[12차시 교안]

<자바스크립트 객체, Dom과 이벤트 처리>

1. Array 객체

가. 자바스크립트에서 배열로 객체 표현

나. 배열을 나타내는 객체 : Array

다. 배열 사용 방법 : array 객체를 new 연산자로 생성하고 배열 요소에 데이터 넣음

```
var myArray = new Array();
myArray[0] = "apple";
myArray[1] = "banana";
myArray[2] = "orange";
```

라. 특징

① 배열의 크기가 자동으로 조절됨

```
var myArray = new Array( );
myArray[0] = “apple”;
myArray[99] = “orange”;
```

② 여러 가지 자료형을 혼합해서 저장 가능

```
var myArray = new Array( );
myArray[0] = “apple”;          // 문자열 저장
myArray[1] = new Data( );      // 객체 저장
myArray[2] = 3.14;             // 실수 저장
```

2. Array 속성과 메소드

(1) 속성

가. length

① 가장 많이 사용되는 속성

② 배열의 길이를 나타냄

③ 배열의 요소를 반복 처리할 사용

(2) 메소드

가. concat(value1[value2[value]]) : 전달된 인수를 배열 끝에 추가

```
<script>
   var x = [1, 2, 3];
   var y = [4, 5, 6];
   var joined = x.concat(y);

   document.writeln(x);    // 출력: 1,2,3
   document.writeln(joined);// 출력: 1,2,3,4,5,6
</script>
```

나. indexOf(searchStr[, starIndex]) : 요소의 값을 가지고 요소의 인덱스 검색

```
<script>
   var fruits = ["apple", “orange", "grape"];
   document.writeln(fruits.indexOf(“orange"));
</script>
```

다. pop( ), push(value)

① push() : 스택에 데이터 삽입하는 메서드

② pop() : 스택에서 데이터 꺼내는 메서드

```
<script>
   var numbers = [1, 2, 3, 4, 5];

   numbers.push(6);
   document.writeln(numbers + '<BR>');        // 출력: 1,2,3,4,5,6
   item = numbers.pop();
   document.writeln(numbers + '<BR>');        // 출력: 1,2,3,4,5,
</script>
```

라. shift( ), unshifit( )

① shift() : 배열의 첫 번째 요소 반환하고 이 첫 번째 요소를 배열에서 제거

② unshift() : shift() 메소드와 함께 사용하면 큐 구현 가능

```
<script>
    var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

    var item = numbers.shift();
    document.writeln(item + '<BR>');        // 출력: 1
    document.writeln(numbers + '<BR>');     // 출력: 2,3,4,5,6,7,8,9,10
</script>
```

마. sort( ) : 배열을 알파벳순으로 정렬

① 숫자 정렬을 위해서는 정렬 기준 함수 필요

② 매개 변수로 함수를 가질 수 있음

바. Array.slice([begin[, end]]))

① 배열의 일부를 복사하여 새로운 배열로 반환

② 시작 인덱스가 없으면 0으로 가정

③ 종료 인덱스가 없으면 배열의 끝까지 복사

사. join(delimeter)

① 배열의 요소들을 하나의 문자열로 출력

② 배열을 서버로 보낼 때 유용

③ 분리자(delimeter)가 각 요소 분리

아. filter( )

① 어떤 조건에 부합하는 요소만을 선택하여 새로운 배열로 만들어서 반환

② 요소를 선택하는 함수를 작성하여 인수로 전달

```
<script>
    var numbers = [10, 20, 30, 40, 50];
    function isBigEnough(element, index, array) {
      return (element >= 30);
    }

    var filtered = numbers.filter(isBigEnough);
    document.write("필터링 된 배열: " + filtered);
  </script>
```

3. 자바스크립트에서의 오류처리

가. 자바스크립트에서 오류가 발생하면 자동적으로 실행 중단되고 대화상자 등장

나. 대화상자에서 확인 버튼을 누르면 소스 파일에서 오류 발생 위치를 보여 줌

다. 자바스크립트에서는 오류를 예외 라고 함

라. 예외(exception)

① 프로그램의 실행 중에 발생하는 이벤트

② Exceptional event 의 약자

③ 발생 원인

a. 개발자의 타이핑 오류로 인한 문법적 오류

b. 브라우저마다 지원하는 특징이 다름

c. 사용자의 잘못된 입력

d. 인터넷 서버 오류

마. 예외 처리(exception handling)

① 자바스크립트에서의 예외 처리기는 try 블록과 catch 블록

② try 블록에서 예외 발생할 수 있음

③ catch 블록에서 예외 처리

```
<html>
<head>
   <script>
      var msg = "";
      function test() {
         try {
            allert("Hello World!");
         }
         catch (error) {
            msg = "다음과 같은 오류가 발생하였음: " + error.message;
            alert(msg);
         }
      }
   </script>
</head>
<body>
   <input type="button" value="try-catch 시험" onclick="test()" />
</body>
</html>
```

바. throw 문장

① 개발자가 오류를 생성할 수 있도록 함

② 예외를 발생시키는 것을 예외를 던진다고 표현함

③ 예외를 고의로 발생시키는 이유

개발자가 어떤 기준을 정하고 이 기준에 맞지 않으면 사용자에게 어떤 경고 메시지를 주기 위해서

④ throw 문장을 사용하여서 오류 처리 가능

⑤ try-catch 문장 사용 : (예) 숫자 맞추기 게임

사. 예외 처리 예제

① 사용자가 1부터 100사이의 있는 특정 숫자를 추출하는 게임

② 사용자가 입력한 값과 정답을 비교해서 높은지, 낮은지를 예외로 생성

③ 값을 잘못 입력하는 경우에도 예외 생성

④ 예외는 즉시 catch 문장에 잡히고 예외에 포함된 메시지가 화면에 표시됨

```
<h1>Number Guess</h1>
  <p>1부터 100 사이의 숫자를 입력하시오.</p>
  <input id="number" type="text">
  <button type="button" onclick="test()">숫자 추측</button>
  <p id="message"></p>
</body>
</html>
```

```
<html>
<body>
  <script>
    var solution = 53;      // 컴퓨터가 정한 정답
    function test() {
      try {
        var x = document.getElementById("number").value;  // 사용자 입력값
        if (x == "") throw "입력 없음";
        if (isNaN(x)) throw "숫자가 아님";
        if (x > solution) throw "너무 큼";
        if (x < solution) throw "너무 작음";
        if (x == solution) throw "성공";
      }
      catch (error) {
        var y = document.getElementById("message");
        y.innerHTML = "힌트: " + error;
      }
    }
  </script>
```

4. 문서 객체 모델(Document Object Model)

(1) 정의

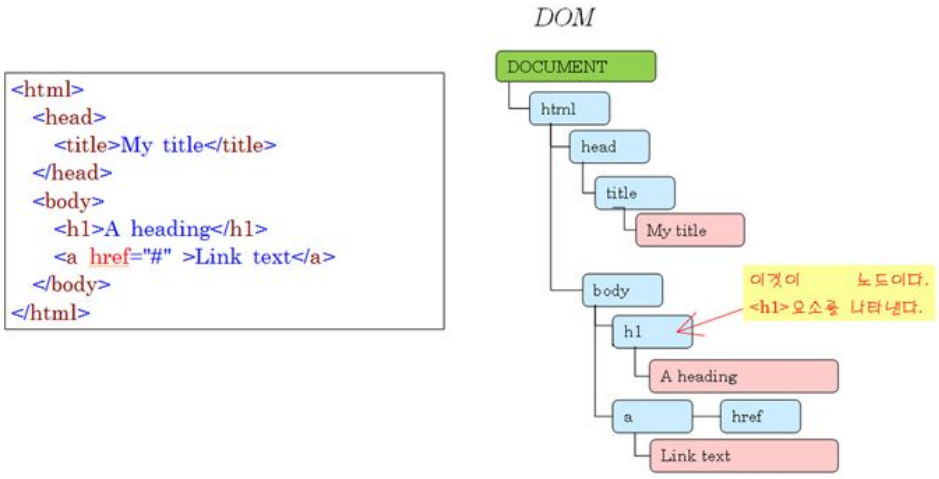
가. 웹 페이지가 적재되면, 브라우저는 페이지의 문서 객체 모델(DOM) 생성

나. DOM은 HTML 문서의 계층적인 구조를 트리(tree)로 표현

다. 트리에 있는 하나의 잎을 노드(node)라고 함

라. DOM에서 노드는 문서 안에 들어있는 요소나 텍스트를 나타냄

마. 예



(2) DOM의 이용

가. 자바스크립트를 사용하여 DOM 트리를 순회하면서 페이지 안 요소의 내용이나 속성, CSS 스타일 변경 가능

① 동적으로 웹페이지 내용과 스타일 변경 가능

나. 각각의 요소에서 발생하는 이벤트에 반응하는 코드 작성 가능

(3) DOM과 BOM

가. DOM(Document Object Model) : HTML 문서를 객체로 표현한 것

나. BOM(Browser Object Model) : 웹브라우저를 객체로 표현한 것

(4) DOM에 존재하는 노드의 종류

가. DOCUMENT_NODE

① DOM 트리의 루트 노드

② HTML 문서를 나타냄

③ window.document

나. ELEMENT_NODE

① HTML 요소를 나타내는 노드

② <body>, <a>, <p>, <script>, <style>, <html>, <h1> 등

다. ATTRIBUTE_NODE

① 속성을 나타내는 노드

② 예 : class="myclass"

라. TEXT_NODE

① 요소 안에 들어 있는 텍스트

5. HTML 요소 찾기

동적인 웹페이지를 작성하려면 원하는 요소를 찾아야 함

(1) 원하는 요소를 찾는 방법

가. 요소의 id로 찾기(가장 간단한 방법) : getElementByID( ) 이용

① 요소를 객체 형태로 반환

② 요소 내용이 아니라 요소 자체 반환

```
<html>
<head>
   <script>
      function process() {
         var obj = document.getElementById("target");
         alert(obj.value);
      }
   </script>
</head>
<body>
   <form name="myform">
      <input type="text" id="target" name="text1">
      <input type="submit" value="제출" onclick="process()">
   </form>
</body>
</html>
```

나. 입력 양식 찾기

HTML 문서 안 모든 입력 양식은 forms 배열 객체에 들어 있으므로 배열 안에서 원하는 입력 양식 검색

```
<html>
<head>
   <script>
      function process() {
         var obj = document.myform.text1;
         alert(obj.value);
      }
   </script>
</head>
<body>
   <form name="myform">
      <input type="text" id="target" name="text1">
      <input type="submit" value="제출" onclick="process()">
   </form>
</body>
</html>
```

다. 태그 이름으로 찾기 : getElementByTagName( ) 이용

태그 이름을 인수로 받아서 이 태그를 사용하는 모든 요소를 배열에 넣어 반환

```
<html>
<body>
   <ul>
      <li>List item 1</li>
      <li>List item 2</li>
      <li>List item 3</li>
      <li>List item 4</li>
      <li>List item 5</li>
   </ul>
   <script>
      var list = document.getElementsByTagName('ul')[0];   // 문서 내의 첫번째 <ul> 요소 찾음
      var allItems = list.getElementsByTagName('li');          // <ul> 안의 <li> 모두 찾음
      for (var i = 0, length = allItems.length; i < length; i++) {
         alert(allItems[i].firstChild.data);
         }
   </script>
</body>
</html>
```

라. DOM 트리 순회

자식노드와 부모 노드 관계를 이용하여 한 노드씩 방문하는 방법

① childNodes : 한 요소의 모든 자식 요소에 접근 가능. 배열 반환

② firstChild : 'childNodes' 배열의 첫 번째 자식 노드 반환. childNodes[0]

③ lastChild : 'childNodes' 배열의 마지막 자식 노드 반환.
childNodes[childNodes.length-1]

④ parentNode : 현재 노드의 부모 노드 반환

⑤ nextSibling : 현재 노드의 다음 형제 노드 반환

⑥ previousSibling : 현재 노드의 이전 형제 노드 반환

6. HTML 요소의 내용 변경

자바스크립트를 이용하여 HTML 문서의 DOM을 변경하는 방법

(1) 요소의 내용 변경

가. innerHTML 속성 사용 : 요소의 시작태그와 종료 태그 사이에 놓여진 모든 HTML 코드와 텍스트

나. innerHTML 속성 : (텍스트) 또는 (HTML + 텍스트)

```
<html>
<head>
   <title></title>
   <script>
      function get() {
         var val = document.getElementById("ex").innerHTML; //요소 내용 출력
         alert(val);
      }
      function set(v) {
         document.getElementById("ex").innerHTML = v;    // 요소 내용 변경
      }
   </script>
</head>
<body>
   <div id="ex">여기가 div로 선언되었습니다.</div>
   <a href="#" onclick="get()"> div 요소 내용 출력하기</a><br />
   <a href="#" onclick="set('변경되었습니다.')"> div 요소 내용 변경하기</a>
</body>
</html>
```

(2) 요소의 속성 변경 : 요소가 가진 속성에 대한 동적 변경

```
<html>
<body>
   <img id="image" src="pome.png" width="120" height="100">
   <script>
      function changeImage() {
         document.getElementById("image").src = "poodle.png";
      }
   </script>
   <input type="button" onclick="changeImage()" value="눌러보세요" />
</body>
</html>
```

(3) 요소의 스타일 변경

```
<html>
<body>
<p id="p1">This is a paragraph.</p>
<script>
   function changeStyle() {
      document.getElementById("p1").style.color = "red";
      document.getElementById("p1").style.fontFamily = "Century Schoolbook";
      document.getElementById("p1").style.fontStyle = "italic";
   }
</script>
    <input type="button" onclick="changeStyle()" value="눌러보세요" />
</body>
</html>
```

7. Dom 노드 삭제와 추가

(1) 새로운 HTML 요소 생성

가. 첫번째 : 추가하기를 원하는 노드 생성(document 객체의 createTextNode())

나. 두번째 : 문서 내에서 추가할 위치 검색

다. 세번째 : 새로운 노드를 기존의 노드에 연결(appendChild( ) 사용)

```
<script>
   function addtext(t) {
      if (document.createTextNode) {  // 이 기능이 지원되는지 검사
         var node = document.createTextNode(t);  // 노드 생성
         document.getElementById("target").appendChild(node); //노드 추가
      }
   }
</script>
<div id="target" onclick="addtext('동적으로 텍스트가 추가됩니다.')"
  style="font: 20px bold;">여기를 클릭하세요.</div>
```

(2) 기존의 HTML 요소 삭제

삭제하고자 하는 요소와 그 부모 요소를 알아야 함

```
<html>
  <head>
  <script>
    function removeNode( ) {
      var parent = document.getElementById("target");
      var child = document.getElementById("p1");
      parent.removeChild(child);
    } // 부모노드를 통해 자식 노드 삭제
  </script>
  </head>
<body>
  <div id="target">
    <p id="p1">첫번째 단락</p>
    <p id="p2">두번째 단락</p>
  </div>
  <button onclick="removeNode( )">누르세요!</button>
</body>
</html>
```

8. 브라우저 객체 모델

(1) 브라우저 객체 모델(BOM: Browser Object Model)

가. 웹 브라우저가 가지고 있는 모든 객체를 의미

나. 최상위 객체는 window

다. 그 아래로 navigator, location, history, screen, document, frames

(2) Window 객체

가. BOM에서 최상위 객체로서 웹 브라우저 윈도우를 나타냄

나. 모든 전역 자바스크립트 객체, 함수, 변수는 자동적으로 window 객체의 멤버

9. 새로운 윈도우 오픈 : open() 메서드

가. 새로운 브라우저 오픈

나. 매개변수

① url : 오픈 할 페이지 URL

② name : 타켓을 지정하거나 윈도우 이름

③ specs : 여러 가지 속성

④ replace : 히스토리 리스트에서 새로운 엔트리인지 아니면 현재 엔트리를 대체하는지 여부(true-replace, false-create)

```
<html>
<head></head>
<body>
  <form>
    <input type="button" value="구글창 열기"
      onclick="window.open('http://www.google.com', '_blank',
                                    'width=300, height=300', true)">
  </form>
</body>
</html>
```

10. setTimeout()

가. 일정 시간이 지난 후, 인수로 전달된 함수를 딱 한번 호출

나. 시간 단위 : 밀리초

다. 버튼을 누르면 3초 후에 경고 윈도우를 오픈하는 예제

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function showAlert() {
      setTimeout(function ( ) { alert("setTimeout()을 사용하여 표시됩니다.") },
                  3000);
    }
  </script>
</head>
<body>
  <p>버튼을 누르면 3초 후에 경고 박스가 화면에 표시됩니다. </p>
  <button onclick="showAlert()">눌러보세요</button>
</body>
</html>
```

11. setInterval()

가. 일정 시간마다 주기적으로 함수 호출

나. setInterval( )은 반드시 개발자가 종료시켜야 함

주기적인 호출 종료 : clearInterval( ) 호출

다. 시간 단위 : 밀리초

라. "중지" 버튼을 누를 때까지 글자를 깜빡이는 프로그램

```
<html>
<head>
  <script>
    var id;
    function changeColor() {
      id = setInterval(flashText, 500);
    }
    function flashText() {
      var elem = document.getElementById("target");
      elem.style.color = (elem.style.color == "red") ? "blue" : "red";
      elem.style.backgroundColor =
                (elem.style.backgroundColor == "green") ? "yellow" : "green";
    }
    function stopTextColor() {
      clearInterval(id);
    }
  </script>
</head>
```

```
<body onload="changeColor();">
  <div id="target">
    <p>This is a Text.</p>
  </div>
  <button onclick="stopTextColor();">중지</button>
</body>
</html>
```

12. 이벤트 처리

(1) 웹 페이지에서 상호작용이 발생하면 이벤트가 일어남

가. 마우스 클릭

나. 웹 페이지 로딩

다. 호버링으로 마우스를 어떤 요소 위에서 움직이는 것

라. HTML 입력 양식에서 입력 박스를 선택하는 것

마. 키보드의 키를 누르는 것

(2) onclick 이벤트

사용자가 라디오 버튼을 클릭하면 <body>요소의 색상이 변경되는 프로그램

```
<html>
<head>
   <script>
      function changeColor(c) {
         document.getElementById("target").style.backgroundColor = c;
      }
   </script>
</head>
<body id="target">
   <form method="POST">
      <input type="radio" name="C1" value="v1"
         onclick="changeColor('lightblue')">파랑색
      <input type="radio" name="C1" value="v2"
         onclick="changeColor('lightgreen')">녹색
   </form>
</body>
</html>
```

(3) onload와 onunload 이벤트

가. onload 이벤트 : 사용자가 웹 페이지 진입 시

나. onunload 이벤트 : 사용자가 웹 페이지를 떠날 때

다. 페이지가 로드되면 경고 대화상자를 띄우고 페이지의 배경색을 빨간색으로 변경하는 프로그램

```
<html>
<head>
   <script>
      function onLoadDoc() {
         alert("문서가 로드되었습니다.");
         document.body.style.backgroundColor = "red";
      }
   </script>
</head>
<body onload="onLoadDoc();">
</body>
</html>
```

(4) onchange 이벤트

가. onchange( ) : 입력 필드 검증 시 종종 사용

나. 사용자가 입력 필드의 내용을 변경하면 소문자로 변경하는 프로그램

```
<html>
<head>
<script>
   function sub() {
      var x = document.getElementById("name");
      x.value = x.value.toLowerCase();
   }
</script>
</head>
<body>
영어단어: <input type="text" id="name" onchange="sub()">  //엔터치면 실행
<p>입력필드를 벗어나면 소문자로 변경됩니다.</p>
</body>
</html>
```

(5) onmouseover 이벤트

가. Onmouseover : 사용자가 HTML 요소 위에 마우스를 올릴 때

나. Onmouseout : 사용자가 HTML 요소를 떠날 때

```
<html>
<head>
   <script>
      function OnMouseIn(elem) {
         elem.style.border = "2px solid red";    // 마우스가 영역에 들어오면 경계선 두껍게
      }
      function OnMouseOut(elem) {
         elem.style.border = "";
      } //마우스가 밖으로 나가면 경계선 없앰
   </script>
</head>
<body>
   <div style="background-color: yellow; width: 200px"
      onmouseover="OnMouseIn (this)" onmouseout="OnMouseOut (this)">
      마우스를 이 요소 위에 이동하세요.
   </div>
</body>
</html>
```

(6) onmousedown 이벤트

가. onmousedown : 마우스 버튼이 클릭될 때

나. onmouseup : 마우스 버튼이 떼어질 때

다. onclick : 마우스 클릭이 완료될 때

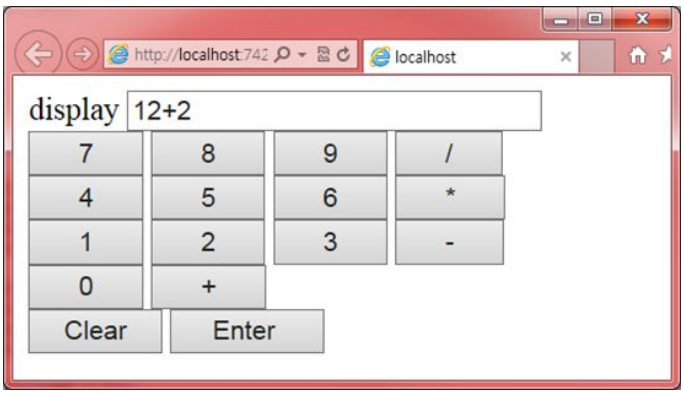
```
<html>
<head>
  <script>
    function OnButtonDown(button) {
            button.style.color = "#ff0000";  //빨강
    }
    function OnButtonUp(button) {
      button.style.color = "#000000";  //검정
    }
  </script>
</head>
<body>
  <button onmousedown="OnButtonDown (this)" onmouseup="OnButtonUp (this)">
   눌러보세요! </button>
</body>
```

13. 계산기 예제

가. 모든 버튼이 클릭되면 버튼에 쓰여 있는 문자를 expression 변수에 추가

나. 이 문자를 인수로 하여 eval( ) 함수를 호출하면 계산값을 얻을 수 있음

① eval( ) : 문자열 형태의 자바스크립트 소스를 받아서 동적으로 실행한 후에 결과값 반환하는 함수



```
<body>
  <form>
    display <input id="display" value="0" size="30">
    <br>
    <input type="button" value="   7   " onclick="add('7')">
    <input type="button" value="   8   " onclick="add('8')">
    <input type="button" value="   9   " onclick="add('9')">
    <input type="button" value="   /   " onclick="add('/')">
    <br>
    <input type="button" value="   4   " onclick="add('4')">
    <input type="button" value="   5   " onclick="add('5')">
    <input type="button" value="   6   " onclick="add('6')">
    <input type="button" value="   *   " onclick="add('*')">
    <br>
    <input type="button" value="   1   " onclick="add('1')">
    <input type="button" value="   2   " onclick="add('2')">
    <input type="button" value="   3   " onclick="add('3')">
    <input type="button" value="   -   " onclick="add('-')">
    <br>
    <input type="button" value="   0   " onclick="add('0')">
    <input type="button" value="   +   " onclick="add('+')">
    <br>
    <input type="button" value="  Clear  " onclick="clearDisplay()">
    <input type="button" value="  Enter  " name="enter" onclick="compute()">
  </form>
</body>
</html>
```

```
<html>
<head>
  <script>
    var expression="";
    function add(character) {
      expression = expression + character;
      document.getElementById("display").value = expression;
    }
    function compute() {
      document.getElementById("display").value = eval(expression);
    }
    function clearDisplay() {
      expression = "";
      document.getElementById("display").value = "0";
    }
  </script>
```