Chapter 12. Javascript : 변수, 함수, 리턴, 배열, 객체, 인라인 모델, 쿼리셀렉터

- 변수의 선언 및 할당 : let , var, const 으로 데이터를 담는 공간을 새로 만들어 값을 부여
  - 변수(variable) : 데이터를 담는 공간
  - 선언(declaration) : 새로 만듬
  - 할당(assigment) : 변수의 값을 부여
  - let : 일반적 변수 선언, 재할당이 가능
  - const : 왜곡되면 안 될 소중한 데이터에 선언, 재할당 불가

- 함수(function) : 미리 약속된 계산식
  - 정의 : JavaScript의 함수는 작업을 수행하거나 값을 계산하는 명령문의 집합인 프로시저(procedure)와 비슷하지만, 프로시저가 함수로 쓰이려면 입력을 반드시 받아야 하고 입력과 명확한 관계가 있는 출력을 반환해야 합니다. 함수를 사용하려면 함수를 호출할 스코프 내에서 함수를 정의해야 합니다.
  - 형식 : function 함수명 (매개변수들) {실행할 내용}
    - 함수명 : 해당 함수의 선언
    - 매개변수들 : 함수가 외부로부터 입력값을 전달 받는 통로, 사용자로부터 입력받은 값을 임시로 저장해 두기 위한 도구
      - 괄호( )로 묶고 쉼표 , 로 구분
    - 함수가 실행할 기능을 중괄호로 묶음 { }

  - 기본 함수 및 설정

```
function hello(name) {
    console.log("hello"+name);
}

let a = hello("kmh")
```

```
hellokmh
```

    - function : 함수 선언
    - hello : 함수명
    - ( ) : 매개변수(parameter) // 함수명( ) 를 의미
    - { } : 중괄호안의 내용으로 해당 함수의 기능을 정의

    - 이제 function의 선언과 함수명, 매개변수, 중괄호로 인하여 hello( ) 의 ( ) 안에 값을 입력시 자동으로 “hello”+( ) 가 실행
    - 즉, 매개변수()의 값을 { } 의 조건으로 변경
      - hello(name) -> {console.log(“hello”+name)}
      - hello(“kmh”) -> hellokmh 출력
    - 함수 및 변수의 정의후, 함수의 호출을 해야 작동

```
function add(a,b){
    console.log(a+b);
}

let a = 1;
let b = 2;
add(a,b)
```

```
3
```

```
function add(a,b){
    console.log(a+b);
}
add(1,2)
```

```
3
```

- 리턴(return) : 지정된 값을 반환, 함수의 가장 마지막에 하는 행동이므로 함수가 종료됨

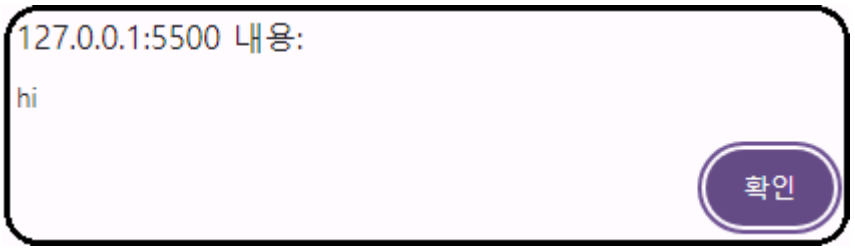
- 배열(array) : 데이터를 순서대로 늘어놓은 것, 대괄호[ ] 를 사용

- 객체(object) : JS의 객체는 데이터를 효과적으로 저장하기 위한 도구
    - 딕셔너리와 유사, 책꽂이와 유사하게 책을 담을 때 순서는 중요하지 않음
    - 변수의 선언과 함께 할당하는 값으로 중괄호를 작성
    - let a = { }

- JSON(JavaScript Object Notation) : 자바스크립트에서 사용하는 객체의 규격

- 이벤트 핸들러 : 사용자가 상호작용할 때 속성값을 수행하는 것

```
<input type="button" value="click" onclick="alert('hi')">
```



    - onclick = “alert(‘Hi’)”
    - onclikc가 속성이며, 그 뒤를 속성값이라고 함

- 인라인 이벤트 모델 : 태그에 onclick 속성을 부여하여 이벤트를 처리하는 방법

```
<body>
  <input type="button" value="click"
  onclick="for(let i=0, i<n; i++){console.log(i)}">
</body>
```

인라인 이벤트 모델

    - 해당 태그의 기능을 한눈에 확인되며 <script> 태그가 별도로 미필요, 쉬운 사용

```
<body>
  <input type="button" value="click" onclick="num(5)">
</body>
<script>
  function num(n){for(let i=0, i<n; i++){console.log(i)}}
</script>
```

*script*를 *JSON* 처리시 매우 간결

    - 짧은 코드, HTML 태그의 간소화 가능

- Query Selector() : HTML 요소를 특정하여 제어, 파라미터에 일치하는 첫 번째 Element를 반환

```html
<body>
    <h1>h1</h1>
    <h2 class="h2">h2</h2>
    <h3 id="h3">h3</h3>
    <h3>h3_2</h3>
    <div>
        <span>Span1</span>
        <span>Span2</span>
    </div>
    <script src="main.js"></script>
</body>
```

h1

h2

h3

h3_2

Span1 Span2

html / 브라우저 화면

```js
const a = document.querySelector("h1");
a.style.color = "red";
```

js

  - tag : document.querySelector("h1")
  - class : document.querySelector(".h2")
  - id : document.querySelector("#h3")
  - div , span : document.querySelector("div span")
    - 첫 번째 element만 반환, "div span:first-child"
    - 두 번째 element에 적용시, “nth-child” 혹은 “last-child” 를 사용[1]

```html
<body>
  <input type="button" value="click" onclick="red()">
</body>
<script>
  function red(){
      let button = document.querySelector("input")
      button.style.backgroundColor = "red"}
</script>
```

querySelector 로 처리시

전후

[1] [JS] 자바스크립트 querySelector 함수 사용법 및 예제

- getElementBy() 와querySelector() 의 차이점
    - 공통 : 배열과 같이 각 항목을 접근하기 위한 인덱스를 제공
    - getElementBy : HTMLCollection 객체를 반환
    - querySelector : NodeList 객체를 반환

종류	프로퍼티	메서드
HTMLCollection	HTMLCollection.length	HTMLCollection.item() HTMLCollection.namedItem()
NodeList	NodeList.length	NodeList.item() NodeList.entries() NodeList.forEach() NodeList.keys() NodeList.values()

```
<div>
  <h1 class="userInfo">홍길동</h1>
  <p class="userInfo">20세</p>
  <p class="userInfo">한국대학교</p>
</div>

<script>
  const info = document.getElementsByClassName("userInfo");
  console.log(info);
</script>
```

```
▶ HTMLCollection(3) [h1.userInfo, p.userInfo, p.userInfo]
```

*getElement~*종류들은 *HTML Collection*을 리턴

```
<body>
  <div>
    <h1 class="fruit">사과</h1>
    <p class="fruit">빨간색</p>
    <p class="fruit">1개 천원</p>
  </div>

  <script>
    const item = document.querySelector(".fruit");
    console.log(item);
  </script>
</body>
```

```
▶ h1.fruit
```

```
<body>
  <div>
    <h1 class="fruit">사과</h1>
    <p class="fruit">빨간색</p>
    <p class="fruit">1개 천원</p>
  </div>

  <script>
    const item = document.querySelectorAll(".fruit");
    console.log(item);
  </script>
</body>
```

```
▶ NodeList(3) [h1.fruit, p.fruit, p.fruit]
```

*querySelector*종류는 *Node List* 를 리턴, 첫째 엘리먼트를 반환 / *querySelectorAll* 은 전체를 반환

- reference
    - [getElementById를 쓸까, querySelector를 쓸까?]
    - [[JavaScript] getElementById와 querySelector 차이점]

- addEventListener
    - 요소.addEventListener(이벤트, 함수)

```
<body>
  <input type="button" value="click">
</body>
<script>
  let button = document.querySelector("input")
  button.addEventListener("click",
  function e(){alert('clicked')})
</script>
```

        - 함수의 {기능}에서 이벤트를 넣을 수 있다.

    - 요소.addEventListener(이벤트, b)
      function b(){기능}에서 관련된 기능을 넣을 수 있다.

```
<script>
    let p;
    function a(){
      p=document.getElementById("p");
      p.addEventListener("mouseover", b);
      p.addEventListener("mouseout", c);
    function b(){p.style.backgroundColor="red";}
    function c(){p.style.backgroundColor="white";}
    }
</script>
```

- 자바스크립트 코드를 별도의 파일로 분리

```
<script src="JSON.js"></script>
```

파일경로를 입력