React 학습을 위한 Javascript

Web 페이지 제작에 필요한 언어들

HTML – Web 페이지의 내용을 취애 전체적인 뼈대를 구성하는 언어

CSS – HTML로 구성한 Web페이지 뼈대에 디자인 요소를 추가하기 위한 언어

Javascript(js) – Web 페이지에 다양한 기능(동적인 움직임)을 부여하기 위한 언어

js는 자바스크립트엔진(ex.V8 엔진)에 의해 실행된다.

자바스크립트 엔진은 웹 브라우저(크롬, 엣지, 사파리 등)에 기본 내장되어 있다.

따라서 웹 브라우저만 PC에 설치되어 있다면 js 코드를 간단히 실행시킬 수 있다.

JS의 자료형(Data Type), 변수, 상수

기본 자료형

Number(숫자), String(문자, 문자열), Boolean, Null, Undefined

참조 자료형

Object(객체), Array(배열), Function(함수)

> JS는 내부적으로는 자료형이 존재하지만 변수를 선언하거나 사용할 때에는 자료형을 지정하지 않는다.

변수 선언 -> let 변수명;

상수 선언 -> const 상수명;

> 이전 버전의 JS에서는 변수 선언 시 var 키워드를 사용하였다.

JS 변수 사용 예시

```
let num1 = 10;
let num2 = 5.5;
const num3 = 3; //값 변경 불가!
let name = “홍길동”;
let addr = ‘울산시’;

let isEmpty = true;
let isEmpty = false;

//Null Type : 아무것도 없다
let data = null;

//Undefined Type : 선언되지 않거나, 선언은 되었지만 초기값이 없다.
let data;
console.log(data); //undefined
```

JS 출력문

```
console.log('hello');
console.log("hello");
console.log(3);
console.log(1 + 5); //6
console.log('abc' + 'def'); // 'abcdef'
console.log('hello' + 5); // 'hello5'

//쉼표로 여러 변수 동시 출력 가능
let num1 = 5; let num2 = 10;
console.log(num1, num2); //5 10

//템플릿 리터럴 문법(백틱 사용)
let name = 'kim'; let age = 30;
console.log(`이름은 ${name}이고, 나이는 ${age}입니다.`);
```

Tip!

문자(열) -> 숫자 변경 : Number('10')

숫자 -> 문자(열) 변경 : String(10)

JS 연산자

산술 연산자 : +, -, *, / , %

복합 대입 연산자 : num +=3; // num = num + 3;

증감연산자 : ++num; num--;

논리연산자 : &&(그리고, and 연산), ||(이거나, or 연산)

부정연산자 : ! -> true는 false, false는 true로 변경

비교연산자 : >, <, >=, <=, ==, !=, ===, !==

자바와 차이점

- JS는 / 연산결과 나누어 떨어지지 않으면 실수 결과가 나옴!
- 같다, 다르다를 ===, !== 사용을 권장!
- == : 데이터가 같은지 비교하지만 자료형이 같은지는 비교하지 않음
- === : 데이터와 자료형이 모두 같은지 비교

JS 조건문과 반복문

조건문(if, switch case break)는 자바와 동일!

반복문(while, for)도 자바와 동일!(단, 반복 시작 변수만 let으로 선언 할 것!)

for… of 문 (배열에서 사용)

```
const arr = [1, 3, 5];
for(const e of arr){
  console.log(e);
}
```

JS의 for-each문 (배열에서 사용)

```
const arr = [1, 3, 5];
arr.forEach(function(e, index){
  console.log(`index = ${index}, e = ${e}`);
});
```

객체를 순회할 때는 for…in 문법을 사용한다.

JS 함수

함수 선언 문법

```
function 함수명(매개변수들){
    실행내용
}

function 함수명(매개변수들){
    실행내용
    return 데이터;
}
```

```
function getSum(a, b){
    console.log('함수 실행~');
    return a + b;
}

let data = getSum(10, 20);
```

JS 함수표현식

함수 표현식 문법

```
const 함수명 = function (매개변수들){
    실행내용
}

const 함수명 = function (매개변수들){
    실행내용
    return 데이터;
}
```

```
const getSum = function (a, b){
    console.log('함수 실행~');
    return a + b;
}

let data = getSum(10, 20);
```

JS 화살표 함수

화살표 함수 문법

```
const 함수명 = (매개변수들) => {
    실행내용
    [return 데이터;]
}
```

```
const getSum = (a, b) => {
    console.log('함수 실행~');
    return a + b;
}

let data = getSum(10, 20);
```

매개변수가 하나라면 소괄호 생략 가능!

함수 안의 실행문이 하나라면 중괄호 생략 가능!

함수 안의 실행 내용이 하나이면서 return문 이라면 중괄호 생략과 동시에 return 키워드도 생략해야 함!

JS 배열

JS는 다수의 데이터를 저장하기 위해 배열과 객체를 사용한다.

배열은 여러 데이터에 순번을 부여하여 관리한다(순번이 존재 함, 중복 데이터 저장 가능)

배열 생성

```
const arr = new Array(); //빈 배열 생성
const arr = []; //빈 배열 생성
cosnt arr = [1, 2, 3]; //1,2,3 세 데이터를 갖는 배열 생성
const arr = [1, 'script', [1,2,3]]; //자료형이 달라도 저장 가능
```

배열의 각 요소의 값을 변경하거나 읽는 문법은 자바와 동일!

자바와 다르게 배열의 크기를 변경할 수 있으며, 자료형이 다른 데이터도 저장 가능!

JS 객체 - 1

JS는 다수의 데이터를 저장하기 위해 배열과 객체를 사용한다.

객체는 여러 데이터를 key와 value의 쌍으로 관리한다.(순번 존재 x, key값 중복 불가)

객체 생성

```
const obj = new Object(); //빈 객체 생성
const obj = {}; //빈 개체 생성
```

```
const person = {
  name : 'kim',
  age : 20,
  'my hobby' : '울산시'
};
```

```
console.log(person.name); // 'kim'
console.log(person['name']) // 'kim'
```

JS 객체 - 2

객체 값 읽기

```
const person = {
  name : ‘kim’,
  age : 20,
  addr : ‘울산시’
};
```

```
//값 읽기
console.log(person.name); //  'kim’
console.log(pserson[‘name’]); // 'kim’

//값 추가 및 수정
person.hobby = ‘study’;
person[‘gender’] = ‘female’;

//값 삭제
delete person.name;
delete person[‘name’];
```

JS Truthy & Falsy

JS는 참, 저깃이 아니더라도 참, 거젓으로 참을 평가한다.

```
if(123)
  console.log('true');
else
  console.log('false');


if (undefined)
  console.log('true');
else
  console.log('false');
```

Falsy한 데이터

```
let data_1 = undefined;
let data_2 = null;
let data_3 = 0;
let data_4 = '';
let data_5 = NaN;
```

Truthy한 데이터

0이 아닌 숫자 데이터

빈 문자(열)가 아닌 문자(열)

JS 구조분해할당-1

배열, 객체를 분해해서 일부분의 데이터를 다른 변수에 할당하는 문법

기존 방식

```
let arr = [1, 2, 3];
let a = arr[0];
let b = arr[1];
let a = arr[2];
```

구조분해할당 방식

```
let arr = [1, 2, 3];
let [a, b, c] = arr; //a=1, b=2, c=3
```

```
let arr = [1, 2, 3];
let [a, b] = arr; //a=1, b=2
```

```
let arr = [1, 2, 3];
let [a, b, c, d] = arr; //a=1, b=2, c=3, d=undefined
```

JS 구조분해할당-2

기존 방식

```
let person = {
  name : 'kim',
  age : 20,
  hobby : '공부'
};
```

```
let name = person.name;
let age = person.age;
let hobby = person.hobby;
```

구조분해할당 방식

```
let {name, age, hobby} = person;
```

* 변수명은 반드시 객체의 속성명과 동일해야 함!

변수명 변경 방식

```
let {name : myName, hobby} = person;
```

JS 구조분해할당-3

구조분해할당은 함수의 매개변수 전달에 자주 사용한다.

```
const person = {
  name : 'kim',
  age : 20
};
```

기존 방식

```
function test(p){
  let name = p.name;
  let age = p.age;
  console.log(name, age);
}

test(person);
```

구조분해할당 방식

```
function test({name, age}){
  console.log(name, age);
}

test(person);
```

JS Spread 연산자

배열, 객체에 저장된 여러 데이터를 개별로 흩뿌려주는 문법

```
let arr1 = [1, 2, 3];
let arr2 = [4, 5, 6];
```

```
let arr1 = [1, 2, 3];
let arr2 = [1, ...arr1, 5, 6];
```

위 두 배열을 이용해서

[4, 1, 2, 3, 5, 6]의 데이터를 얻으려면?

JS 배열에서 많이 사용하는 명령어-1

push() : 배열 마지막 요소에 데이터를 추가하고 배열의 길이를 리턴

```
let arr = [1, 2, 3];
arr.push(5);
let arr_length = arr.push(11, 12, 13);
console.log(arr_length); //7
```

unshift() : 배열의 첫 요소로 데이터 추가하고 배열의 길이를 리턴

```
let arr = [1, 2, 3];
let arr_length = arr.unshift(0);
console.log(arr, arr_length); //[0, 1, 2, 3] 4
```

JS 배열에서 많이 사용하는 명령어-2

shift() : 배열의 첫 요소를 제거하고 리턴

```
let arr = [1, 2, 3];
let removed_data = arr.shift();
console.log(arr); //[2, 3]
console.log(removed_data); //1
```

unshift() : 배열의 첫 요소로 데이터 추가

```
let arr = [1, 2, 3];
arr.unshift(0);
console.log(arr); //[0, 1, 2, 3]
```

slice() : 배열의 특정 범위를 짤라 새 배열로 리턴

```
let arr = [1, 2, 3, 4, 5];
let arr1 = arr.slice(1, 4); //첫번째부터 4번째 전까지!!
console.log(arr); //[1, 2, 3, 4, 5]
console.log(arr1); //[2, 3, 4]
let arr2 = arr.slice(2); //[3, 4, 5]
```

concat() : 배열에 다른 배열을 추가

```
let arr1 = [1, 2, 3];      let arr2 = [4, 5, 6];
let arr3 = arr1.concat(arr2);
console.log(arr3); //[1, 2, 3, 4, 5, 6]
```

JS 배열에서 많이 사용하는 명령어-3

아래 명령어는 학습 초반 아주 어렵지만 매우 강력한 기능을 제공합니다.(java에도 비슷한 내용이 있음)

find(callback) : 배열에서 조건을 만족하는 첫번째 데이터를 리턴

filter(callback) : 배열에서 조건을 만족하는 모든 요소로 모아 새로운 배열로 리턴

map(callback) : 배열 데이터를 새로운 형태의 데이터로 가공해 리턴

reduce(callback) : 배열 데이터의 특정 조건에 맞는 하나의 값으로 리턴