<공부방법>

1. 결과물을 보고 일단 따라 만들어보려고 시도해본다.
2. 막히는 부분이 생기면 완성된 코드를 본다.
3. 코드를 봐도 알 수 없는 부분들은 강의를 다시 본다.
4. 강의를 봐도 알 수 없는 부분들은 구글에 검색해서 찾는다.
5. 알게 된 것들을 여기 메모해 둔다(그 상황만을 해결하는 방법보다는 일반화된 해결 방법론을 메모하는 것이 좋다). 잘못 메모해둔 부분은 그때그때 고친다.
6. 결과물을 다 따라 만들었다면, 막히는 부분이 없을 때까지 1~5를 반복한다(이때 메모해둔 것들을 참고해도 된다).
7. 나중에 공부가 부족하거나 빠진 부분이 있다고 판단되면, 결과물의 일부분만 따라해본 것은 아닌지 체크해본다. 만약 그렇다면 결과물에 다른 부분의 코드를 쭉 읽어보면서 모르는 부분을 체크하고 찾아본다.
8. 그래도 공부가 부족하면 모든 부분을 따라 만든다(1~6).

<코드를 깔끔하게 정리하는 법>

* 자바스크립트 파일은 html이나 css에 비해 양이 방대하고 복잡한 경우가 많으므로 한 화면에 있다 하더라도 기능 단위로 쪼개서 문서를 만들어 최대한 가독성을 높이자.
* 반복되어 쓰이는 문자열이나 숫자 등은 전부 const 변수들로 선언해 두자. 수정해야 할 때 여러 번 수정해야 하는 번거로움을 없앨 수 있다. Const 변수는 주로 대문자로 표기하여 선언한다.

Ex) const BASE\_COLOR

* 함수화 -> 특정 함수 안에 들어갈 내용이 조금이라도 길어지거나 복잡해진다 싶으면 따로 함수로 빼내서 호출하는 방식을 사용하자!! 가독성을 크게 높인다. 무조건 해야 함!!
* Init() 함수 -> 자바스크립트를 문서를 초기화할 때 사용하는 함수. 거의 모든 자바스크립트 문서에 다 들어간다. 보통 다른 함수들이 init()함수 내부에서 호출되는 방식으로 사용된다. 보통 문서의 개요를 보면 --- 1. 전역변수선언 2. 함수선언 3. init함수 선언(여기서 다른 함수들 호출) 4. Init() 호출 --- 이런식으로 이루어지게 된다.

Ex)

const title = document.getElementById("title");

const BASE\_COLOR = "rgb(255, 0, 0)";

const OTHER\_COLOR = "rgb(0, 0, 255)";

function handleClick() {

const currentColor = title.style.color;

if (currentColor === BASE\_COLOR) {

title.style.color = OTHER\_COLOR;

}

else {

title.style.color = BASE\_COLOR;

}

}

function init() {/\*\* \*/

title.style.color = BASE\_COLOR;

title.addEventListener("click", handleClick);}

init();

* 객체 선언 시 key를 부여할 때, key 이름에 객체 외부에 있던 변수 이름을 그대로 사용하고 싶을 땐 다음과 같이 줄여 쓸 수 있다.

문법 :

Const 객체이름={

key이름 : 외부변수이름 => key이름

}

Ex)

Const latitude=100;

Const userLocation={

Latitude : latitude => latitude 이렇게 줄여 쓸 수 있다!

((객체 key)lat : (위에 선언된 const)lat)

}

<자바스크립트의 장점>

JavaScript는 웹사이트로 Request를 보내고 응답을 통해서 데이터를 얻을 수 있는데

가져온 데이터를 Refresh 없이도 나의 웹사이트에 적용시킬 수 있다.

→ ex) 이메일을 확인할 때, 웹사이트를 새로고침하지 않아도

실시간으로 메일이 오는 것을 확인할 수 있다.

왜냐하면, JavaScript가 보이지 않는 곳에서 계속 데이터를 가져오고 있기 때문이다.

<html에 js파일 추가하는 법>

body태그 밑에

<script src="추가할 js파일의 주소와 이름"></script>

을 집어넣으면 됨.

Ex)

<body>

(…)

<script src="index.js"></script>

</body>

<객체 문법>

Const(let) 객체이름={

객체 요소 이름1 : 객체 요소 값1,

객체 요소 이름2 : 객체 요소 값2

}

<console.log 출력 원리>

* 여러 문자열이나 숫자 , 변수 등을 이어붙일 수 있는데 <,>로 붙일 경우 space가 한 칸 추가되며, <+>로 붙일 경우 space는 추가되지 않는다. (이 방식이 불편하기 때문에 <``>)(백틱)을 사용하게 된다)

<디자인 팁>

flatuicolors.com -> 깔끔한 색 배치를 눈으로 보고 복붙할 수 있는 사이트.

<백틱(<``>)>

문자열 사용 시 <’’>나 <””>대신 <``>을 일반적으로 사용한다. 변수가 들어갈 땐 ${변수} 이런식으로 넣는다.

Ex) console.log(`hello ${name}! you are ${age} years old.`);

* hello ghun! you are 23 years old.

<비교 문법>

===,!==,&&,|| 사용.

<객체와 함수>

객체 안에 함수가 들어갈 수도 있는데, 이 경우 문법이 조금 다르다.

문법 :

* 선언

Const 객체이름={

함수이름 : function(변수들){

함수내용~~~

}

}

* 사용

객체이름.함수이름(변수들)

Ex)

const calculator = {

    plus: function (x, y) {

*return* x + y;

    }

}

(…)

console.log(calculator.plus(2, 3)); //-> 5

<forEach>

배열에 대해 반복문 실행. 이때 배열의 원소가 순서대로 하나씩 forEach 속 함수의 인자에 전달될 수 있음.

문법 : 배열이름.

배열이름.forEach(~~~~~);

forEach 안에 함수가 들어간다면 =>

배열이름.forEach(function (원소를 전달받게 될 인자 이름(마음대로)) {

~~~~~~~~~~(반복할 내용)

});

<DOM>

DOM은 Document Object Model의 약자. DOM은 넓은 의미로 웹 브라우저가 HTML 페이지를 인식하는 방식을 의미한다. 조금 좁은 의미로 본다면 document 객체와 관련된 객체의 집합을 의미할 수도 있다. DOM은 tree 형식의 자료구조를 가지고 있다.

* Document 객체 -> 자바스크립트와 연결된 html 문서를 객체화 시킨 것. 추가적으로 getElementById, querySelector등의 dom function들을 이용해 특정 태그나 id, class명을 가진 세부요소들만 따로 지정할 수 있다. 이외에도 몇몇 기능을 제공한다.

Ex) const title\_from\_html = document.getElementById("title");

* title이라고 하는 id명을 가진 세부요소를 자바스크립트와 연결된 html문서에서 가져와서 title\_from\_html 변수 안에 객체화하여 담는다.

<getElementById>(id이름으로 css처럼 태그 지정하는 법)

문법 : document.getElementById("id이름") -> html에서 id이름을 가진 태그를 찾아서 객체화한다.

Ex) const title\_from\_html = document.getElementById("title");

* title이라고 하는 id명을 가진 태그를 자바스크립트와 연결된 html문서에서 가져와서 title\_from\_html 변수 안에 객체화하여 담는다.

<querySelector>

특정 id명, class명 등 조건을 만족하는 자식 노드들 중 제일 먼저 나온 노드를 지정하는 함수.

문법 : document.querySelector("조건")

* 조건 : id이름일 경우 #id이름, class이름일 경우 .class이름 등등임.

Ex) const title\_from\_html = document.querySelector("#title");

<createElement()>(자바스크립트에서 html 객체(element)를 생성하는 법)(생성만 하고 삽입은 아직 하지 않는다.)

문법 : const 객체이름= document.createElement("html 태그이름");

Ex) const listItem = document.createElement("li"); (여기서 listItem은 자바스크립트 안에서만 쓰는 변수이름임. 이름이 뭐든 상관없음.)

<appendChild>(객체(부모 element)에 자식 추가하는 법)

문법 : 객체(부모 element)이름.appendChild(추가할 자식 객체이름);

Ex) listItem.appendChild(span);

* html에서는 다음과 같다.

<li><span></span></li>

<push()>(array에 원소 추가하는 법)

배열이름.push(추가할 원소 이름);

Ex) myClassmates.push(cheolSu);

<removeChild>(객체(부모 element)에 자식 삭제하는 법)

문법 : 객체(부모 element)이름.removeChild(삭제할 자식 객체이름);

Ex) studentList.removeChil(cheolSu);

* html에서는 다음과 같다.

<ul>

<li>cheolSu</li>

<li>youngHee</li>

<li>youngSu</li>

</ul>

원래 이랬던 게

<ul>

<li>youngHee</li>

<li>youngSu</li>

</ul>

이렇게 됨.

<dir>(처음 보는 객체 공부하는 법)

객체 탐색/공부를 위한 함수.

문법 : console.dir(객체 이름);

이렇게 입력해서 console창을 보면 객체가 콘솔창에 출력되어있는데, 이를 클릭해보면 하위 요소들이 쭈루룩~~나열되어있다. 이걸 보고 공부하면 된다. 수정할 데이터의 태그의 이름이 무엇인지 찾는다던가 하는 식으로…

Ex) html 속 id=”text”인 태그의 텍스트 내용을 바꾸고 싶다면? (텍스트 내용을 original에서 new로)

1. 태그 지정 - const text = document.getElementById("text");
2. Text 객체 탐색 - console.dir(text); -> 콘솔창에서 innerHTML : “original” 임을 찾아냄.
3. 텍스트 변경 : text.innerHTML = "new";

<innerHTML>(텍스트 바꾸는 법)

태그 내부의 텍스트 내용을 바꾸어주는 함수.

Inspect를 해 보면, 콘솔 창에 있는 html 내부의 text내용이 바뀌어있게 됨.

문법 : 객체이름.innerHTML = “바꿀 텍스트 내용”;

Ex) title.innerHTML = "Hi!!";

※innerHTML과 innerText의 차이 - “바꿀 텍스트 내용”에 해당하는 문자열 부분을 innerText는 문자열로, innerHTML은 태그로 인식.

Ex) title.innerHTML = "<span>hi!</span>"; => hi!

title.innerText = "<span>hi!</span>"; => <span>hi!</span>

<style.color>(색깔 바꾸는 법)

태그 요소의 색을 바꾸는 함수.

Inspect를 해 보면, 콘솔 창에 있는 html 내부의 태그에 style 속성이 추가되어 있다.

문법 : 객체이름.style.color=”바꿀 색깔 이름”;

Ex) title.style.color = "red";

<title>(문서 상단에 있는 제목 바꾸는 법)

문서의 제목을 바꾸는 함수.

문법 : document.title=”바꿀 제목이름”;

Ex) document.title = "i own you,html";

<event>(이벤트 개념설명)(반응형 웹/앱 만들기)

자바스크립트는 이벤트에 반응하기 위해 만들어졌다. 웹/앱 등의 서비스에는 다양한 종류의 이벤트들(click,resize,submit,…)이 있고, 우리는 중간에 이러한 이벤트들을 가로챌(intercept)수 있다. 특정 객체마다 가로챌 수 있는 이벤트의 종류들은 정해져 있다(예를 들어, input객체에서 resize(크기조절)이벤트를 가로챌 수는 없다).

‘javascript event mdn’이라고 구글링하면 모든 event들을 찾아볼 수 있다.

문법 : 객체이름. addEventListener(“이벤트 이름”, 이벤트 발생 시 실행할 함수 이름);

★이때 함수이름()이라고 하면 안된다. 함수이름()이라고 하면 이벤트 발생 여부에 상관없이 함수가 무조건 실행된다.

Ex)

function handleResize(event) {

/\*event 인자는 이벤트 함수 호출 시 자동으로 자바스크립트가 함수 안에 넣어줌. 우린 그걸 선언해서 쓰기만 하면 됨. \*/

console.log("screen size has been resized.");

console.dir(event);

}

window.addEventListener("resize", handleResize); ->(handleResize()가 아님)

<event.preventDefault();>(이벤트의 기본 동작을 없애는 법)

어떤 이벤트들은 이벤트가 발생했을 때 다른 동작들이 자동으로 연이어 발생하게 하는 트리거가 된다(예를 들어, submit 이벤트 발생 시 웹사이트가 자동으로 새로고침된다). 이때 다른 동작들이 발생하지 않게 하고 싶다면 이 메소드를 사용한다.

특정 이벤트가 발생했을 때 호출되는 함수의 인자 부분에 event 인자를 추가한 후, event.preventDefault();라고 써주면 된다.

Ex)

function handleSubmit(event) {

event.preventDefault();

(…)

}

(…)

form.addEventListener("submit", handleSubmit); -> 호출

<색깔 저장하는 법>

“Rgb(0~255, 0~255, 0~255)”로 저장. 문자열이어야 함에 주의. 2번째와 3번째 숫자 앞에 스페이스도 꼭 한칸 있어야 함(…).

Ex) const BASE\_COLOR = "rgb(255, 0, 0)”

<classList>( 클래스명을 추가 / 삭제 / 검사하는 법)

자바스크립트와 css 사이의 역할을 완전히 분리하는 방향이 보다 권장된다. 자바스크립트에서는 로직만을 처리하고, css에서는 디자인만을 처리하고 싶은 것. 그러려면 자바스크립트를 통해 html을 조작하여 클래스명을 추가/삭제/검사할 수 있어야 한다.

문법 :

1. 추가

객체이름(element).classList.add(“추가할 클래스 이름”);

Ex) text.classList.add(TARGET\_CLASS); (TARGET\_CLASS=”added-class”)

1. 삭제

객체이름(element).classList.remove(“삭제할 클래스 이름”);

Ex) text.classList.remove(TARGET\_CLASS);

1. 검사

객체이름(element).classList.contains(“검사할 클래스 이름”)

* Element에 클래스이름이 있을 경우 1, 없을 경우 0을 반환

Ex)★(toggle 참고) function handleClick() {

const hasClass = title.classList.contains(TARGET\_CLASS);

if (hasClass) {

title.classList.remove(TARGET\_CLASS);

}

else {

title.classList.add(TARGET\_CLASS);

}

}

1. toggle

위 ★의 handleClick과 동일한 결과를 내는 함수임. 클래스명이 없다면 추가하고, 있으면 삭제해줌.

객체이름(element).classList.toggle(“토글할 클래스 이름”);

Ex) text.classList.toggle(TARGET\_CLASS);

<local storage>(소규모 데이터를 유저 컴퓨터에 저장하는 법)

문법 :

* 저장 => localStorage.setItem(“키 이름”,데이터);
* 사용 => localStorage.getItem(“키 이름”);

위 메소드를 이용해 local storage에 저장을 하면, 웹사이트를 inspect해서 application->local storage에 들어가보면, key-value에 값이 저장되어 있는 걸 확인할 수 있다. 웹사이트를 새로고침해도 여기 있는 데이터는 없어지지 않는다.

<input에 입력된 데이터을 자바스크립트로 가져오는 법>

문법 : const 저장받을 변수 = input객체(element)이름.value;

Ex) const currentValue = input.value;

<현재 시간 출력하는 법>

setInterval(), Date() 객체 이용.

<Date 객체>

자바스크립트에 내장되어 있는 객체인 것 같다. 현재 시각을 알아야 할 때 사용한다.

문법 :

* 선언 : const 현재시각변수 = new Date();
* 시각 받아오기 : 현재시각의 일부분(ex.시,분,초)변수 = 현재시각변수.get(ex.Hours, Minutes, Seconds)(); => 숫자 형태로 반환됨.

Ex)

const date = new Date();

const minutes = date.getMinutes();

const hours = date.getHours();

const seconds = date.getSeconds();

이것 말고도 getFullYear, getMonth, getDate, getDay, getHours, getMinutes, getSeconds, getMilliseconds가 있다.

<setInterval>(작업을 일정 시간 간격으로 반복해서 수행하고 싶을 때)

자바스크립트 내장함수인 것 같다.

문법 : setInterval(실행할 함수 이름, 밀리세컨드);

Ex) setInterval(printHello, 1000); ->1초에 한번씩 헬로 출력.

<난수 생성하는법>

Math.random() => 0과 1 사이의 복잡한 소수 난수 생성.

Math.floor => 소숫점 이하의 값을 떼버림. 가우스 함수의 역할과 동일.

ex) 2과 12 사이의 자연수 난수를 생성하고자 한다면?

const number = Math.floor(Math.random() \* 10)+2;

(0과 1 사이의 난수 → 0과 10 사이의 난수 → 0과 10 사이의 자연수 난수 → 2과 12 사이의 자연수 난수 순으로 변형됨.)

<이미지 삽입하는 법>(image객체 다루기)

1. 이미지 객체를 선언한다.

=> Const 이미지 객체 이름 = new Image();

1. 이미지가 있는 주소를 할당해 준다.

=> 이미지객체이름.src=”이미지 주소”;

1. 이미지의 수정을 css에서 해주고 싶다면 클래스명을 부여한다.

=> 이미지객체이름.classList.add(“부여할 클래스명”);

1. 이미지를 원하는 곳에 삽입한다.

=> 삽입할 곳.append(이미지객체이름);

<parentNode.append>

ParentNode.append() 메서드는 ParentNode의 마지막 자식 뒤에 Node 객체 또는 DOMString 객체를 삽입한다. (이와 비슷하게 prepend도 있다.)

자세한 것은 공식 문서 참고. (https://developer.mozilla.org/ko/docs/Web/API/ParentNode/append)

<api>

다른 서버로부터 쉽게 데이터를 가져올 수 있는 수단. 컴퓨터끼리 소통하기 위해 고안됨. Api key가 필요한 경우 로그인을 한다던지 해서 api 키를 복붙해서 변수에 저장하면 됨.

Ex) const API\_KEY = `629dfcedd7e5ae9c7ca5b325da4742b6`;

<fetch>(api로부터 데이터를 받는 방법)

문법 : fetch(“api 형식에 맞춘 문자열(사이트에서 찾아야 함.)”);

Ex) fetch(`https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lng}&appid=${API\_KEY}&units=metric`);

<.then()>

then=>온점(.)이전 작업이 완전히 끝난 후 괄호 안의 내용을 실행시킴. 이렇게 하지 않고 평소처럼 ;로 문장을 끝나고 바로 다음 문장을 썼을 경우, 이전 작업이 시간이 걸려 다 완료되지 않았는데도 다음 문장이 실행되어 오류가 발생할 수 있음.

<.json();>

스트림을 자바스크립트 객체로 바꾸는 역할을 함. (유튜브 풀스택에서 자세히 다룸)

스트림 객체는 대충 이런식으로 생겼음 : {"coord":{"lon":126.9536,"lat":37.3906},"weather": ……..

<날씨 정보를 받아오는 법>

이 부분의 정확한 문법이나 작동 원리는 유튜브 풀스택 코스에서 자세하게 다룸. 지금은 완전히 이해할 수 없으니 대략적인 설명만 이해하도록 하자.

function getWeather(lat, lng) {

fetch(`https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lng}&appid=${API\_KEY}&units=metric`)

.then(function (response) {

return response.json();

}).then(function (jsoned) {

//console.dir(jsoned);

const temperature = jsoned.main.temp;

const place = jsoned.name;

weather.innerText = `temperature of ${place} is ${temperature} degrees Celsius.`;

});

}

<JSON>(자바스크립트 객체를 string화, 혹은 문자열을 객체화하는 법)

JSON =>javascript object notation의 준말. 데이터가 자바스크립트 외부/내부로 오고갈 때 형식을 맞추기 위해 사용.

문법 :

* Object -> string : JSON.stringify(문자열로 바꿀 자바스크립트 객체)
* String -> object : JSON.parse(객체화시킬 문자열)

Ex)

(Object -> string)

const userLoaction = {

lat : ~~~~~~~~~~,

lng : ~~~~~~~~~~~~~

};

localStorage.setItem(“current\_location”, JSON.stringify(userLoaction));<= 객체를 문자열로.

(String -> object)

const loadedCoords = localStorage.getItem(“current\_location”); <=문자열 정보임

const parsedCoords = JSON.parse(loadedCoords);

<현재 위치(위도, 경도) 파악하는 법>

Navigator.geolocation 객체의 getCurrentPosition 함수를 이용하여 위도와 경도를 구한다.

문법 : navigator.geolocation.getCurrentPosition(파악 성공시 실행 함수, 실패시 실행 함수);

이때 성공 시 실행되는 함수한테 getCurrentPosition함수가 자동으로 위치 정보를 인자로 담을 수 있게 건네줌. 성공시 실행되는 함수는 그걸 인자에서 선언해서 사용하기만 하면 됨. 그 인자에 대한 정보는 chrome 콘솔에서 console.dir해서 찾아볼 수 있음.

Ex)

navigator.geolocation.getCurrentPosition(handleGeoSuccess, handleGeoError);

function handleGeoSuccess(position) {//건네준 정보를, 성공 함수에서 position으로서 선언.

const lat = position.coords.latitude;//dir로 찾아보면 나옴.

const lng = position.coords.longitude;

(…)

}

function handleGeoError() {

console.log("can't access geoLocation");

}

<event.target>

Event.target은 특정 이벤트를 일으킨 element를 의미한다. 주로 이벤트 발생 시 실행되는 함수에서 이벤트를 일으킨 element를 다시 제어하거나 그 위치를 참조하고 싶을 때 사용한다.

문법 : 이벤트이름.target

Ex)

function whenClicked(event) {

const btn = event.target; //이벤트가 일어난 element를 다시 변수에 담음.

const li = btn.parentNode;

(…)

});

toDos = cleanToDos;

saveToDos();

}

//(외부함수에서 이벤트 발생)

btn.addEventListener("click", whenClicked);

<element.parentNode>(특정 객체(element)의 부모 element로 접근하는 법)

Html 문서는 트리 구조로 되어 있다. 그래서 특정 객체에 대한 부모 원소로 접근할 수 있다.

문법 : 자식html객체.parentNode

Ex) const div = span.parentNode; => div변수 안에는 span을 담고 있던 html div객체가 담기게 됨.

<array.filter>

말 그대로 배열을 필터로 거른다. 필터 내부에는 함수가 있는데, 배열의 모든 원소에 함수가 실행되어 그 리턴값이 참인 경우만 필터에서 걸러져 나온다.

문법 :

const 필터링된 배열 이름 = 필터링할 배열 이름.filter(function (필터링할 배열 원소이름) {

return 참 또는 거짓이 되는 문장(이항 연산자 등);

});

Ex)

const filteredArray = Array.filter(function (element) {

return element.id !== 1;//filteredArray에는 id가 1인 것을 뺀 나머지 원소들이 들어감.

});

<parseInt>(문자열을 정수로 바꾸는 법)

문법 : parseInt(string);

Ex) const intData = parseInt(“123”); //intData=123;