**이미지 갤러리 프로젝트 브라우저**

**ep1. Json 파일을 브라우저 통신으로 받아오기.**

1. js 문서에 XMLHTTP를 request 받아와 요청하는 함수 적용.

var req = new XMLHttpRequest();

req.open("GET", "json/image\_list.json"); *// json 문서 파일 얻어오도록 요청*

req.onreadystatechange = function () {

*// 콜백함수 등록 == 두번 째 인자에 부합하는 조건 값이 대입되면 함수 호출*

*if* (*this*.readyState == 4) {

*// 데이터 통신이 정상적으로 수신 됐을 때*

console.log(*this*.response);

*// response 출력*

}

};

req.send();

json 파일 : 자바스크립트의 객체를 문자열로 반환

🡺 파싱하여 {키 : 값}으로 가져올 수 있음.

JSON.stringfy( ) : 해당 인자 값으로 들어오는 데이터들을 문자열로 반환

JSON.parse( ) : 해당 인자의 문자열 데이터들을 파싱하여 값으로 반환.

[

"/image/1.jpeg",

"/image/2.jpeg",

"/image/3.jpeg",

"/image/4.jpeg",

"/image/5.jpeg",

"/image/6.jpeg",

"/image/7.jpeg",

"/image/8.jpeg",

"/image/9.jpeg",

"/image/10.jpeg",

"/image/11.jpeg"

]

통신이 성공됐으면 해당 조건을 통해 html document를 제작하는 함수 작성.

var req = new XMLHttpRequest();

req.open("GET", "json/image\_list.json"); *// json 문서 파일 얻어오도록 요청*

req.onreadystatechange = function () {

*// 콜백함수 등록 == 두번 째 인자에 부합하는 조건 값이 대입되면 함수 호출*

*if* (*this*.readyState == 4) {

*// 데이터 통신이 정상적으로 수신 됐을 때*

*// console.log(this.response);*

*// response 출력*

var data = JSON.parse(*this*.response);

*// image\_list에 정제된 json 파일의 데이터를 파싱하여 값으로 반환하는 반복.*

*for* (var i = 0; i < data.length; i++) {

var div = document.createElement("div");

*// div 변수를 선언하여 document에 div 엘리먼트를 생성하는 함수 할당*

div.setAttribute("class", "image");

*// class = "image"를 포함하고 있는 div 태그 생성*

var img = document.createElement("img");

*// img 태그 생성*

img.src = data[i]; *// src에 datap[i] 인덱스 값을 소스로 전달.*

div.appendChild(img);

*// img content를 포함하는 div 태그 생성.*

document.body.appendChild(div);

*//body 태그 안에 div 태그 생성.*

}

}

};

req.send();



그렇게 적용된 코드는 브라우저에서 위와 같이 작동함

**ep2. js에서 class를 생성하고 제거할 수 있는 api와 함수 코드 작성.**

1. 반복문 스코프에 조건을 작성하여 class가 존재할 때 안 할때 작동하는 함수 적용

div.onclick = function () {

*if* (*this*.getAttribute("class").indexOf("image-selected") == -1) {

*this*.setAttribute("class", "image image-selected");

} *else* {

*this*.setAttribute("class", "image");

}

*//<div class ="image image-selected"> 생성*

};

그러나 위의 코드는 효율적이지 못 함.

2. 브라우저에서 제공하는 api classList 메서드를 통해 개선 가능.

div.onclick = function () {

*this*.classList.toggle("image-selected");

*//<div class ="image image-selected"> 생성*

};

위의 두 코드는 작동하는 방식은 같으나 메모리의 처리의 효율과 작동 처리 방식에서 차이가 있다.

**ep3. html 태그에 bind된 this를 찾아 인자 값을 전달하여 해당 value를 수정하는 js 작성.**

function selectAll(btn) {

var images = document.getElementsByClassName("image");

*// image를 포함한 div 태그를 검색*

*for* (var i = 0; i < images.length; i++) {

*if* (btn.value === "Unselect All") {

images[i].classList.remove("image-selected");

} *else* {

images[i].classList.add("image-selected");

}

}

*if* (btn.value === "Unselect All") {

btn.value = "Select All";

} *else* {

btn.value = "Unselect All";

}

*// btn.value = "Unselect All";*

}

개발자 콘솔에서 해당 반복에 디버그를 적용시켜보면 call stack으로 onclick을 참조 함.

해당 onclick의 this는 input태그를 바인딩하고 있다.

<body>

<input *type*="button" *value*="Select All" *onclick*="selectAll(*this*)" />

</body>

그렇게 인자 값으로 this를 전달하면 selectAll function의 인자로 this를 받게되고 btn은 input태그의 값(value)을 참조 할 수 있음.

**ep4. mouse api를 활용한 js 함수 기능.**

div.onmouseover = function () {

var element = *this*;

*this*.timerId = setTimeout(function () {

element.classList.add("image-magnified");

}, 1000);

};

div.onmouseout = function () {

clearTimeout(*this*.timerId);

*this*.classList.remove("image-magnified");

};

* mouseover event : 마우스 커서가 엘리먼트 위에 올라간 경우 발생하는 이벤트
* mouseout event : 마우스 커서가 엘리먼트에서 밖으로 나간 경우 발생

익명함수를 통해 계속해서 호출해서 작동하는 것이 아닌, 해당 이벤트가 발생할 때 마다 작동해야 하는 원리이기에 익명함수를 통해 일회성 코드를 작동.

풀이

onmouseover의 스코프에 존재하는 div태그에 setTimeout이 작동하게 되면 해당 element에 image-magnified class를 포함하게 하여 css적용.

onmouseout은 그 반대로 작동하도록 함수 적용.

**ep4. slide 기능 구현해보기.**

function slideShow(btn) {

var images = document.getElementsByClassName("image");

var index = 0;

images[index].classList.add("image-magnified");

var intervalId = setInterval(function () {

images[index].classList.remove("image-magnified");

index++;

*if* (index < images.length) {

images[index].classList.add("image-magnified");

} *else* {

clearInterval(intervalId);

}

}, 1000);

}