221012 수업 -1

키워드	검증로직
날짜	@2022년 10월 12일 오전 9:09

JavaScript

- Web에서 해석 실행하는 언어
- 웹 브라우저에서 해석(해석기:엔진)
  - 라인 단위(순서) 실행&해석
- 객체 지향 언어 > class 개념 > 상속 , 다형성, 재 사용성

JavaScript 사용 이유

1. html 의 content , attribute 변경 , 삭제 , 추가 동적으로 가능
2. css 속성 변경 , 삭제 , 추가 동적으로 가능
3. 유효성 검증
   - id 입력, 주민 번호 맞는지 , 데이터 숫자 등을 검증
4. 동적인 웹 화면 구성 → 화면 구성
5. 전 세계적으로 1위 언어
   - 모든 개발 회사 , 개발자 관심 ....
   - javascript > 효율적으로 > 코드량 줄이고 > 많이 쓰는 재사용 > lib or FrameWork
   - 50줄의 코드 구현 .. 내가 만든 것을 당신이 ... 5줄로 할 수 있어
   - jquery.js (내부적인 코드 : javascript) >> 안에 있는 자원을 활용 > 생산성 극대
   - angular.js(MVC) , vue.js , react.js ..... 100여가지가 넘는 ....
   - 우리나라 개발 회사 (javascript : FrameWork)
   - 신규 : vue.js , react.js (Front) 단점: 버전변경 .... 사라질 수 도 있다
   - Front 최소 : vue.js 또는 react.js
   - javascript 언어 웹 브라우져 없이 로컬 서버에서 동작 (마치 java 처럼)
   - Node.js®는 Chrome V8 JavaScript 엔진으로 빌드된 JavaScript 런타임입니다.
   - back 단의 코드 javascript 대체 >> 서브 (채팅 서버, 보조 서버) >> npm 학습

6. 문법 (java 흡사 ...)
    - 대소문자를 엄격하게 구분
    - 종결자 ;
    - 타입(var , const , let) : let i="A"; let s = "aaa"; const NUM=100;
        - 유연한 언어 : var v; v=10 이때 내부적으로 (값을 받는 시점에) >> number

      ex) int i =10; double d = 3.14;
        - 연산자 , 제어문(java 동일) , 객체
    - 사용법 (css 사용법과 동일)
        - in-line (태그 안쪽에 : <p onclick="<script>...")
        - internal(page 안에서 일반적으로 <head><script> ...코드</head>)
        - external(외부 파일 :common.js >> link 방식) 선호 (유지 보수 , 재 사용)

---

in-line 방식

```
<input type="button" value="inline" onclick="alert('hello')">
```

internal 방식

```
<script>
  function call(){
      alert('internal');
    }
</script>
```

```
<input type="button" value="internal" >
```

external 방식

```
function excall(){   /* script/common.js */
  alert('external');
}
```

```
<script type="text/javascript" src="script/common.js"></script>
```

```
<input type="button" value="external" >
```

304코드 : 서버에서 수정된 페이지를 받음

```
console.log("디버깅, 결과 미리보기, 오류메시지 확인");
console.log(10+10);

document.write("<b>hello world</b>");
document.write('<table border="1">');
document.write('<tr><td>aa</td><td>bb</td></tr>B')
document.write('</table>');
```

- 문서가 실행되면 웹 프라우저 메모리에 body 안에 요소가 로딩
- 메모리에 DOM tree 형태로 필요에 따라 접근

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Insert title here</title>
        <script type="text/javascript">=
            function print(){
                let ele = document.getElementById("demo");
                console.log(ele);
                ele.innerHTML = "아 배고프다";
            }
        </script>
    </head>
    <body>
        <button onclick="print()">클릭</button>
        <p id="demo">demo라는 id는 p태그의 주소(식별자)</p>
    </body>
</html>
```

1. 변수 생성 > 값 입력 > 타입 설정(number, string)

```
var a; //정의x
let b; //정의 x
a=10;  //타입설정 number
b="A"  //타입설정 string
```

- ES5 이전 문법 (var) >> 2015 ES6(ECMA) >> let, const

var, let, const

- 권장 : let, const

초기화(처음 값을 갖는 것)

javascript(정수, 실수, boolean, 문자)

내부적으로 가지고 있는 타입 : Number, boolean, Array, Object 타입

global : 전역 변수

- let name = "kglim"; → script 태그 안에 생성한 경우

local 지역변수

- function call() { let age = 20; } → 함수 안에 생성한 경우

산술 (+ - / %)

```
let num1 = 10;
let num2 = 3;
document.write(num1/num2 + "<br>"); //실수
document.write(num1%num2 + "<br>"); //나머지
```

관계 == , === (값과 타입도 비교), != , >= …

```
let a = 3;
let b = 5;
console.log((a == b));
console.log((a != b));
console.log((a > b));
```

논리 && ||

```
console.log((10 > 5) && (1 != 3));
console.log((10 > 5) || (1 != 3));
```

삼항 연산자

```
let result2 = (4%2 == 0) ? "짝수" : "홀수";
```

대입 연산자(+=, -=, *=, /=)

```
let p = 10;
let k = 5;
p += k; // p = p + k;
console.log(p);
```

```
let x = 5+5;
let y = "5" + 5; // + (산술, 결합) >> 결합
```

```
function 함수명 (파라미터 블럭) {
    실행 블럭
}
```

- void(x), return type(x)

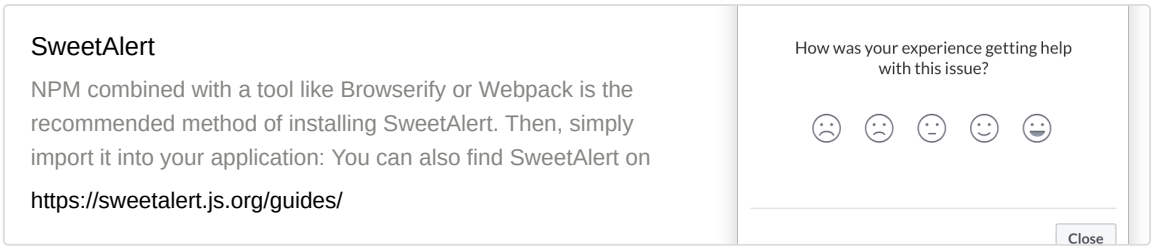
---

confirm() 내장함수 return true or false

```
<script type="text/javascript">
    function callConfirm(){
      if(window.confirm("삭제 할건가?")){ //confirm() 내장함수 return true or false
        alert("네");
      } else{
        alert("아니오");
      }
    }
</script>

<body>

  <h3>내장함수</h3>
  <button onclick="callConfirm()">클릭 사건 발생(함수(행위) 호출)</button>

</body>
```

팝업창

- 요즘은 div 태그를 만들어 띄움
- sweetAlert 사용함



팝업창 생성 전통적 방법

```
<script>
  function showPopup(){
      window.open("Ex06_popup.html", "zipcode", "width=200, height=200"); //내장함수
  }
</script>

<button onclick="showPopup()">팝업창</button>
```

팝업창 Object 생성

```
<script>
  function myFunc(){
    let popupwindow = window.open("Ex06_popup.html", "zipcode", "width=200, height=200");
    //popupwindow가 팝업창 주소 담김
    popupwindow.document.write("<p>this is zipcode window</p>")
  }
<script>

<button onclick="myFunc()">팝업 Object</button>
```

---

javascript는 파라미터의 변수명만 작성

```
<script>
  function goUrlTime(url){
    window.setTimeout("location.href='" + url + "'", 3000);
    //3초가 지나면 첫번째 파라미터를 실행
  }
</script>
```

- location.href= “ “ → 특정 위치로 이동

익명 함수

- 이름 없음
- 한번만 쓸거임 -> 재사용 x
- 원칙 : changeColor = function();
- javascript는 function도 객체로 인정
    - 함수도 주소를 가진다고 생각함

```
changeColor = function(){
    window.document.bgColor = "silver";
    window.document.fgColor = "red";
}
```

콜백함수

- 프로그램 논리에 의해 자동 호출되는 함수

window.setInterval()

```
<!DOCTYPE html>
<html>
<body>

<h1>The Window Object</h1>
<h2>The setInterval() Method</h2>

<p id="demo"></p>

<script>
  const element = document.getElementById("demo");
  setInterval(function() {element.innerHTML += "Hello"}, 1000);
</script>

</body>
</html>
```

- 1초 간격으로 첫번째 파라미터인 함수를 호출
- 문법

> myInterval = setInterval(*function*, *milliseconds*);

로그인 검증 로직

- kosa.or.kr 접속 > 회원가입 클릭 > 회원가입 페이지 (client)

1. client >> 회원가입 입력(id, pwd) >> 서버로 전송 >> 서버가 값을 받아요 >> 검증 >> Client >> 응답
2. client >> 회원가입 입력(id, pwd) >> 전송(x) >> javascript 검증 >> 서버로 전송

- 클라이언트에서 할 수 있는 모든 것은 Client에서 해결

- 유효성 검증 >> 문제가 없으면 >> 서버로 전송

```
<script type="text/javascript">
  function send(){
//유효성 검증 >> 문제가 없으면 >> 서버로 전송
```

```
            //원하는 값이 아니면 재입력 요청

            //1.요소 정보 가져오기
            //document.forms[0].elements[0].value ...
            //권장 : DOM script 사용 (document.getElementById()) 모든 요소는 id = ""

            let userid = document.loginform.txtuserid;
            //userid => input 태그를 가리킴
            console.log(userid);
            console.log(userid.value);

            //DOM 반드시 요소의 id 식별자가 있어야 함
            let pwd = document.getElementById("txtpwd");

            //검증 로직 개발자 마음 ...
            //1. 입력 했는지 안했는지
            //2. 입력한 값이 로직에 맞는지 확인
            if(userid.value == "" || (userid.value.length >= 3 && userid.value.length <= 10)){
                alert("다시 입력해 주세요");
                document.loginform.txtuserid.fucus();
                userid.select();
            }else{
                alert("검증완료");
                //서버로 전송
                //form action="Ex08_login.jsp"
                document.loginform.action = "Ex08_login.jsp";
                document.loginform.submit(); //form태그가 submit 함수를 내장하고 있음
            }
        }
    </script>
```