JS Study

제목 없는 데이터베이스

#1 Basic of JavaScript

Variable

종류 및 선언 방식
const, let, var
const 와 letd의 차이점, var 사용하면 안되는 이유 > **항상 const, 업데이트가 필요한 변수는 let, var은 사용x**
boolean(true, false), null, undefined
let a = 5; 와 a = 7 의 차이 > 왼쪽은 변수를 생성과 동시에 값을 초기화 & 오른쪽은 변수의 값을 업데이트

Arrays

선언 방식, 값을 가져오는 방법, 값을 추가하는 방법(push, unshift),

Objects

array 와 objects의 차이점
array에 item을 넣을 때는 item마다 의미를 넣기 힘들지만 objects는 item마다 의미부여 가능!!
console은 objects
const 는 update 불가
const로 objects 를 선언 후 const를 다른 type으로 변환 불가
const objects = { } 선언 후 objects = bollean (x) but!! constant(objects) 안의 속성을 update하는 것은 가능!!
objects에 속성을 추가 objects.properties = "value";

Function

function 은 어떤 코드를 캡슐화해서 실행을 여러번 가능하게 한다.
( ) 는 function을 실행하는 방법
argument는 function을 실행하는 동안 데이터를 보내는 방법 > ( ) 안에 argument 값을 넣어서 function에 데이터를 전송
argument의 순서 중요
예) 필요한 argument가 하나만 존재할 때 전달되는 argument가 여러개가 전달될 경우, 제일 첫번째 argument를 값으로 받고
그 뒤에오는 argument는 받지 않는다.
argument 명은 지역변수로 존재
argument는 function의 body { } 안에서만 사용 가능 body 외부에서 접근 불가

objects 안에 function 선언

```
const objects = {
  function명: function (argument) {
  내용
```

```
  }
}
```

Returns

console.log를 사용하지 않고, function이 계산의 결과를 나에게 제공

console.log나 alert를 사용하면 결과를 console창이나 페이지 경고창으로 보여주고 끝,

function을 사용하는 궁극적인 목적은 function의 결과를 가지고 운영하는 데 있다.

return을 사용하지 않고, console.log를 사용하면 function의 결과 값은 undefined

하지만 return을 사용하면 그 계산 결과를 계속해서 활용 가능

```
const calculator = {
  plus: function (a, b) {
  alert(a + b);
  }
};

console.log(calculator.plus(2,3))
실행 결과 > 실제 calculator.plus의 값은 undefined
```

```
const calculator = {
  plus: function () {
  alert("hi");
  }
};

console.log(calculator.plus)
실행 결과 > function 명시
```

!!return 사용 시 function은 종료

```
const calculator = {
  plus: function (a, b) {
  console.log("hello")
  return a + b
  console.log("bye bye")
```

```
  }
}
실행 시 console.log("hello") 는 실행되지만,  console.log("bye bye")는 실행안됨
```

Conditionals

```
const age = parseInt(promt("How are you?"));

console.log(isNaN(age));
```

▼ **사용 function**

promt ( ): 사용자에게 입력을 받는 function (아주 오래된 사용 방법)

css 적용 불가, 브라우저에서 기본으로 지원

type of : value의 type을 확인 하기 위한 function

parseInt( ) : String type의 value를 Number type으로 변환

isNaN( ) : 무엇인가 NaN인지 판별하는 function

```
if(condition) {
  // condition === true
} else {
  // condition === false
}
```

condition(조건)은 boolean type 이어야 한다.(true or false)

```
if (isNaN(age) || age < 0) {         //조건문 ||(OR) 은 둘 중 하나만 TRUE면 TRUE
  console.log("Write A Number");
} else if (age < 18) {
  console.log("you are too young.");
} else if (age >= 18 && age <= 50) {  //조건문 &&(AND) 는 둘다 TRUE 일 때 TRUE
  console.log("you can drink");
} else {
  console.log("you can't  drink");
}
```

else 는 선택사항

= & == & === !==

= : 하나의 value를 할당 하는 것

=== : equals 같은지 판단

#2 JavaScript On The Browser

JavaScript는 HTML에 접근하고 읽을 수 있게 설정되어있다.

브라우저가 HTML 정보가 아주 많이 들어 있는 document 라는 object를 전달해 준다.

Document

document 인터페이스는 브라우저가 불러온 웹 페이지를 나타내며, 페이지 컨텐츠(DOM트리)의 진입점 역할을 수행.

console.dir(document)

DOCUMENT

첫번째 document의 항목을 가져오기

getElementById

-ID를 사용하여 HTML의 Element 가져오기

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <h1 id="title">Grab Me!!</h1>
    <script src="0314.js"></script>
  </body>
</html>
```

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

console.log("title");
--------------------------------------------------------------------------------
실행 시 : <h1 id="title">Grab Me!!</h1> // h1태그를 보여줌
```

getElementsByClassName

class를 사용하여 Element 가져오기

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <h1 class="hellos">Grab Me!!</h1>
    <h1 class="hellos">Grab Me!!</h1>
    <h1 class="hellos">Grab Me!!</h1>
    <h1 class="hellos">Grab Me!!</h1>
    <script src="0314.js"></script>
  </body>
</html>
```

```
const hellos = document.getElementByClassName("hello");

console.log("hellos");
--------------------------------------------------------------------------------
실행 시 :
HTMLCollection(4) [h1.hellos, h1.hellos, h1.hellos, h1.hellos]
0: h1.hellos
1: h1.hellos
2: h1.hellos
3: h1.hellos
length: 4

array 형태로 각 h1의 elements를 보여줌
objects형태가 아님
```

getElementsByTagName

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <div>
      <h1>Grab Me!!</h1>
    </div>
    <div>
      <h1>Grab Me!!</h1>
    </div>
    <div>
      <h1>Grab Me!!</h1>
    </div>
    <script src="0314.js"></script>
  </body>
</html>
```

```
const title = document.getElementsByTagName("h1");

console.log("title");
--------------------------------------------------------------------------------
실행 시 :
HTMLCollection(4) [h1.hellos, h1.hellos, h1.hellos, h1.hellos]
0: h1.hellos
1: h1.hellos
```

```
2: h1.hellos
모든 h1 tag 를 array형태로 가져옴
objects 형태가 아님
```

querySelector(All)

element를 css selector로 가져올 수 있다.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <div class="hello">
      <h1>Grab Me!!</h1>
    </div>
    <script src="0314.js"></script>
  </body>
</html>
```

```
const hellos = document.querySelector(".hello h1");

console.log(hellos);
--------------------------------------------------------------------------------
실행 시 :
<h1>Grab Me!!</h1>

단 하나만의 element를 return함
.hello h1 이 여러개 일 경우
Returns the first element that is a descendant of node that matches selectors.
여러개를 불러오고 싶을 경우
querySelectorAll(".hello h1") 사용
CSS SELECTOR 를 사용하는 방식으로 가져옴
CLASS > .
ID    > #

getElementsByClassName() 에서 ( )안에는  . 을 안쓰는 이유
>> function 자체가 class를 가져오는 것을 명시했기 때문
```

getElementById(”hello”)

querySelector(”#hello”)

이 두개의 결과는 동일 but! getElementById 는 querySelector(”#hello h1”) 처럼 하위 요소를 선택 불가

Events

Event를 Listen하기 위해서는 HTML Element를 가져와서

addEventListener function을 실행시켜 준다.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <div class="hello">
      <h1>Grab Me!!</h1>
    </div>
    <script src="0314.js"></script>
  </body>
</html>
```

```
const hellos = document.querySelector(".hello h1");

function handleTitleClick() {
  console.log("title was clicked");
}

hellos.addEventListener("click", handleTitleClick);
--------------------------------------------------------------------------------
실행 시 :
Grab Me !! 라는 문구를 클릭 시 console창에 title was clicked

주의!! addEventListener에 사용된 매개변수 handleTitleClick 함수는 ()를 붙이지 않는다.
이유 event가 일어난 뒤 js function을 동작하게 하기 위해서

첫번째 매개변수는 어떤 event를 liste하고 싶은지 명시하고,
두번째 매개변수에는 event를 listen했을 때, 어떤 함수를 실행할지 정한다.
```

console.dir 을 사용하여 element를 확인!!

“on”이 붙은 objects는 event listener이다.

eventlistener 예제)

```
const hellos = document.querySelector(".hello h1");

function handleTitleClick() {
  hellos.style.color = "blue";
}

function handleMouseEnter() {
  hellos.innerText = "mouse is here!!";  //마우스가 위에 있을 때
}

function handleMouseLeave() {
  hellos.innerText = "mouse is gone!!";  //마우스가 떠났을 때
}

hellos.addEventListener("click", handleTitleClick);
//== hellos.onclick = handleTitleClick;
hellos.addEventListener("mouseenter", handleMouseEnter);
hellos.addEventListener("mouseleave", handleMouseLeave);

.addEventListener 를 선호하는 이유는
.removeEventListener 를 이용하여 eventListener를 지우기 때문
```

window

```
function handleWindowResize() {
  document.body.style.backgroundColor = "tomato";
}
function handleWindowCopy() {
  alert("copier!!");
}

window.addEventListener("resize", handleWindowResize);
window.addEventListener("copy", handleWindowCopy);
-------------------------------------------------------------------------------

document.body 중요!! document.div 불가!!
title이나 head같은 중요요소는 접근 가능
div나 h1같은 요소는 querySelector를 이용하여 접근
```

```
const h1 = document.querySelector(".hello h1");
```

```
function handleTitleClick() {
  const currentColor = h1.style.color;
  let newColor;
  if (currentColor === "blue") {
    //h1.style.color === "blue"
    newColor = "tomato";
  } else {
    newColor = "blue";
  }
  h1.style.color = newColor;
}
h1.addEventListener("click", handleTitleClick);
--------------------------------------------------------------------------------
```

CSS를 활용한 JavaScript

```
body {
  background-color: beige;
}

h1 {
  color: blue;

}

.active {
 color: tomato;
}
```

```
const h1 = document.querySelector(".hello h1");

function handleTitleClick() {
  if (h1.className === "active") {
    h1.className = "";
  } else {
    h1.className = "active";
  }
}

h1.addEventListener("click", handleTitleClick);
```

--

```
const h1 = document.querySelector(".hello h1");

function handleTitleClick() {
  const activeClass = "active";
  if (h1.className === activeClass) {
    h1.className = "";
  } else {
    h1.className = activeClass;
  }
}

h1.addEventListener("click", handleTitleClick);
```

—

classList사용

```
const h1 = document.querySelector(".hello h1");

function handleTitleClick() {
  const activeClass = "active";
  if (h1.classList.containss(activeClass)) {
    h1.className = "";
  } else {
    h1.className = activeClass;
  }
}

h1.addEventListener("click", handleTitleClick);
```

.remove 사용

```
const h1 = document.querySelector(".hello h1");

function handleTitleClick() {
  const activeClass = "active";
  if (h1.classList.containss(activeClass)) {
    h1.classList.remove(activeClass);
  } else {
    h1.classList.add(activeClass);
  }
}

h1.addEventListener("click", handleTitleClick);
```

toggle 사용

```
const h1 = document.querySelector(".hello h1");

function handleTitleClick() {
  h1.classList.toggle("active");
}
h1.addEventListener("click", handleTitleClick);

--------------------------------------------------------------------------

toggle은 h1의 classList에 active class가 이미 있는지 확인해서 있다면
제거해주고, 없다면 추가해준다.
```