**변수 :** 선언, 초기화, 재할당

* Const : 변하지않는 데이터 값

Const 변수1 = ‘코딩’;

* Let : 변하는 데이터 값

Let 변수2 = ‘자바스크립트’;

**데이터 타입**

* String : 문자열
* Number : 숫자
* bigInt : 큰값의 숫자
* Boolean : true, false
* Symbol : 중복되지 않는 고유값
* Null / undefined : 수동적/ 자동적 빈 값
* Object : 객체 ( 함수, 배열, 객체)

**연산자**

* 산술 연산자
* 할당 연산자
* 문자열 연산자 : +

Let data = ‘happy’ + ‘2022!’

* 비교 연산자
* 논리 연산자 : &&, ||, !

**배열**

* Const travel\_spot = [‘방콕’, ‘뉴욕’, ‘파리’];
* Const travel\_spot = new Array(‘방콕’, ‘뉴욕’, ‘파리’);
* Const travel\_spot = new Array();

Travel\_spot[0]=’방콕’;

* Const 배열일 때 완전히 새로운 배열의 재할당은 불가능하지만 추가 삭제는 가능하다
* 추가
  + 맨 뒤에 추가 : Travel\_spot.push(‘토론토’);
  + 맨 앞에 추가 : Travel\_spot.unshift(‘토론토’);
  + Splice : 원하는 지점에 추가 및 삭제

Splice (시작 인덱스 , 삭제하고 싶은 원소의 개수, 추가하고 싶은 원소들)

Travel\_spot(4, 0, ‘멜버른’); // 4번인덱스에 추가됨

* 삭제 : Travel\_spot.pop(‘토론토’);
  + 뒤에 원소 삭제 : travel\_spot.pop();
  + 맨 앞의 원소 삭제 : travel\_spot.shift();
  + Splice : 원하는 지점에 추가 및 삭제

Splice (시작 인덱스 , 삭제하고 싶은 원소의 개수, 추가하고 싶은 원소들)

Travel\_spot(3, 3); // 3번 인덱스부터 세개 삭제됨

**객체**

* Key : value 형태로 저장
* 직접 객체 생성

Const pooh = {

Name : ‘pooh’,

Species : ‘bear’,

Job : ‘disney character’,

Gender : ‘boy’

sayHi : function() {

console.log(“Hi ~”);

};

}

* + Value를 함수로 줄 수 있음 -> method라고 부름( 다른 것들은 property)
* 객체 접근

Pooh.species

Pooh[‘species’]

* + Key 값에 특수 문자가 존재할 경우에는 무조건 [‘’]를 이용하여 객체에 접근 해야함
  + Value가 함수 일때, Poop[‘sayHi’]() 이렇게 접근하여 함수 사용
* 객체에 추가 및 삭제
  + Method 추가

Pooh[‘say-Bye’] = function() {

console.log(“Bye ~”);

};

* + Property 추가

Pooh.favorites = [‘honey’, ‘friends’, ‘cake’];

* + Property 삭제

Delete pooh.favorites;

* 생성자 함수 : 틀

Function Character(name, species, job, gender) {

This.name = name;

This.species = species;

This.job = job;

This.gender = gender;

This[‘sayHi’] = function () {

console.log(“I’m ${this.name}, Hi ~”);

};

This[‘say-Bye’] = function() {

console.log(“Bye ~”);

};

}

* 생성자를 사용하여 객체 생성

Const pooh = new Character(‘pooh’, ‘bear’, ‘ Disney character’, boy’);

**함수**

* Arrow function

Const mutiply10 = (num) => { // 매개변수가 한 개이면 괄호 생략 가능

Const result = num \* 10 ;

Return result;

} //코드가 한줄이면 괄호도 생략가능

* + 일반 함수와 달리 function키워드 쓰지 않고 const로 정의됨
  + 여러가지를 생략해서 함수를 간단하게 쓸 수 있음

Const multiply10 = num => num\*10;

**조건문**

* If문

If (조건) {

실행할 코드

}

* + false : 0, -0, null, false, NaN, undefined
* 삼항연산자

조건 ? 참일때 실행될 코드 : 거짓일때 실행될 코드;

* Swith문

DOM

Const body = Document.getElementsByTagName(‘body’)

HTMLcollection : 유사배열 (키값이 0, 1, 2…처럼 인덱스임) //유사배열이기 때문에 인덱스값을 활용해서 값을 가져올수있음

Console.log(body[0].innerText); // 바디안에 텍스트만 모두 보여줌

innerHTML은 태그포함해서 모두 보여줌

Const $container = Document.getElementsByClassName(‘container’)

Class 명으로 가져오기

Const $title = Document.getElementById(‘title)

Id 명으로 가져오기 < - 얘는 결과값이 유사배열이 아니므로 하나만 리턴됨

Const $cityList = Document.getElementById(‘ciryList’)

Const $cities = $cityList. getElementsByTagName (‘li’) // 변수를 사용해서 가져오기도 가능 이러면 li태그의 리스트들이 유사배열 형태로 오게됨

Const $cities = $cityList. querySelector(‘li’) // 이 조건에 만족하는 첫번째 원소만 가져와서 넣어줌

Const $cities = $cityList. querySelectorAll(‘li’) // Nodelist 형태로 얘는 다 가져옴 암튼 유사배열임

querySelector로 class이름 가져올 때 (.클래스네임) 이렇게 해주면됨

$cities.forEach(city => console.log(city.innerText); //유사배열이기 때문에 map 메소드 사용못해서 forEach를 사용해서 각 리스트의 원소를 하나씩 가져올 수 있음! forEach는 노드리스트를 사용 가능하지만 HTMLcollection은 사용 불가함 그래서 쿼리셀렉션을 더 마니 사용한다~

-유사배열을 배열로 변경하는 법 -> map메소드 사용가능한

1. spread operator

Const newCities = […$cities]

2. Array.from

Const newCities = Array.from($cities)

-js에서 css 변경하기 ( 권유되진않음)

$body[0].style.backgroundColor = ‘teal’

3. element 추가

Const worstCity = Document.createElement(‘ul’);

Const worstCitiyArry = [‘cairo’, ‘london’, ‘risbon’,’sydney’];

worstCityArr.map(city => {

const worstCityItem = document.createElement(‘li’);

worstCityItem.innerText = city;

worstCity.appendChild(worstCityItem); // worstCityItem을 하나하나씩 ul 안에 자식으로 넣어줌

})

$cityDiv.insertAdjacentHTML //이 메소드를 사용하면 맨뒤에가 아닌 여러가지 위치에 넣어줄수있음

Classlist를 통해서 그머냐 클래스 추가 가능

setAtrribute를 통해서 속성 추가 가능