1. **함수 사용**

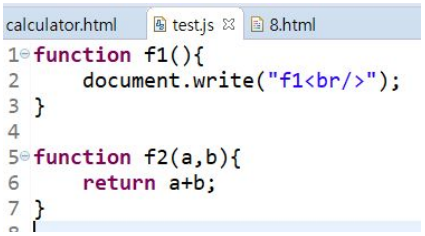
   A. **javascript 파일로 기능 끌어다 html에서 사용하기**

      a. .js 파일 생성 방법

         => new => general ⇒ file => test.js

      b. 함수 정의 *html에서 직접 선언할 경우 head에 씀

      - 타입 쓰지 않고 정의한다.



      c. html에서 js파일 함수 끌어다가 쓰기(import)

      - src로 이미 작성되어 있는 자바 스크립트를 가져오겠다.

```
<head>
<meta charset="EUC-KR">
<title>Insert title here</title>
<script type="text/javascript" src="test.js"></script>
</head>

<body>
<script>
f1()
res = f2(2,3)
document.write("2+3="+res)
</script>
</body>
</html>
```

      *상대경로는 못 불러오는 경우 있음.
      따라서 절대경로를 사용해서 참조하는 것이 안전

   B. **html 내에서 함수 선언/사용하기**

      - 변수를 함수 밖에서 선언 => 전역변수
      - 변수를 함수 안에서 선언 => 지역변수
      - 변수를 함수 안에서 선언 없이 사용 => 전역변수

```
<head>
<meta charset="EUC-KR">
<title>Insert title here</title>
<script type="text/javascript">
var x=10; //전역변수(선언 o)
function f1(){
    var x=100;//지역변수(선언 o)
    y=200;//전역변수(선언 x)
    alert("x:"+x+"/y:"+y);/*active(가장 가까운) 영역의 변수 사용*/
    document.write("f1<br/>");
}

function f2(a,b){
    alert("x:"+x+"y:"+y); /*전역 변수 끌어다 사용*/
    return a+b;
}

</script>
</head>
```

```
<body>
<script>
f1()
res = f2(2,3)
document.write("2+3="+res)
</script>
</body>
```

/*결과*/

2. javascript 이벤트 핸들링

- Event :: UI에 일어나는 사건
  ex. click, double-click,
  focus(여러 개 창이 띄워져 있을 때 가장 위에 띄워져 있는 창이 focus를 얻었다고 표현),
  blur(focus 얻었다가 잃은 경우),
  submit(submit 버튼을 누른 경우),
  mouseover(마우스를 특정 범위에 올려놓은 경우),
  mouseout, change(입력한 값이 변경된 경우),
  key-down(키보드를 누른경우),
  key-up(눌렀다가 뗀 경우).....
- 이벤트 핸들러 등록 속성 => on + event명
- 사용 예시:: button이 클릭되면 a함수를 호출하는 코드

```
<title>Insert title here</title>
<script type="text/javascript">
function a(){
    alert("button clicked");
}
</script>
</head>
<body>
<form action="">
    <input type="button" value="click" onclick="a()">
</form>
</body>
```

3. 구조

window(창을 제어하는 객체) > document(해당 창에 텍스트나 이미지를 뿌리는 객체) > (폼)form > (폼 안의 구성요소. 입력양식)text, password, checkbox, radio…
ex. f.id.value => f이름을 가진 form의 id의 value값
*window와 document는 하나밖에 없어서 생략하고 사용.
ex. window.alert (x) alert(o)

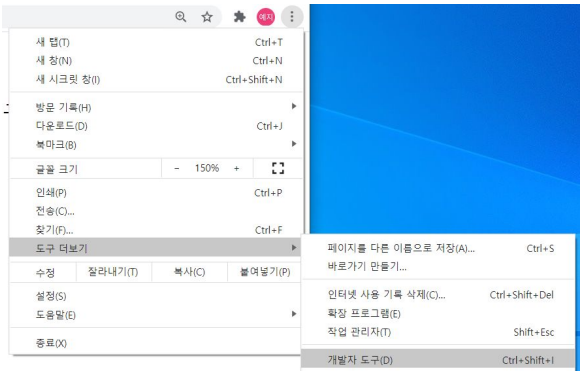
```
function check(){
    f.id.value = "아이디를 입력하라";
    alert(f.id.value.length);/*id 길이 확인*/
    /* alert(f.id.value); */
}
```

- 예제1: 회원가입(id, pwd 길이 check)

```
<script type="text/javascript">
function check(){
    if (f.id.value.length<6){
        alert("아이디는 6글자 이상");
        f.id.focus();
        return;
    }

    if (f.pwd.value.length<6){
        alert("비밀번호는 6글자 이상");
        f.id.focus();
        return;
    }
    var html = "가입 정보\n";
    html += "id:"+f.id.value+"\n";
    html += "pwd:"+f.pwd.value+"\n";
    var h = "";
    //hobby(input): 해당 이름의 checkbox 복수개이기 때문에 hobby라는 배열을 가짐
    for(i=0;i<f.hobby.length;i++){
        //checked란? 해당 박스 check되었는지 확인.
        //form에서 사용될 경우 default로 항목을 check 해놓음.
        if(f.hobby[i].checked){
            h+=f.hobby[i].value+" / ";
        }
    }
    html += "hobby:"+h+"\n";
    var g = f.gender[0].checked ? "f":"m";
    html += "gender:"+g+"\n";
    //grade(select): option이 복수개이기 때문에 options라는 배열을 가짐
    html += "grade:"+f.grade.options[f.grade.selectedIndex].value+"\n";
    html += "msg:" + f.msg.value;
    alert(html);
    res = confirm("가입정보가 맞습니까?");
    if(res){
        f.submit();//폼 데이터를 서버로 전송
    }else{
        alert("가입이 취소되었습니다.")
    }
}
```

- javascript error 확인



- 연습문제: 계산기
    1) 입력한 숫자 계속 이어지게
    2) = 누르면 계산 결과 나오게

- 내 코드

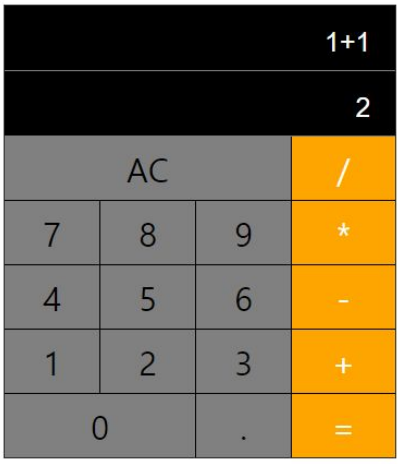
```
<title>계산기</title>
<script>
function add(char){
    //식 입력할 인풋 태그 불러오기:getElementById('name')
    var display = document.getElementById('display');
    display.value = display.value + char;
}
function calculate(){
    //식 계산할 인풋 태그 불러오기:getElementById('name')
    var display = document.getElementById('display');
    var result = eval(display.value);
    //계산할 값 넣어줄 인풋 태그 불러오기
    document.getElementById('result').value = result;
}
function reset(){
    document.getElementById('display').value = "";
    document.getElementById('result').value = "";
}
</script>
```

```
<style>
/*테이블*/
table{
    border-collapse: collapse;/*테두리 한줄로 정리*/
}
/*모든 td 태그*/
td{
    padding: 5px 10px;/*셀 내부 공간 부여*/
    text-align: center;/*글자 가운데 정렬*/
}
/*모든 input 태그*/
input{
    text-align: right; /*오른쪽 정렬*/
    border: none;/*테두리 제거*/
}
input:focus{
    outline: none;
}
/*상단 입출력창*/
.screen{
    background-color:black;
}
```

```
/*왼쪽 사이드*/
.leftside{
    color:black;
    background-color:gray;
    border-color:black;
}
/*오른쪽 사이드*/
.rightside{
    color:white;
    background-color:orange;
    border-color:black;
}
/*입력창*/
#display{
    background-color:black;
    color:white;
}
/*출력창*/
#result{
    background-color:black;
    color:white;
}
</style>
</head>

<body>
<table border = '1'>
    <tr>
        <td class=screen colspan='4'>
            <input type="text" id="display" style="color:white; background-color:black;">
        </td>
    </tr>
    <tr>
        <td class=screen colspan='4'>
            <input type="text" id="result" style="color:white; background-color:black;">
        </td>
    </tr>
    <tr>
        <td class=leftside colspan='3' onclick="reset()">AC</td>
        <td class=rightside onclick="add('/')">/</td>
    </tr>
    <tr class=leftside>
        <td onclick="add(7)">7</td>
        <td onclick="add(8)">8</td>
        <td onclick="add(9)">9</td>
        <td class=rightside onclick="add('*')">*</td>
    </tr>

    <tr class=leftside>
        <td onclick="add(4)">4</td>
        <td onclick="add(5)">5</td>
        <td onclick="add(6)">6</td>
        <td class=rightside onclick="add('-')">-</td>
    </tr>
    <tr class=leftside>
        <td onclick="add(1)">1</td>
        <td onclick="add(2)">2</td>
        <td onclick="add(3)">3</td>
        <td class=rightside onclick="add('+')">+</td>
    </tr>
    <tr class=leftside>
        <td colspan='2' onclick="add(0)" >0</td>
        <td onclick="add('.')">.</td>
        <td class=rightside onclick="calculate()">=</td>
    </tr>
</table>
</body>
```



- 선생님 코드 => calculator.html

- 예제2: 이미지 변환

```
<script type="text/javascript">
//onload:(페이지 시작하자마자)
window.onload=function{//익명함수
}
```

참고:

```
<script type="text/javascript">
var arr = ['daum.png','img1.png','naver.png','universe.png','universe2.jpg'];
idx = 0

function prev(){
    idx--;
    document.img.src = '../img/'+arr[idx%5];
}
function next(){
    idx++;
    document.img.src = '../img/'+arr[idx%5];
}

</script>
</head>
<body>
<img src="../img/daum.png" name='img'><br/>
<input type="button" value="prev" onclick="prev()">
<input type="button" value="next" onclick="next()">
</body>
```

- 연습문제: 방명록

- 흐름
  a. 사용자가 시작페이지(글 목록) URL로 접근시도
  b. List Controller(servlet) :: 사용자의 request를 받아 해당 request를 list.jsp에 전달 *forward 방식(db에서 받아 request에 저장한 내용 그대로 전달)
  c. list.jsp :: 글 목록 페이지
  d. 사용자가 글작성 버튼 클릭
  e. FormController(servlet) :: 사용자의 request를 받아 해당 request를 form.jsp에게 전달. *forward 방식?
  f. form.jsp :: 글 작성 페이지. 작성완료후 write controller에게 request 전달
  g. Write Controller(servlet) :: request 받아서 새 글을 db에 저장 *sendredirect 방식(새로고침 때마다 해당 글 추가 방지)
  h. List Controller(servlet) :: 새 db 받아서 request에 담아 list.jsp에 전달

i. list.jsp :: 글 목록 페이지(갱신된 모습)

1) DB

--테이블 생성

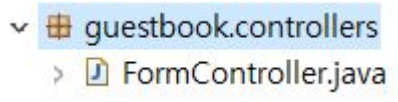
```
create table guestbook(
num number primary key,
writer varchar2(20) not null,
pwd varchar2(20) not null,
w_date date,
content varchar2(100)
);
```

--시퀀스 생성

```
create sequence seq_questbook;
```

2) db 연동 설정: ojdbc6.jar lib에 넣어두기
3) MODEL 생성
   - VO
   - DAO(interface-Impl)
   - SERVICE(interface-Impl)
4) Controller 생성(servlet: 클라이언트로 부터 요청을 받아 view를 바로 보여줄 것인지, model을 동작시킬 것인지를 결정)

=>guestbook.controllers 패키지 생성 => new => CREATE servlet (FormController)

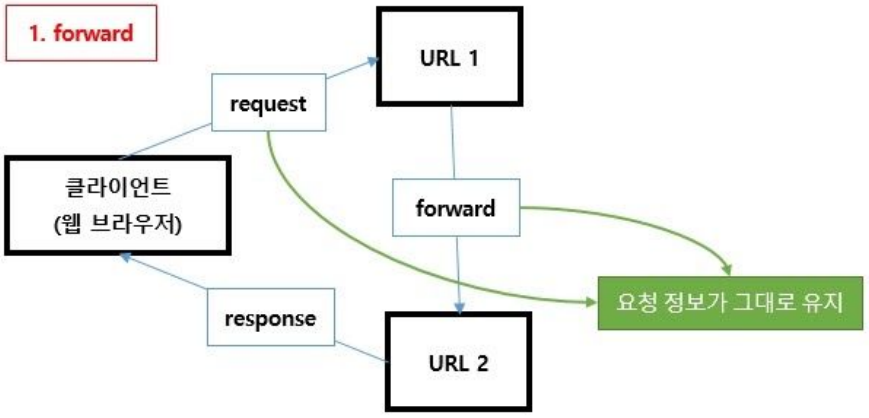


- URL 변경(실제 폴더 아니더라도 가상주소로 사용 가능)

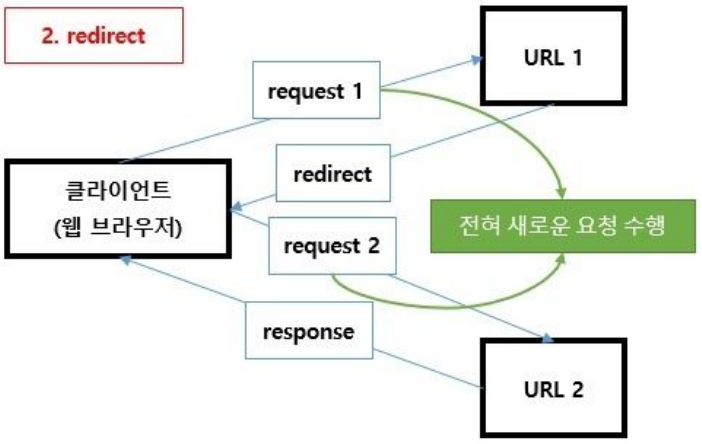
```
@WebServlet("/FormController")
```

```
@WebServlet("/guestbook/form")
```

- 페이지 이동 방식[사용자의 동작은 동일하지만 처리되는 과정이 상이]
  1) forward 방식: 사용자가 요청할 때 생성된 request 객체
     서버에서 response 객체를 보내기 전까지 유지됨
     : a.jsp -> b.jsp 등으로 이동할 때도 처음에 받은 response객체를 동일하게 사용
     *새로고침할 경우: 해당 url이 계속 반복되므로 추가/수정/삭제 등 기능 계속 실행됨



2) sendredirect 방식
: a.jsp -> b.jsp 등으로 이동할 때는 사용자에게 새로운 요청을 하게 유도
*새로고침해도 새로운 url으로 변경되므로 이상 무



redirect의 경우 최초 요청을 받은 URL1에서 클라이언트에 redirect할 URL2를 리턴하고, 클라이언트에게 전혀 새로운 요청을 생성하여 URL2에 다시 요청을 보낸다. 따라서 처음 보냈던 최초의 요청정보는 더이상 유효하지 않게된다.

다시 정리해보자면 redirect와 forward의 차이점은 크게 두가지로 나눌 수 있다.
첫째, URL의 변화여부(변화 O -> redirect, 변화 X -> forward)
둘째, 객체의 재사용여부(재사용 O -> forward, 재사용 X -> redirect)

위와 같은 차이점 때문에 웹 애플리케이션을 작성할 때 forward와 redirect 두 가지 방식 중 하나를 적절히 선택하여 사용해야한다. 예를 들어 게시판 애플리케이션을 작성한다고 해보자. 사용자가 보낸 요청정보를 이용하여 글쓰기 기능을 수행하는 CGI(Common Gateway Interface)가 있다면, 이 CGI의 응답 페이지는 forward와 redirect 중 어느 것을 사용해야 될까? 정답은 redirect이다.
사용자가 실수 혹은 고의로 글쓰기 CGI응답 페이지에서 새로고침을 누르면 어떻게 될까? forward의 경우 요청정보가 그대로 살아있기 때문에 똑같은 글이 여러번 등록될 수 있다. 하지만 redirect의 경우 처음 글을 작성할 때 보냈던 요청정보는 존재하지 않는다. 또한 글쓰기 기능을 하는 URL1이 아닌 URL2로 요청을 보내기 때문에 글쓰기가 여러번 수행되지 않는다.
즉, 시스템(session, DB)에 변화가 생기는 요청(로그인, 회원가입, 글쓰기)의 경우 redirect방식으로 응답하는 것이 바람직하며, 시스템에 변화가 생기지 않는 단순조회(리스트보기, 검색)의 경우 forward방식으로 응답하는 것이 바람직하다.

5) guestbook 폴더에 form.jsp 파일 생성

- String writer;
  String pwd;
  String content;
  입력필수사항이기 때문에 체크해주기
- action:/guestbook/write이라는 이름의 jsp
  method:post

6) write controller 서블릿 파일 생성
   url “/guestbook/write” 인 서블릿 파일 생성
   request.getParameter()로 폼파라메터 받아옴
   service 객체의 addArticle() 호출해서 글 추가
   requestDispatcher로 forward로 view 페이지로 이동
   view page url “/guestbook” 시작화면으로 이동