워드로 선언.

- hoisting(호이스팅):
  - 선언의 위치에 상관없이 존재하면 메모리에 할당된다.
  - 초기화 작업은 호이스팅 되지 않고 선언만 호이스팅 된다.

※ var 키워드를 사용하지 않으면 호이스팅 되지 않는다.

- var 키워드를 사용하더라도 여러 번 선언이 가능하기 때문에 수정될 수 있다. 목적에 맞지 않는 사용이다.

i. global.x : Node.js 에서의 최상위 객체 (전역 객체)
ii. window.x : 브라우저에서의 최상위 객체(전역 객체)
iii. var x : 지역 변수와 이름이 겹치지 않는 전역변수
★iv. globalThis.x : global과 window 객체를 상황에 맞게 사용해주는 전역 객체.
혼동을 막기 위해 전역변수 선언에는 보통 globalThis로 선언한다.

7. 상수

i.

let	수정 가능	변수처럼 사용, 함수 영역 안에서 사용할 때
const	수정 불가능	상수로만 사용, 함수 영역 밖에서 선언할 때

var를 사용하면 중복 선언이 가능하고
let으로 선언하면 중복 선언이 불가능
하지만 값 변경은 가능하다.

➢ cont기 + ` -> console in VS code

➢

day02 -

2023년 2월 7일 화요일 오전 9:09

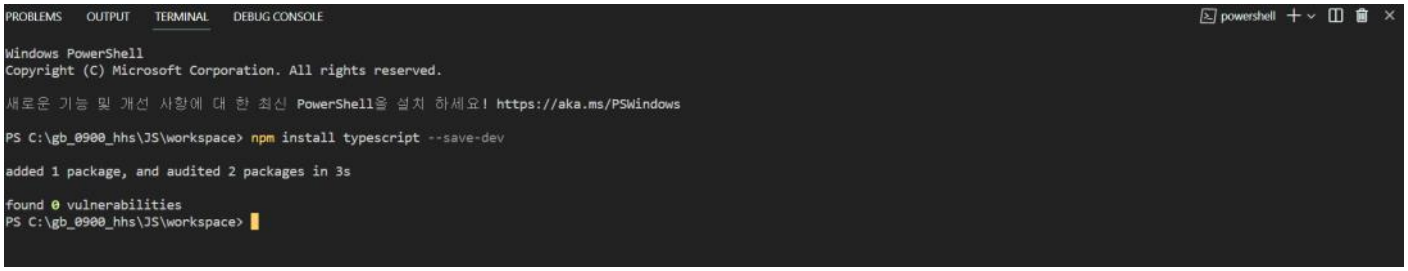
- 함수
  - function
  - 코드의 재사용을 목적으로 사용된다.
  -

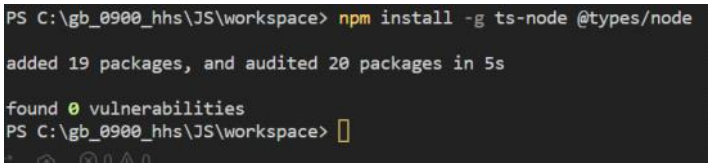
인자(parameter)	매개변수(선언부)
인수(argument)	매개변수에 들어가는 값(사용부, 호출부)

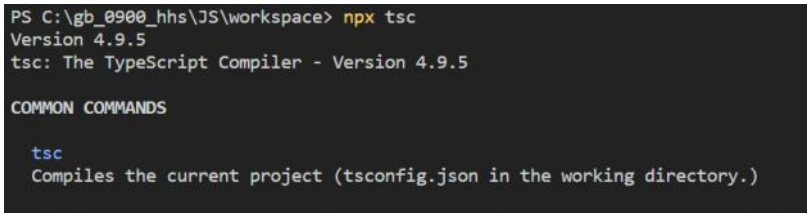
- 함수의 선언
  - function 식별자(arg1, arg2, arg3, ...) {
    실행할 문장;
    return 리턴값
    }
  -

function	함수 선언 표시
식별자 (identifier)	동사로 작성
parameter	여러 개 있을 때에는 콤마로 분리하고 자료형을 따로 작성하지 않는다. 생략가능
return	리턴 타입은 따로 작성하지 않고, return을 만나면 함수는 즉시 종료된다. 생략가능

- JS에서는 overloading을 지원하지 않는다.
  - 같은 이름으로 선언된 함수들 중 가장 아래에 선언한 함수가 적용된다.







- 관리자권한 실행

// typescript를 사용하고 싶을 때에는 다운로드 후 사용해야 한다.
// npm install typescript --save-dev
// npm install -g typescript

```
// npm install -g ts-node @types/node
// npx tsc
// Set-ExecutionPolicy Unrestricted (관리자 권한으로 실행)
// tsc --init
```

- JS에서는 매개변수의 개수에 맞추어 값을 전달 할 필요가 없다.
  - call back function, typescript
  - 만약 매개변수 개수보다 부족하게 값을 전달하면 값을 못 받은 매개변수에는 undefined가 들어간다.
  - 만약 매개변수보다 넘치게 값을 전달하면 잘린다.
- JS에서는 함수를 값으로 취급하기 때문에 매개변수로 전달이 가능하다.
  - 현재 함수에서 나온 결과 값을 다른 함수로 전달할 때에는 callback이라는 식별자로 결과 값을 전달받아 온다.

[task01.html]

```
<!--
  마우스를 올려보세요
  ----------(hr)
  여기에 마우스를 올리면 배경색이 노란색으로 변합니다.

  ※ 배경색 변경은 노란색으로 하고, 초기 배경색은 흰색으로 한다.
  ※ 태그이름.style.background = '색이름';
-->
```

[task02.html]

```
<!--
  월화수목금토일
  ----------(hr)
  000은 출근

  사용자에게 요일을 입력받고 월~목은 출근, 다른 날은 휴일을 출력한다.
  요일은 "월", "화", ... 형식으로 입력받는다.
-->
```

[task03.html]

```
<!--
  분석 결과
  ----------(hr)
  통과!

  정확한 암호가 입력될 때까지 계속 prompt()를 사용하여 암호를 입력받는다.
  암호는 "호랑이"이다.

-->
```

[task04.html]

```
<!--
  pr("%", 5);
  결과 : %%%%%

  위의 결과가 나오는 함수를 선언하여 사용한다.
  출력은 콜백함수를 사용한다.
-->
```

```
// 상품명, 가격, 개수를 전달받은 뒤 전체 금액을 출력한다.
// 1. 상품명과 가격, 개수는 A함수에서 전달받는다.
// 2. B함수에서는 상품명과 전체 금액을 전달받아서 출력한다.
// 3. A함수는 B함수를 callback함수로 전달받는다.
```

- 전역함수

eval()	문자열 형태의 수식을 전달받아 계산한다.	eval("2+3*6-7") == 13;
parseInt()	문자열을 정수로 전환	parseInt("34")===34;
isNaN()	값이 NaN이면 true 값이 NaN이 아니면 false	isNaN(value)
isFinite()	값이 NaN이면 false 값이 NaN이 아니면 true	isFinite(value)

-

(function( ){ })( )	선언과 동시에 함수 사용 가능
===	type과 값 모두 같은 지 확인
==	값이 같은 지 확인

➢

➢ 객체

- 객체의 고유한 속성을 프로퍼티(property)라고 부르며, 여러 프로퍼티와 쌍으로 표현된다.
- 객체가 호출되는 함수는 메소드라고 부른다.

```
account={name:'한동석', number: '110-11-1232142', password:'1234'}
account.name
account.number
account.password
account.["name"]
account.["number"]
account.["password"]

account.age = 20;

account.deposit=function(){...}

account.deposit
```

- 프로토타입
  - new 뒤에 나오는 생성자를 자바스크립트에서는 프로토타입이라고 부른다.
  - 프로토 타입은 함수로 선언하며 반드시 대문자로 시작한다.

- 프로퍼티
  - 프로퍼티 중 key에 ""가 있으면 JSON이고, 없으면 JS Object이다.
  - JSON 선언시 ``로 꼭 감싸주어야 한다.

JSON.parse(JSON 문자열)	JSON을 객체로 전환
JSON.stringify(객체)	객체를 JSON으로 전환

-

- JS의 Array 객체는 길이를 설정하지 않아도 원하는 인덱스에 원하는 값을 바로 추가할 수 있다.
- 또한 타입이 지정되어 있지 않기 때문에 다양한 타입도 동시에 담을 수 있다.

join()	원하는 구분점을 문자열로 전달하여 각 요소를 구분한 뒤 문자열로 리턴
push()	가장 마지막 인덱스로 요소 추가
slice(begin, end)	원하는 부분을 추출하기 위해 시작 인덱스(inclusive)와 마지막 인덱스(exclusive)를 전달한다.
slice(begin)	begin부터 마지막 인덱스까지 추출

forEach(callback)	반복문
	예) datas.forEach(function(data){console.log(data);});
	=== datas.forEach(data => console.log(data));
	
	※ forEach(callback(값, 인덱스, Array객체));
	예) datas.forEach(function(data, i, datas){
	console.log(datas[i]);
	또는

	datas[i] = Math.pow(data, 2); //제곱함수
	});

indexOf(value)	값을 해당 Array에서 찾은 뒤 인덱스 번호 리턴 못 찾으면 -1 리턴
fill("value")	원하는 값으로 배열 모두 채우기
map(callback)	기존 값을 원하는 값으로 변경하여 리턴 예) datas.map(function(data){return data/2;}).forEach(data => console.log(data));

split("구분점")	구분점을 기준으로 문자열을 나눠서 String배열로 리턴
	예) "월화수목금토일".split("").forEach(data => console.log(data));
	
	예) let dayOfWeek = new Array(7)
	dayOfWeek.push("월");...
	아래가 낫다
	let dayOfWeek = new Array()
	"월화수목금토일".split("").forEach(data => console.log(data));
	또는
	let dayOfWeek = new Array(7);
	dayOfWeek.fill("");
	"월화수목금토일".split("").forEach(function(data, i, array){
	dayOfWeek[i]=data;
	또는
	dayOfWeek.push(data);
	});

day03 -

2023년 2월 8일 수요일 오전 9:00

➢ 복습

- JS에서는 오버로딩을 지원하지 않는다.
  대신 함수 사용 시 매개변수의 개수를 맞출 필요 없다.

- 비밀번호 변경
  1. 로그인 후 비밀번호 변경 페이지
  2. 마이페이지에서 내 정보 수정 페이지

- 객체 선언 방법
  1. {}
  2. new Object()
  3. prototype : 딱 한번만 선언하자
     1) static선언
        User.prototype.number = 1;

- Array

join()	원하는 구분점을 문자열로 전달하여 각 요소를 구분한 뒤 문자열로 리턴
push()	가장 마지막 인덱스로 요소 추가
slice(begin, end)	원하는 부분을 추출하기 위해 시작 인덱스(inclusive)와 마지막 인덱스(exclusive)를 전달한다.
slice(begin)	begin부터 마지막 인덱스까지 추출

forEach(callback)	반복문
	예) datas.forEach(function(data){console.log(data);});
	=== datas.forEach(data => console.log(data));
	
	※ forEach(callback(값, 인덱스, Array객체));
	예) datas.forEach(function(data, i, datas){
	console.log(datas[i]);
	또는
	datas[i] = Math.pow(data, 2); //제곱함수
	});

indexOf(value)	값을 해당 Array에서 찾은 뒤 인덱스 번호 리턴 못 찾으면 -1 리턴
fill("value")	원하는 값으로 배열 모두 채우기
map(callback)	기존 값을 원하는 값으로 변경하여 리턴 예) datas.map(function(data){return data/2;}).forEach(data => console.log(data));

split("구분점")	구분점을 기준으로 문자열을 나눠서 String배열로 리턴
	예) "월화수목금토일".split("").forEach(data => console.log(data));
	
	예) let dayOfWeek = new Array(7)
	dayOfWeek.push("월");…
	아래가 낫다
	let dayOfWeek = new Array()
	"월화수목금토일".split("").forEach(data => console.log(data));
	또는
	let dayOfWeek = new Array(7);
	dayOfWeek.fill("");
	"월화수목금토일".split("").forEach(function(data, i, array){
	dayOfWeek[i]=data;
	또는
	dayOfWeek.push(data);
	});

reduce((accumulator, currentVal) => accumulator + currentVal)	누적 함수 accumulator(누적기) ->초기값도 따로 지정 가능 (initial value) accumualtor에 계속 쌓임 ※ reduce(total, data)일 경우, total값의 타입은 항상 data의 타입과 동일하게 설정된다.

- Math

Math.floor()	버림(소수점이 있다면 무조건)
Math.ceil()	올림(소수점이 있다면 무조건)
Math.square(n)	n제곱
Math.floor(Math.randon() * n)	0 ~ n-1 까지의 난수

- Date

getFullYear()	4자리 년도
getMonth()	0~11사이의 정수(0: 1월, 1: 2월, ..., 11: 12월)
getDate()	한 달 내의 날짜(28~31)
getDay()	한 주 내 요일(0: 일요일, 1: 월요일, ..., 6: 토요일)
getHours()	0~23사이의 정수
getMinutes()	0~59사이의 정수
getSeconds()	0~59사이의 정수
getMilliseconds()	0~999사이의 정수

getTime()	1970년 1월 1일 0시 0분 0초 기준 현재까지의 밀리초
setFullYear(year)	년도 저장
setMonth(month)	월 저장
setDate(date)	한 달 내의 날짜 값 지정
setHours(hour)	시간 저장
setMinutes(minute)	분 저장
setSeconds(second)	초 저장
setMilliseconds(ms)	밀리초 저장
setTime(t)	밀리초 단위인 t값으로 시간 저장

```
let date = new Date("2023-01-01 15:30:30");
console.log(date.getMonth());
console.log(date.getDate());
console.log(date.getHours());
console.log(date.getMinutes());
console.log(date.getSeconds());
```

- File

초기 선언문	let file = require('fs');
파일 출력	file.writeFile('경로', '내용', '인코딩', '콜백함수')
파일 읽기	file.readFile('경로', '인코딩', '콜백함수')

➢ Array실습1

- 아래의 세가지 방법으로 다 풀어볼 것

  1. 제어문
  2. callback 함수
  3. arrow expression

☐ 1. 1-10까지 Array 객체에 담은 후 짝수만 출력
☐ 2. 한글을 정수로 변경
☐ 3. 정수를 한글로 변경
☐ 4. 1-100까지 합 출력

➢ Array + Prototype 실습1

☐ 1. 동물원에 동물이 3마리 있다.
☐ 2. 각 동물별 정보는 이름, 종, 나이이다.

☐3. 동물원의 동물 전체 평균 나이를 구한다.

➢ Array + Prototype + file 실습1

☐1. 상품명, 가격, 재고를 프로퍼티로 담고 있는 Object를 3개 선언한다.
☐2. 3개의 Object를 1개의 Array 객체에 모두 담는다.
☐3. JSON으로 변경시킨다.
☐4. 각 상품별 총 가격을 구한 뒤 출력한다.
☐5. callback 함수를 사용한다.

☐6. 외부에서는 JSON 데이터가 전달된다.
☐7. shop.json에 변환된 JSON형식의 문자열을 작성한다.
☐8. shop.json을 읽어 온 뒤 Array 객체로 변환한다.
☐9. 총 가격과 총 재고 수를 Object에 담은 뒤 sum.json으로 출력한다.

➢ 콜백함수 실사용

```
<body>
    <h1>안녕하세요</h1>
    <button onclick="changeColor(function(color){console.log(color)})">클릭
</button>
</body>
<script>
    function changeColor(callback){
        const h1 = document.getElementsByTagName("h1")[0];
        h1.style.color="red";
        if(callback){
            callback(h1.style.color);
        }
    }
</script>
```

➢ file입출력 실습1

☐ 상품명, 가격, 재고를 JS 객체에 프로퍼티로 담는다.
☐ JSON 형식으로 변환한 뒤, product.json 으로 출력한다.
☐ product.json에 있는 JSON 형식을 JS Object 타입으로 변환하여 각 프로퍼티를 출력한다.