

웹퍼블리싱 강의

2022. 6

강 현 준

human@human.or.kr

강의 목차

1. HTML
2. CSS
3. JAVA SCRIPT
4. JQuery
5. AJAX

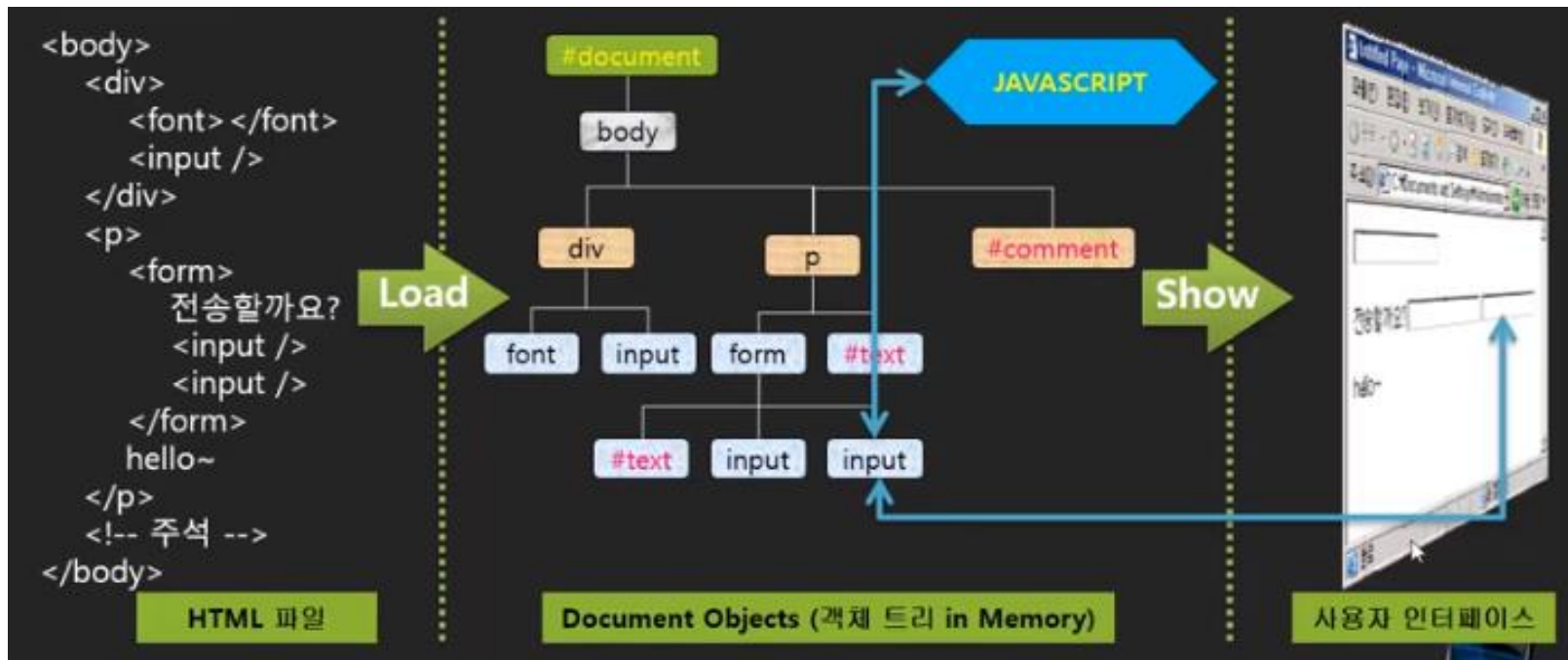
3장. Java Script

3.1. JAVA Script 란?

- 1단계 : HTML 문서란 document 객체로 변환하여 로딩
- 2단계 : document 객체 기반으로 화면에 보여짐
- 중간에서 JAVA Script는 이 객체의 정보에 접근하여 관리가 가능함.

JAVA Script

JAVA Script 역할

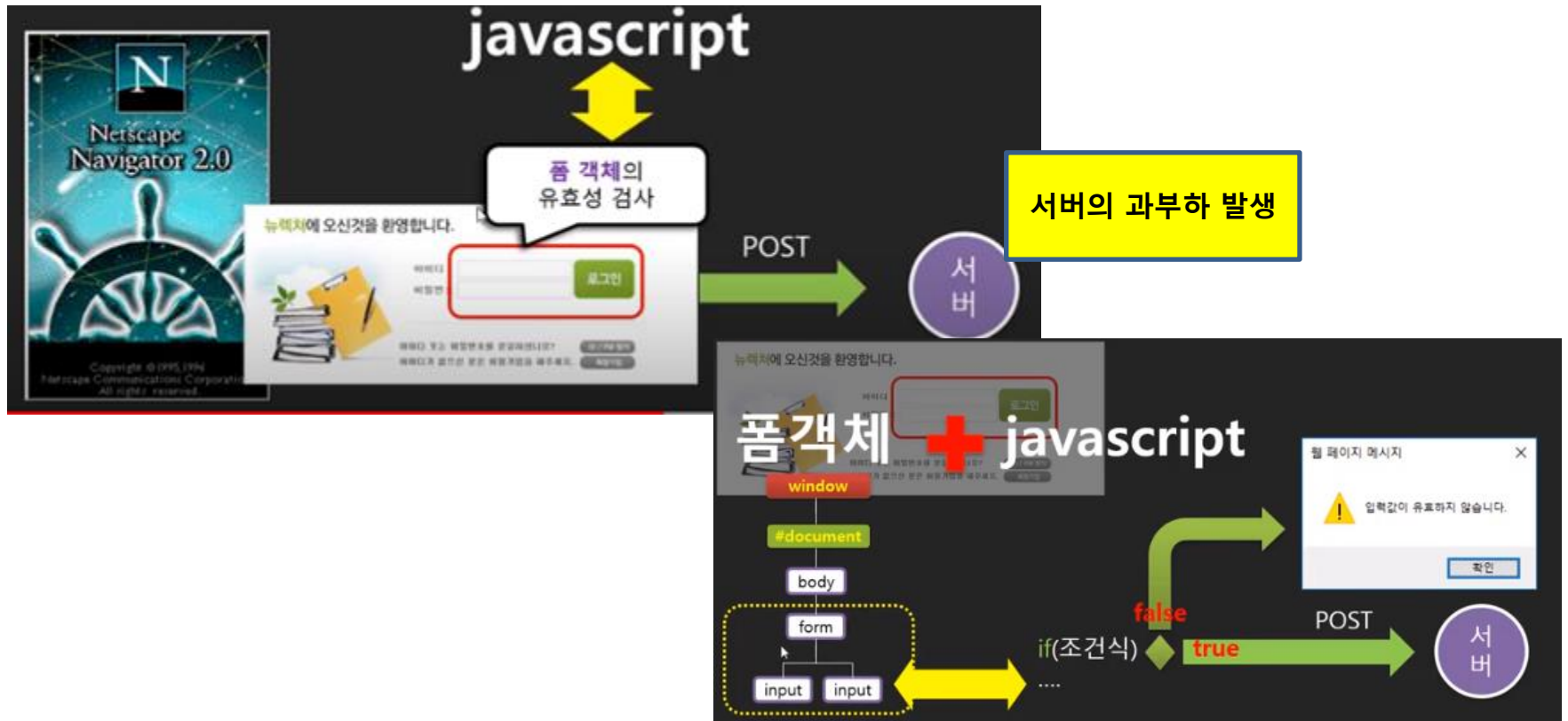


3.2. JAVA Script 탄생 배경

- WWW가 활성화되면서 호출시 유효하지 않은 데이터 기반으로 호출이 잦아 유효성 검사 필요함
- 화면의 form 객체를 제어하는 프로그램을 통해서 이 문제를 해결함

JAVA Script 탄생배경

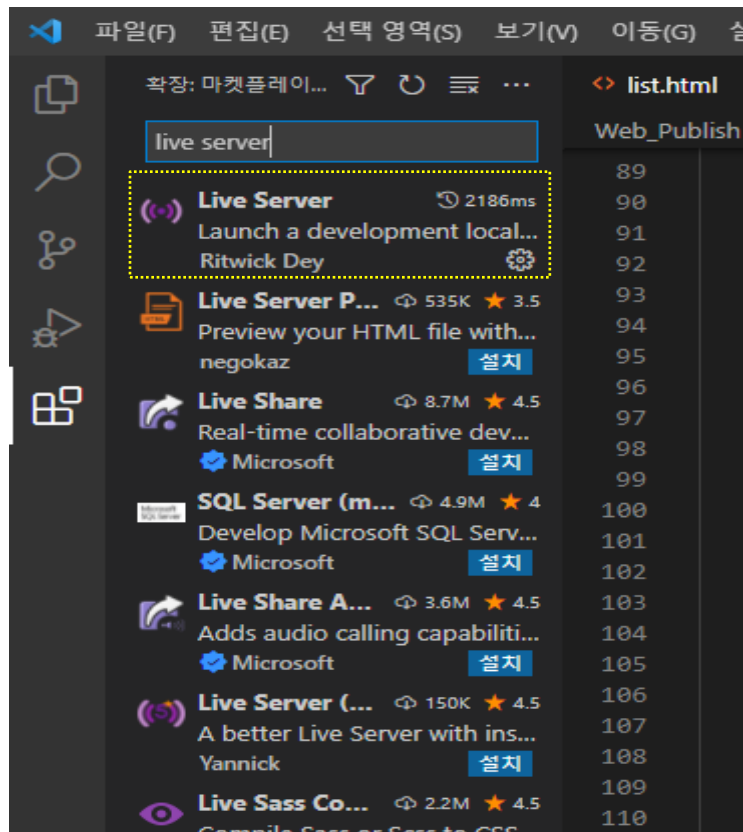
탄생배경



3.3. 실습환경 준비

- Visual studio code와 live Server설치

실습환경 준비



Visual studio code



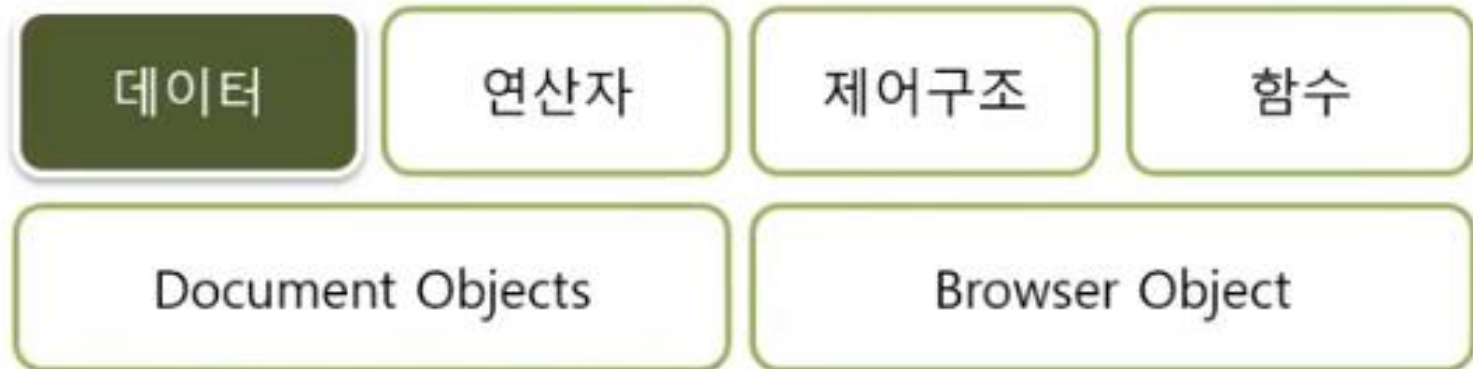
Live server

3.4. JAVA Script 프로그램 범위

- 자바스크립트는 다른 프로그래밍언어와 비슷한 범주가 있음
- Document Objects, Browser Objects에 대한 이해도 필요함

JAVA Script 범위

JAVA Script 범위



3.4.1. 데이터 – 값의 종류 및 변수

- 자바스크립트의 변수의 선언은 3가지가 있음.
- 변수의 선언방법 : let, var, const

변수의 종류

- 자바스크립트는 다른 프로그램과 달리 프로그램이 실행되면서 값의 타입을 정의함.
- 변수의 타입을 사전에 정의 안해도 됨. (객체지향의 레퍼런스 타입으로 정의됨)
- 모두 var, let, const 로 변수를 선언하면 됨.

정수	실수	문자	문자열
int	double	char	String
3	3.7	'A'	"Hello"

```
var x = 3;  
var x = 3.7;
```

```
var x = 'A';  
var x = "Hello";
```

```
var x;  
x = 3;
```


3.4.2. 데이터 - 참조형 변수

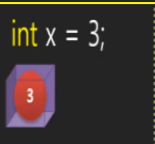
- 자바스크립트는 실행시점에 데이터의 크기를 확인하여 처리함 (auto-boxing)
- 변수 타입 : Boolean, Number, String

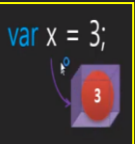
변수의 종류

- 객체지향의 Literal 값을 참조하는 형태로 변수가 생성됨. (참조형 변수)
- 그래서 다른 값이 들어가도 에러가 안남.

부울	정수	실수	문자	문자열
Boolean	Number	Number	String	String

`var x = 3;`
`var x = new Number(3);`

`int x = 3;`


`var x = 3;`


[값의 타입확인]

```
<script>

/* 시작
    alert ("Hello");

    var x = new Number(3);
    var y = 3;
    alert (x+y);
    z=3;
    alert (typeof z); // number가 나옴.
*/
```

3.4.3. 데이터 – 배열 변수

- 자바스크립트는 배열변수를 push, pop 메서드를 통해서 데이터 관리 (Stack 방식)

배열변수

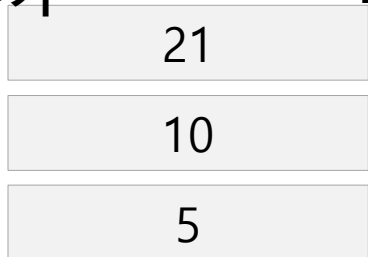
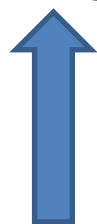
```
var nums = new Array();

nums.push(5);    var n1 = nums.pop();
alert(nums);    alert(nums);

nums.push(10);   var n2 = nums.pop();
alert(nums);    alert(nums);

nums.push(21);   var n3 = nums.pop();
alert(nums);    alert(nums);
```

PUSH :
데이터 쌓기



POP :
데이터 제거



[Stack 실습예제]

```
/*
var nums = new Array();
nums.push(3);
nums.push(13);
nums.push(25);
console.log(nums); // f12 눌러서 개발자 도구 또는 화면에서 찾기.

var n1 = nums.pop();
console.log(n1);
console.log(nums);

console.log(nums.pop());
console.log(nums.pop());

var num_test = new Array();
num_test[3]=100;
console.log(num_test);
*/
```

3.4.4. 데이터 - 배열 변수 초기화

- 자바스크립트는 변수의 형식이 느슨하기 때문에 다양한 데이터가 섞일 수 있다

배열변수 초기화

```
var nums = new Array();    배열의 초기화
var nums = new Array(5);   5개 공간을 준비함
var nums = new Array(5, 10, 21); 2개 이상일 때는 초기값으로 인식
var nums = new Array(5, 10, 21, "hello"); 가능함
alert(typeof nums[3]);    String 확인됨
var nums = new Array(5, 10, 21, "hello", new Array(2, 3, 4)); 배열 내에 배열 초기화 가능
alert(nums[4][1]);        4번째의 2번째 값을 반환 → "3"
```

[5, 10, 21, "Hello", [2,3,4]]

3.4.5. 데이터 – list 관리

- Splice 메서드를 통해서 데이터의 list 관리가 가능함

배열변수 데이터 관리

[splice 메서드를 통한 list 관리]

```
var nums = new Array(5, 10, 21, "hello");
```

`nums.splice(1)` `nums.splice(1, 1)` `nums.splice(1, 2)`

[실습예제]

```
var num1 = new Array(2,3,"hello",7);
console.log (num1);
num1.splice(2); // 3~4번이 삭제됨.
num1.splice(2,1); // 3번째에서 1개만 삭제됨.
num1.splice(2,1, "HUMAN"); // 3번째에서 1개만 삭제 후, 그곳에 HUMAN 넣어라..
num1.splice(2,0, "COMP"); // 3번째에서 0개만 삭제 후, 그곳에 COMP 넣어라.. ==> 이는 삽입을 의미함.
console.log (num1);
```

3.4.6. 데이터 – Object 객체

- 배열변수는 Object 객체를 활용하여 생성이 가능함.
- 배열변수는 Key-Value 의 형태로 사용가능함 → JSON 태동

배열변수 데이터 관리

Object 객체 활용

```
var exam = new Object();  
exam.kor = 30;  
exam.eng = 70;  
exam.math = 80;
```

```
alert(exam.kor + exam.eng);
```

→ 100

Key를 이용한 데이터 관리 (MAP)

```
var exam = new Object();  
  
exam["kor"] = 30;  
exam["eng"] = 70;  
exam["math"] = 80;
```

```
alert(exam["kor"]);
```

→ 30

```
exam = {"kor":30, "eng":70, "math":80}
```

실습예제

```
// javascript는 object라는 것을 제공하는데, 객체지향처럼 사용가능.  
// 그리고 클래스 선언없이 사용가능함.  
var exam = new Object();  
exam.kor = 90;  
exam.eng = 80;  
exam.math = 70;  
console.log(exam.kor + exam.math);  
  
// JAVA에서는 MAP이란 것을 제공하는데.  
// javascript에서는 eng는 key이고, 80은 value에 해당함.  
var key = "eng";  
console.log(exam[key]);
```

3.4.7. 데이터 – JSON

- JSON : JavaScript Object Notation
- 2차원 데이터를 관리하기 쉬운 JavaScript 타입.

JSON

실습예제

```
var exam = {"kor":90, "eng":80, "math":70};  
console.log(exam.kor + exam.math);  
  
var ar = [3,4,5,6,exam, [7,8,9]];  
console.log(ar[1]);  
console.log(ar[4]);  
console.log(ar[4][0]);      // undefined  
console.log(ar[4]["kor"]);  // exam.kor 출력  
console.log(ar[4].math);    // exam.math 출력  
console.log(ar[5]);         // 7,8,9 배열 출력  
console.log(ar[5][1]);      // 7,8,9 배열 중 2번째
```

2차원으로 구성된 데이터를
Key : value의 형태로
데이터 관리가 가능함.

```
var notices = [  
  {"id":1, "title":"hello json"},  
  {"id":2, "title":"hi json"},  
  {"id":3, "title":"json is ..."},  
];  
↓  
notices[1]
```

3.4.8. 데이터 관리의 방법 – CSV, XML, JSON

- 데이터 관리 방법 : CSV, XML, JSON
- 브라우저에서 해석하기 쉬운 JSON 방식으로 많이 사용함

JSON 활용의 타당성

XML : 코딩량이 많음

```
<notices>
  <notice id="1" title="hello json" />
  <notice id="2" title="hi json" />
  <notice id="2" title="title":"json is ..." />
</notices>
```

JSON이 많이 사용됨

- 예제 : 공공데이터, upbit

선택

```
[
  {"id":1, "title":"hello json"},
  {"id":2, "title":"hi json"},
  {"id":3, "title":"json is ..."}
]
```

```
1, hello json
2, hi json
3, json is ...
```

CSV

- 칼럼이 없음
- 복잡한 데이터 표현 어려움

3.4.9. JSON Parse 예제

- 문자열로 들어온 JSON 데이터를 Parse를 통해서 데이터 관리 할 수 있음

JSON Parse

실습예제

```
/*
    var notice1 = {id:1, title:"human"}; //문제없으나 보통 아래와 같이 사용해야함. 묵시적으로 문자열로 해석함.
    var notice2 = {"id":1, "title":"human"}; //Key에 해당하는 것은 문자열이 되어야 함.
    var notice3 = {"n id":1, "title":"human"}; //Key에 해당하는 것은 문자열이 되어야 함.
    console.log (notice3["n id"]); //이렇게 사용가능함.

    //
    var notice4 = {"id":1, "title":"human"}; //데이터는 이와 같은 형태의 문자열로 옴.
    console.log (notice4);

    //
    console.log (JSON.parse(notice1)); 문자열이 아닐때는 에러 발생

    var data1 = JSON.parse(notice4); // JSON은 까다롭게 형태를 관리함. 문자열로 올때만 파싱 가능함.
    console.log (data1); // 정상처리됨.

    //
    그러면 notice1 같이 올때는 어떻게 할 것인가?
    data2 = JSON.stringify(notice1); // 문자열로 묶어서 처리됨.
    console.log (data2, data2[0], data2[1] ); // 문자열로 묶인 것 확인.

    //
    이제는 data2가 문자열로 묶였기 때문에 JSON 처리 가능함.
    final_data2 = JSON.parse(data2);
    console.log (final_data2, final_data2["id"], final_data2["title"] ); // data 파싱 가능.
*/
```


3.5. 연산자

- JAVA 스크립트의 연산자

연산자



기본 값	숫자: 12 / 문자열 : "hello"
산술연산자	+, -, *, /, %
비교연산자	<, >, <=, >=, ==, !=, ===, !==
관계연산자	&&,

3.5.1 연산자

- 값의 비교 : == 또는 !=
- 주소의 비교 : === 또는 !==

var x = 3;
(JAVA스크립트는 참조형 변수)

값 비교 및 주소의 비교

```
var x=3;  
var y=3;  
document.write (x==y);    //값을 비교함.  
document.write ("<br>");  
document.write (x===y);   //주소를 비교함.
```

true

true

```
document.write ("<br>");  
document.write ("<br>");
```

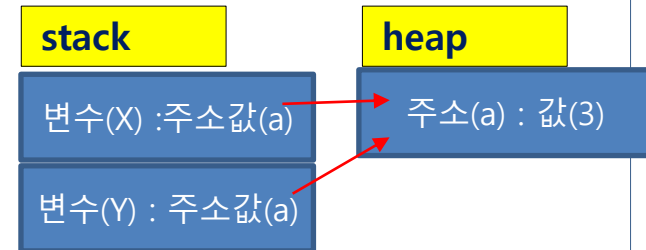
```
var x=3;  
var y=new Number(3);  
document.write (x==y);    //값을 비교함.  
document.write ("<br>");  
document.write (x===y);   //주소를 비교함.
```

true

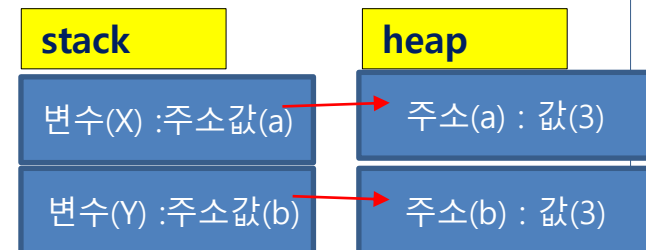
false

```
document.write ("<br>");  
document.write (typeof(x));    //타입 검토함.
```

- * document.write : html 문서로 출력
- * console.log : 개발자 모드에서 log 확인
- * alert : 팝업으로 alert 처리



Y는 새로운 객체를 생성하였으므로
새로운 주소로 운영됨



3.6. 제어구조

- JAVA 스크립트의 제어구조

제어구조

데이터

연산자

제어구조

함수

Document Objects

Browser Object

조건문

if, Else, elseif

반복문

while, for, for-in

선택문

else, else if, switch

```
<script>
  var ar = ["철수","영희","맹구","동천"];

  /* for(var i=0; i<4; i++)
    alert(ar[i]); */

  for(i in ar)
    alert(ar[i]);
</script>
```

3.6.1 제어구조

- JAVA 스크립트의 제어구조

제어구조

```
var ar = ["HUMAN", "human", "Computer", "World"];
```

```
for (var i=0; i<ar.length; i++) {  
  document.write (ar[i]+"<BR>");  
}  
document.write ("<BR>");
```

```
for (i in ar) {  
  document.write (i + ". " + ar[i]+"<BR>");  
}  
document.write ("<BR>");
```

```
var ob = {id:"1",title1:"human", title2:"Computer", title3:"World"};  
for (a in ob) {  
  // a는 JSON의 KEY가 들어감.  
  document.write (a + ". " + ob[a]+"<BR>");  
}
```

HUMAN
human
Computer
World

0. HUMAN
1. human
2. Computer
3. World

id. 1
title1. human
title2. Computer
title3. World

3.7. 함수

- JAVA 스크립트는 함수는 있으나 다른 언어처럼 정의하지는 않고, 만드는 형태임.

함수

데이터

연산자

제어구조

함수

Document Objects

Browser Object

```
int add(int x, int y){  
    return x+y;  
}
```

일반적인
함수 정의



```
var add = new Function("x, y", "return x+y;");
```

```
alert(add(3, 4));
```

자바스크립트
함수 생성

```
var add = function (x, y){  
    return x+y;  
}
```

```
function add(x, y){  
    return x+y;  
}
```

3.7.1. 함수

- JAVA 스크립트는 함수 3가지 형태 숙지 필요.
- AJAX 실습시 활용 예정

함수 실습

```
// 함수의 문법 이해 필요.
```

```
var add1 = new Function("x,y", "return x+y");  
alert (add1(3,4));  
document.write (add1(3,4));
```

정상 수행됨

```
var add2 = function(x,y) {  
    return x+y;  
}  
alert (add2(3,4));  
document.write (add2(3,4));
```

정상 수행됨

```
function add3(x,y) {  
    return x+y;  
}  
alert (add3(3,4));  
document.write (add3(3,4));
```

정상 수행됨

3.7.2. 함수 - 주의사항

- 일반 프로그램의 함수와는 달리 호출인자와 매핑되는 것이 아니라
- 호출인자는 arguments에 넣어놓고, 앞에서부터 채우는 방식임

함수 실습

```
function add(x,y),  
{  
  
    return x+y;  
};
```

arguments 라는 곳에 담아두고
함수내에서 꺼내어 사용

```
var sum = add(2, 3);
```

이상없음

```
var sum = add(2, 3, "hello", 3, 4, 5, 6, 7, 87);
```

에러 없음

```
/* 가변 함수 */  
// 에러가 나지 않는 이유는 arguments라는 곳에 담아두고, 담아둔 것을 하나씩 매핑하는 것이기 때문에.  
/*  
    function add1(x,y) {  
        document.write (arguments);  
        document.write ("<BR>");  
        document.write (arguments[0], arguments[1], arguments[5]);  
        document.write ("<BR>");  
  
        return x+y;  
    }  
    document.write (add1(3,4,5,6,7,"Hello"));  
*/
```

3.7.3. 함수 - 지역변수 / 전역변수

- 변수 선언 : var, let, const임
- var가 포함되지 않은 변수는 전역변수로 활용됨. (window.변수명)

전역변수 / 지역변수

- 전역변수 : 프로그램 전체를 대상으로 활용 가능한 변수
- 지역변수 : 해당 영역안에서만 활성화 되는 변수
 - . 지역안에 있을 때만 메모리에 살아있으며, 지역을 벗어날 경우는 메모리에서 소멸됨

```
/* 전역변수 */  
// var로 선언하지 않으면 window 객체의 변수로 인식하여 전역변수가 됨.  
/*  
    function f1(){  
        var a=1;    // 지역변수이므로 결과 나오지 않음.  
    //      a=1;      // window.a 라는 전역변수로 인식하여 결과 나옴.  
    }  
    f1();  
    console.log(a);  
*/
```



이런 문제들이 있기 때문에
var 대신 let 사용을 권장함

3.8. Browser 플랫폼

- 브라우저 플랫폼 : 브라우저 객체를 활용
- 브라우저 객체 핸들링을 통해서 브라우저의 활용이 가능함

브라우저 플랫폼 및 객체

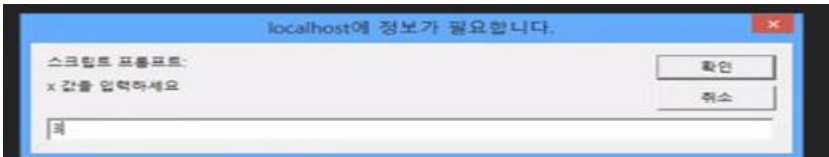


3.8.1. Browser 플랫폼 – 사용자와 상호 가능한 메서드

- alert : 사용자에게 알람을 주는 메서드
- prompt : 사용자에게 입력을 받는 메서드
- confirm : 사용자의 선택을 확인하는 메서드

사용자와 소통하는 메서드

prompt 메서드



```
<script>
var x, y;
x = prompt("x 값을 입력하세요", 0);
y = prompt("y 값을 입력하세요", 0);
alert(x+y);
</script>
```

```
/* window 객체(1) - prompt */
// x=eval(window.prompt("x값 입력 : ", 0));
// y=eval(window.prompt("y값 입력 : ", 0));
// window.alert (x+y);

// x=parseInt(window.prompt("x값 입력 : ", 0));
// y=parseInt(window.prompt("y값 입력 : ", 0));
// window.alert (x+y);

// x=parseFloat(window.prompt("x값 입력 : ", 0));
// y=parseFloat(window.prompt("y값 입력 : ", 0));
// window.alert (x+y);
```

confirm 메서드

```
<script>
var answer;
answer = confirm("정말로 삭제하시겠습니까?");
if(answer)
    alert("삭제 되었습니다.");
</script>
```

```
/* window 객체(2) - confirm */
/*
    var answer = confirm ("정말로 삭제하실 것인지요?");
    window.document.write (answer);           // true, false

    window.document.write ("<BR>");

    if (answer) window.document.write ("삭제");
    else window.document.write ("취소");
*/
```

3.8.2. Browser 플랫폼 – 이벤트 기반 프로그래밍

- 사용자 이벤트에 의해 처리하는 방식
- 사용자 이벤트 : Keyboard 입력 / 마우스 move / 마우스 클릭 등이 있음

이벤트 기반 프로그램 처리 Flow

처리 문법

```
<script>
```

페이지가 읽혀
질 때 실행

```
</script>
```

```
<input onXXX=" " />
```

이벤트가 발생
할 때 실행

```
<input onclick=" " />
```

```
<input onmouseover=" " />
```

confirm 메서드

```
<body>
```

```
<input type="button" value="출력"  
      onclick="alert('안내 메시지');" />
```

```
</body>
```

버튼 클릭시 alert 발생

3.8.3. Browser 플랫폼 – 이벤트 기반 함수호출

- 사용자 이벤트를 함수적으로 호출하여 프로그램의 간소화 가능

이벤트 기반 함수 호출

복잡한 코드

```
<input type="button" value="출력"  
onclick="var x, y; x=prompt('x 값을 입력  
하세요'); y=prompt('y 값을 입력하세  
요'); alert(x+y);" />
```

간결한 코드

```
<script>  
function printResult()  
{  
    var x, y;  
    x = prompt("x 값을 입력하세요", 0);  
    y = prompt("y 값을 입력하세요", 0);  
    alert(x+y);  
}  
</script>  
</head>  
<body>  
<input type="button" value="출력"  
onclick="printResult();" />
```

3.8.4. Browser 플랫폼 – 객체 활용하여 HTML 문서 변경

- 윈도우 객체에 값 전달하여 HTML 문서의 변경

JavaScript를 통한 문서의 변경

윈도우 객체 변경 (속성활용)

```
<input type="button" value ="실행전" onclick ="print2();" id="btnPrint2">  
button 클릭시 value 변경
```

```
function print2() {  
    var x,y;  
    x = prompt ("x 값 입력: ", 0);  
    y = prompt ("y 값 입력: ", 0);  
  
    btnPrint2.value = x+y;  
    btnPrint2.type = "text";  
}
```

Browser의 button의 type과 value
변경 가능함

Element id로 html 문서 변경

```
<input type="button" value ="실행전" id="btnPrint3">  
함수에서 버튼이벤트 정의  
<script>  
    function print3(){  
        var x,y;  
        x = prompt ("x 값 입력: ", 0);  
        y = prompt ("y 값 입력: ", 0);  
        btnPrint3.value = x+y;  
    }  
    btnPrint3.onclick = print3;  
</script>
```

Browser의 button의 type과 value
변경 가능함

3.8.5. Browser 플랫폼 – 여러 스크립트 활용시 문제점

- 윈도우 객체에 값 전달하여 HTML 문서의 변경

JavaScript를 통한 문서의 변경

잘못된 상황

```
<!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>
  <script>
    // 먼저 실행되는데, 실행시점에는 btn1이 아직 활성화안되기 때문에 에러발생.
    // 그래서 window.onload 활용함. ==> index3.html에서 점검.
    function print () {
      var x,y;
      x = prompt ("x 값 입력: ", 0);
      y = prompt ("y 값 입력: ", 0);
      btn1.value = x+y;
    }
    btn1.onclick = print;
  </script>
</head>
<body>

  <input type = "button" value = "클릭" id="btn1">
  <BR>

</body>
</html>
```

Btn1이 나중에 만들어짐

btn1 not define 에러 발생

onload로 해결

```
<!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>
  <script>
    function print () {
      var x,y;
      x = prompt ("x 값 입력: ", 0);
      y = prompt ("y 값 입력: ", 0);
      btn1.value = x+y;
    }
    function init() {
      btn1.onclick = print;
    }
    window.onload = init;
  </script>
</head>
<body>

  <input type = "button" value = "클릭" id="btn1">
  <BR>

</body>
</html>
```

문서가 load 완료까지 대기함.
문서 완료시 btn 객체 생성됨

3.8.6. Browser 플랫폼 – getElementById

- getElementById : 엘리먼트(일반적으로 tag) ID를 통한 엘리먼트 객체 획득

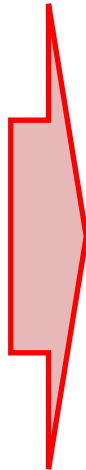
getElementById 처리

실습 - 1

```
<!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-s
  <title>Document</title>
  <script>
    function print1 () {
      var btnPrint1 = document.getElementById("btn1");
      var x,y;
      x = prompt ("x 값 입력: ", 0);
      y = prompt ("y 값 입력: ", 0);
      btnPrint1.value = x+y;
    }
    function init() {
      var btnPrint1 = document.getElementById("btn1");
      btnPrint1.onclick = print1;
    }
    window.onload = init;
  </script>
</head>
<body>

  <input type = "button" value = "클릭1" id="btn1">
  <BR>

</body>
</html>
```



실습 - 2 (function의 합체)

```
window.onload = function() {
  var btnPrint1 = document.getElementById("btn1");
  btnPrint1.onclick = function() {
    // var btnPrint1 = document.getElementById("btn1");
    var x,y;
    x = prompt ("x 값 입력: ", 0);
    y = prompt ("y 값 입력: ", 0);
    btnPrint1.value = x+y;
  };
};
```

3.8.7. Browser 플랫폼 – onload시 2개의 함수 호출방법

- onload를 통해 1개의 함수 호출은 가능하나 2개의 기능을 순차적 실행은 불가함.
- 이럴 때는 addEventListener를 통해서 처리 가능

addEventListener처리



Main 프로그래머

```
<!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,
  <title>Document</title>
  <!-- onLoad를 두개에 동시에 사용불가. -->
  <!-- 그래서 addEventListener가 필요한 -->
  <script src="index6_1.js"></script>
  <script src="index6_2.js"></script>
</head>
<body>
  <input type = "button" value = "클릭1" id="btn1">
  <BR>
</body>
</html>
```

```
window.addEventListener ("load", function() {
  var btnPrint1 = document.getElementById("btn1");
  btnPrint1.onclick = function() {
    // var btnPrint1 = document.getElementById("btn1")
    var x,y;
    x = prompt ("x 값 입력: ", 0);
    y = prompt ("y 값 입력: ", 0);
    btnPrint1.value = x+y;
  } ;
});
```

[index6_1.js]



onload시 동작 구현

```
window.addEventListener ("load", function() {
  alert ("안녕하세요. HUMAN 교육센터입니다.");
});
```

[index6_2.js]



onload시 동작 구현

3.8.8. Browser 플랫폼 – 계산기 실습 예제 (1)

- getElementById를 통해서 value 값 획득 후 처리

계산기 실습예제- 1

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE">
  <meta name="viewport" content="width=device-wi
  <title>Document</title>
  <script src="index7_1.js"></script>
</head>
<body>

  <input type = "text" id="txt1">
  +
  <input type = "text" id="txt2">
  <input type = "button" id="add" value = "=">
  <input type = "text" value=0 id="sum">

</body>
</html>
```

```
window.addEventListener ("load", function() {
  var btnPrint1 = document.getElementById("add");
  btnPrint1.onclick = function() {
    var x,y;
    x = parseInt(document.getElementById("txt1").value);
    y = parseInt(document.getElementById("txt2").value);
    sum.value = x+y;
  };
});
```

3.8.8. Browser 플랫폼 – 계산기 실습 예제 (2)

- getElementById 및 getElementsByTagName
- getElementsByTagName은 배열로 반환됨. (복수형임을 주의할 것)

계산기 실습예제- 2

```
<!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, height=device-height">
  <title>Document</title>
  <script src="index7_2.js"></script>
</head>
<body>
  <section id = "section1">
    <input type = "text" id="txt1">
    +
    <input type = "text" id="txt2">
    <input type = "button" id="add" value = "=">
    <input type = "text" value=0 id="sum">
  </section>
</body>
</html>
```

```
window.addEventListener ("load", function() {
  var btnAdd = document.getElementById("add");
  btnAdd.onclick = function() {
    var section = document.getElementById ("section1");
    var inputs = section.getElementsByTagName("input");
    console.log (inputs);
    console.log (inputs[0].value);

    var x,y;
    x = parseInt(inputs[0].value);
    y = parseInt(inputs[1].value);
    sum.value = x+y;
  };
});
```

3.8.8. Browser 플랫폼 – 계산기 실습 예제 (3)

- getElementById 및 getElementsByClassName
- getElementsByClassName은 배열로 반환됨. (복수형임을 주의할 것)

계산기 실습예제- 3

```
<!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">
  <title>Document</title>
  <script src="index7_3.js"></script>
</head>
<body>
  <section id = "section1">
    <input type = "text" class="txt1">
    +
    <input type = "text" class="txt2">
    <input type = "button" id ="add" value = "=">
    <input type = "text" value=0 id="sum">
  </section>
</body>
</html>
```

```
window.addEventListener ("load", function() {
  var btnAdd1 = document.getElementById("add");
  btnAdd1.onclick = function() {
    var section = document.getElementById ("section1");
    var input_x = section.getElementsByClassName("txt1")[0];
    var input_y = section.getElementsByClassName("txt2")[0];
    console.log (input_x, input_y);

    var x,y;
    x = parseInt(input_x.value);
    y = parseInt(input_y.value);
    sum.value = x+y;
  } ;
});
```

3.8.8. Browser 플랫폼 – 계산기 실습 예제 (4)

- getElementById 및 querySelector 활용
- querySelector는 클래스 선택자 사용함.

계산기 실습예제- 4

```
<!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, height=device-height">
  <title>Document</title>
  <script src="index7_4.js"></script>
</head>
<body>
  <section id = "section1">
    <input type = "text" class="txt1">
    +
    <input type = "text" class="txt2">
    <input type = "button" id ="add" value = "=">
    <input type = "text" value=0 id="sum">
  </section>
</body>
</html>
```

```
window.addEventListener ("load", function() {
  var btnAdd1 = document.getElementById("add");
  btnAdd1.onclick = function() {
    var section = document.getElementById ("section1");
    var input_x = section.querySelector(".txt1");
    var input_y = section.querySelector(".txt2");
    console.log (input_x, input_y);

    var x,y;
    x = parseInt(input_x.value);
    y = parseInt(input_y.value);
    sum.value = x+y;
  } ;
});
```

3.8.8. Browser 플랫폼 – 계산기 실습 예제 (5)

- getElementById 및 querySelector 활용
- querySelector는 [속성=value] 활용함.

계산기 실습예제- 5

```
<!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">
  <title>Document</title>
  <script src="index7_5.js"></script>
</head>
<body>
  <section id = "section1">
    <input type = "text" name="txt1">
    +
    <input type = "text" name="txt2">
    <input type = "button" id = "add" value = "+">
    <input type = "text" value=0 id="sum">
  </section>
</body>
</html>
```

```
window.addEventListener ("load", function() {
  var btnAdd1 = document.getElementById("add");
  btnAdd1.onclick = function() {
    var section = document.getElementById ("section1");
    var input_x = section.querySelector("input[name='txt1']");
    var input_y = section.querySelector("input[name='txt2']");
    console.log (input_x, input_y);

    var x,y;
    x = parseInt(input_x.value);
    y = parseInt(input_y.value);
    sum.value = x+y;
  } ;
});
```

3.8.8. Browser 플랫폼 – 계산기 실습 예제 (6)

- getElementById 및 querySelector, 그리고 ChildNode 활용
- Childnode는 빈공간 등도 인식하므로 사용이 어렵고, tag 기준으로 처리하는 children 활용

계산기 실습예제- 6

```
<!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">
  <title>Document</title>
  <script src="index7_6.js"></script>
</head>
<body>
  <section id = "section1">
    <div class="box">
      <input type = "text" name="txt1">
      +
      <input type = "text" name="txt2">
      <input type = "button" id="add" value = "+">
      <input type = "text" value=0 id="sum">
    </div>
  </section>
</body>
</html>
```

```
// childNode 활용하는 방법
// childNodes와 children의 차이.
// childNodes는 빈공간도 인식하나 children은 tag 기준으로 인식함.
window.addEventListener ("load", function() {
  var btnAdd1 = document.getElementById("add");
  btnAdd1.onclick = function() {
    var section = document.querySelector("#section1");
    var box = section.querySelector (".box")
    console.log (box);

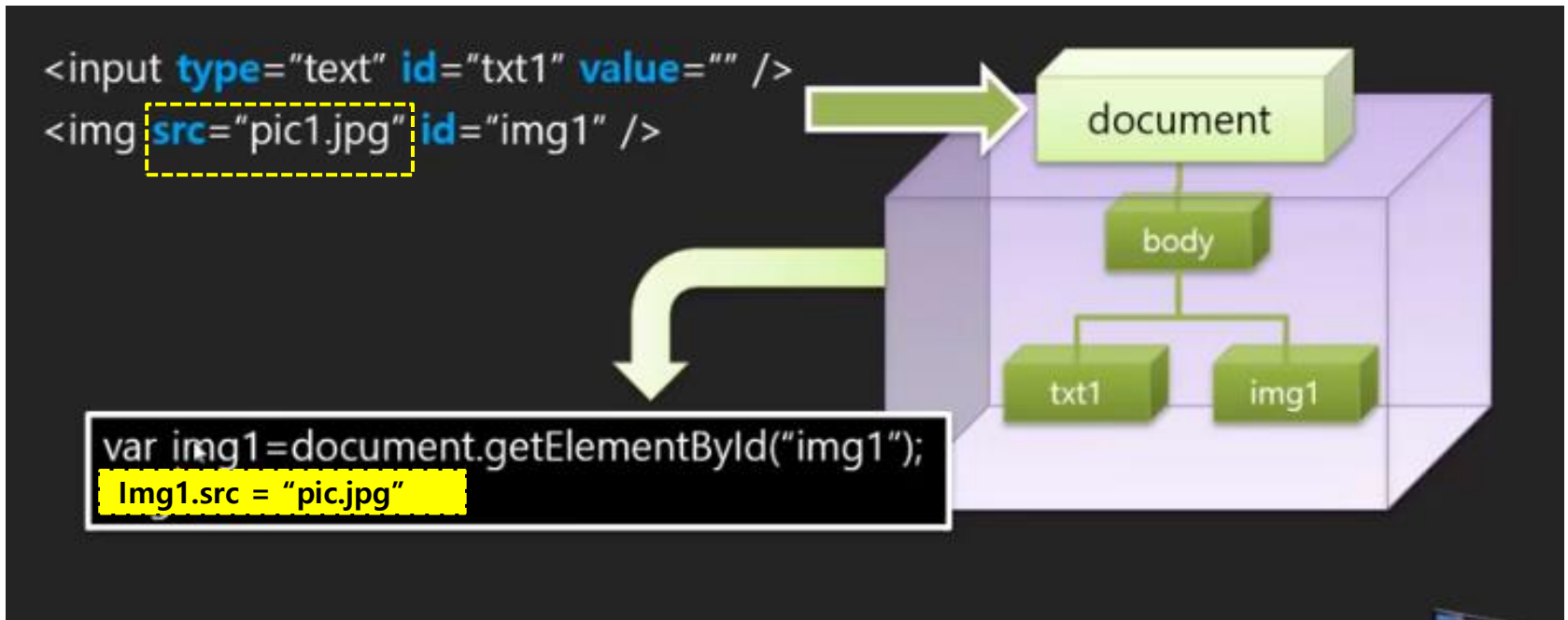
    // var input_x = box.childNodes[0];
    // var input_y = box.childNodes[2];
    var input_x = box.children[0];
    var input_y = box.children[1];
    console.log (input_x, input_y);

    var x,y;
    x = parseInt(input_x.value);
    y = parseInt(input_y.value);
    sum.value = x+y;
  } ;
});
```

3.9. DOM 모델

- DOM : Document Object Model
- 문서내의 Node는 객체이고, 각 객체의 속성값을 통해서 문서를 제어할 수 있음

DOM 모델



3.9.1. DOM 모델 실습 – img 모델 (1)

- img 속성값을 통해서 자바스크립트로 문서 변경하기

DOM 모델 – img(1)

```
<!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>
  <script src="node_01_01.js"></script>
</head>
<body>
  <section id = "section">
    <div>
      <input class="src-input">
      <input class="change-button" type="button" value="변경하기">
    </div>
    <div>
      
    </div>
  </section>
</body>
</html>
```

```
// 이미지 변경
window.addEventListener ("load", function() {
  var section = document.querySelector ("#section");
  var srcInput = section.querySelector(".src-input");
  var changeButton = section.querySelector(".change-button");
  var img = section.querySelector(".img");
  changeButton.onclick = function() {
    // alert ("HI");
    img.src = "images/" + srcInput.value;
  } ;
});
```


3.9.1. DOM 모델 실습 – img 모델 (2)

- img 속성값을 통해서 자바스크립트로 문서 변경하기

DOM 모델 – img(2)

```
<!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">
  <title>Document</title>
  <script src="node_01_02.js"></script>
</head>
<body>
  <section id = "section">
    <div>
      <input class="src-input">
      <select class="img-select">
        <option value="img1.jpg">img1</option>
        <option value="img2.jpg">img2</option>
        <option value="img3.jpg">img3</option>
      </select>
      <input class="change-button" type="button" value="변경">
    </div>
    <div>
      
    </div>
  </section>
</body>
</html>
```

```
window.addEventListener ("load", function() {
  var section = document.querySelector ("#section");
  var srcInput = section.querySelector(".src-input");
  var imgSelect = section.querySelector(".img-select");
  var changeButton = section.querySelector(".change-button");
  var img = section.querySelector(".img");
  changeButton.onclick = function() {
    img.src = "images/" + imgSelect.value;
  };
});
```

3.9.1. DOM 모델 실습 – img 모델 (3)

- img 속성값을 통해서 자바스크립트로 문서 변경하기

DOM 모델 – img(3)

```
<section id = "section">
  <div>
    <input class="src-input" list="img-list">
    <datalist id = "img-list">
      <option value="img1.jpg">img1</option>
      <option value="img2.jpg">img2</option>
      <option value="img3.jpg">img3</option>
    </datalist>
    <select class="img-select">
      <option value="img1.jpg">img1</option>
      <option value="img2.jpg">img2</option>
      <option value="img3.jpg">img3</option>
    </select>
    <input class="change-button" type="button" value=
  </div>
  <div>
    
  </div>
</section>
```

```
window.addEventListener ("load", function() {
  var section = document.querySelector("#section");
  var srcInput = section.querySelector(".src-input");
  var imgSelect = section.querySelector(".img-select");
  var changeButton = section.querySelector(".change-button");
  var img = section.querySelector(".img");
  changeButton.onclick = function() {
    img.src = "images/" + srcInput.value;
  } ;
});
```

3.9.2. DOM 모델 실습 – CSS 스타일 변경하기

- 자바스크립트로 CSS 스타일 변경하기

DOM 모델 – CSS 스타일 변경하기

```
<section id = "section">
  <div>
    <input class="src-input" list="img-list">
    <datalist id = "img-list">
      <option value="img1.jpg">img1</option>
      <option value="img2.jpg">img2</option>
      <option value="img3.jpg">img3</option>
    </datalist>
    <select class="img-select">
      <option value="img1.jpg">img1</option>
      <option value="img2.jpg">img2</option>
      <option value="img3.jpg">img3</option>
    </select>
    <input class="color-input" type="color">
    <input class="change-button" type="button" value="변경하기">
  </div>
  <div>
    
  </div>
</section>
```

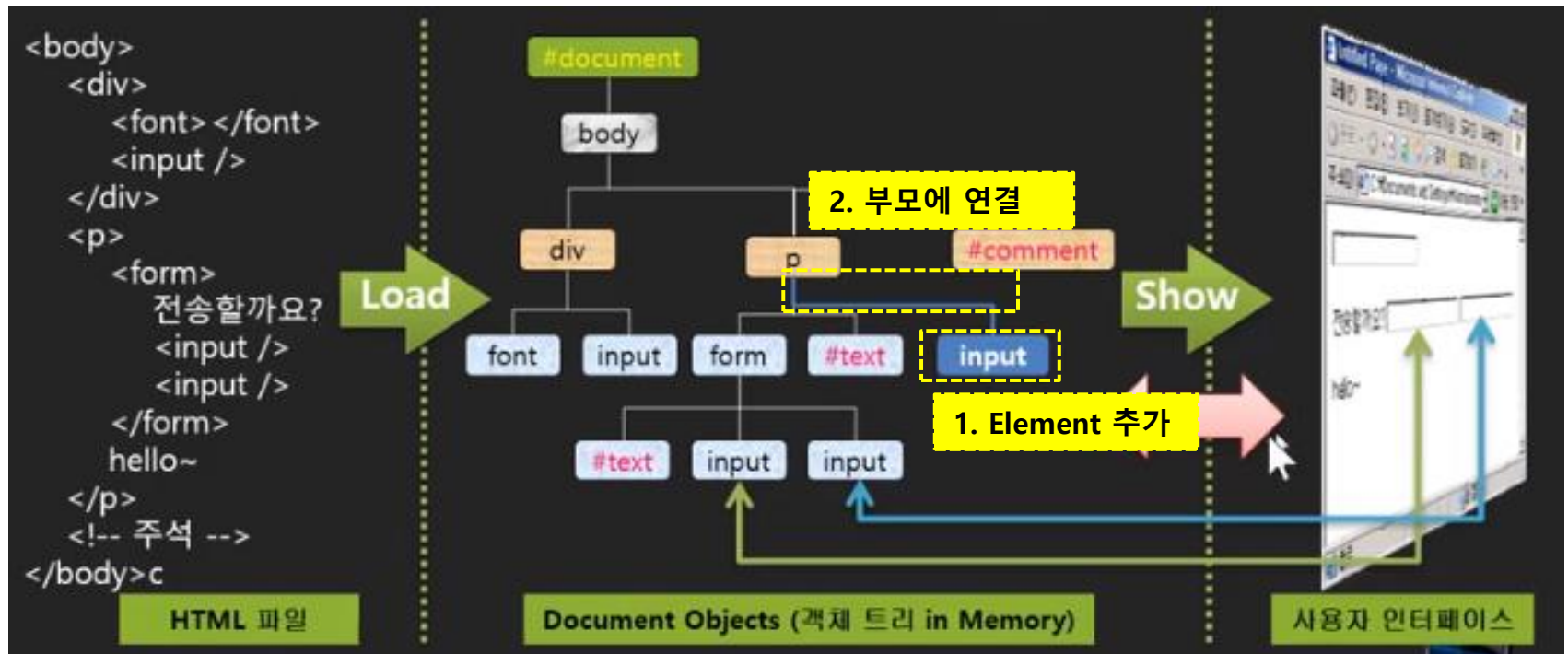
해당 스타일 변경함.

```
window.addEventListener ("load", function() {
  var section = document.querySelector("#section");
  var srcInput = section.querySelector(".src-input");
  var imgSelect = section.querySelector(".img-select");
  var changeButton = section.querySelector(".change-button");
  var img = section.querySelector(".img");
  var colorInput = section.querySelector(".color-input");
  changeButton.onclick = function() {
    console.log(colorInput.value);
    img.src = "images/" + srcInput.value;
    // img.style.color-border = ?; // color-border는 안됨.
    // 아래는 둘 중 하나 사용 가능함.
    img.style['border-color'] = colorInput.value;
    img.style.borderColor = colorInput.value;
    // 이미지의 주의사항. class 확인할 때는 class가 아닌 className
    console.log(img.className);
  };
});
```

3.9.3. DOM 모델 실습 – Element 추가하기

- 자바스크립트로 TEXT 추가하기
- 단계 : Element 추가 ➔ 부모노드에 연결하기

DOM 모델 – TEXT 추가하기



3.9.3. DOM 모델 실습 – TEXT 추가하기 (1)

- 자바스크립트로 TEXT 추가하기
- 단계 : Element 추가 ➔ 부모노드에 연결하기

DOM 모델 – TEXT 추가하기

```
<!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-scal
  <title>Document</title>
  <script src="node_03_01.js"></script>
</head>
<body>
  <section id = "section">
    <div>
      <input class="title-input" name="title">
      <input class="add-button" type="button" value="추가">
      <input class="del-button" type="button" value="삭제">
    </div>
    <div class="menu-list">
    </div>
  </section>
</body>
</html>
```

1. ADD 버튼 클릭시 Menu-List에 추가
2. DEL 버튼 클릭시 Menu-List에서 한개씩 삭제

```
window.addEventListener ("load", function() {
  var section = document.querySelector ("#section");
  var titleInput = section.querySelector (".title-input");
  var addButton = section.querySelector (".add-button");
  var delButton = section.querySelector (".del-button");
  var menuListDiv = section.querySelector (".menu-list");

  addButton.onclick = function() {
    // 사용자가 입력한 값에 대해 노드 생성
    var textNode = document.createTextNode(titleInput.value);
    // menuList에 추가함.
    menuListDiv.appendChild(textNode);
  };
  delButton.onclick = function() {
    // menuList에 여러개가 있을텐데, 그 중 한개 선택
    var textNode = menuListDiv.childNodes[0];
    // menuList에서 선택한 것을 지움.
    menuListDiv.removeChild(textNode);
    // 그런데 맨 앞에 화이트 스페이스 있음.
    // 그러나 문서 리스트를 만들때는 list Item을 통해서 처리하기 때문
  };
});
```

3.9.3. DOM 모델 실습 – TEXT 추가하기 (2)

- 자바스크립트로 TEXT 추가하기
- 단계 : Element 추가 ➔ 부모노드에 연결하기

DOM 모델 – TEXT 추가하기

```
<!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>
  <script src="node_03_02.js"></script>
</head>
<body>
  <section id = "section">
    <div>
      <input class="title-input" name="title">
      <input class="add-button" type="button" value="추가">
      <input class="del-button" type="button" value="삭제">
    </div>
    <div class="menu-list">
      <li>HTML</li>
      <li>CSS</li>
      <li>JAVA Script</li>
    </div>
  </section>
</body>
</html>
```

Data 관리는 List Item으로
관리하는 것이 바람직함

```
window.addEventListener("load", function() {
  var section = document.querySelector("#section");
  var titleInput = section.querySelector(".title-input");
  var addButton = section.querySelector(".add-button");
  var delButton = section.querySelector(".del-button");
  var menuListDiv = section.querySelector(".menu-list");

  addButton.onclick = function() {
    // 사용자가 입력한 값에 대해 노드 생성
    var textNode = document.createTextNode(titleInput.value);
    var liNode = document.createElement("li");

    // 첫번째 방법. 생성된 NODE를 부모에 추가함
    liNode.appendChild(textNode);
    menuListDiv.appendChild(liNode);

    // 두번째 방법. 생성된 부모노드에 추가하는 방법 (innerHTML)
    // menuListDiv.innerHTML =
    // menuListDiv.innerHTML + "<li>"+titleInput.value+"</li>";

  };

  delButton.onclick = function() {
    // 선택된 TEXT의 LI 중 가장 앞에 있는 것 선택
    // children은 TAG 기준의 NODE만을 선택함
    var liNode = menuListDiv.children[0];

    // menuList에서 선택한 것을 지움.
    menuListDiv.removeChild(liNode);
  };
});
```

3.9.4. DOM 모델 실습 – NODE 복제 (Table 예제) - 1

- 자바스크립트로 Table의 데이터 복제

DOM 모델 – NODE 복제

```
<section id = "section">
  <div>
    <input class="clone-button" type="button" value="복사">
    <input class="input-button" type="button" value="입력">
    <input class="temp-button" type="button" value="템플릿">
  </div>
  <table border="1" class="notice-list">
    <thead>
      <tr>
        <td>번호</td>
        <td>제목</td>
        <td>내용</td>
        <td>작성자</td>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>1</td>
        <td>가입신청</td>
        <td>가입을 신청합니다.</td>
        <td>강*준</td>
      </tr>
    </tbody>
  </table>
</section>
```

```
window.addEventListener ("load", function() {
  var notices=[
    {id:5, title:"가입인사2", content:"안녕하세요2", writer:"김*성"},
    {id:6, title:"가입인사3", content:"안녕하세요3", writer:"한*국"},
  ];
  var section = document.querySelector("#section");
  var noticeList = document.querySelector(".notice-list");
  var cloneButton = section.querySelector(".clone-button");
  var inputButton = section.querySelector(".input-button");
  var tempButton = section.querySelector(".temp-button");

  cloneButton.onclick = function() {
    // table의 tbody tr이 clone 대상임
    // querySelector는 첫번째 한개만 가져옴.
    var trNode = noticeList.querySelector("tbody tr");
    // querySelectorALL 할 경우는 배열로 처리해야함
    // var cloneNode = noticeList.querySelectorAll("tbody tr")[0];
    // true: 하위 전체 카피, false: 최상위 노드만 카피
    var cloneNode = trNode.cloneNode(true);
    // console.log(trNode);
    // console.log(cloneNode);
    var tbodyNode = noticeList.querySelector("tbody");
    tbodyNode.appendChild(cloneNode);
  };
});
```

3.9.4. DOM 모델 실습 – NODE 삽입 (Table 예제) -2

- 자바스크립트로 Table의 데이터 삽입

DOM 모델 – NODE 삽입

```
<section id = "section">
  <div>
    <input class="clone-button" type="button" value="복사">
    <input class="input-button" type="button" value="입력">
    <input class="temp-button" type="button" value="템플릿">
  </div>
  <table border="1" class="notice-list">
    <thead>
      <tr>
        <td>번호</td>
        <td>제목</td>
        <td>내용</td>
        <td>작성자</td>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>1</td>
        <td>가입신청</td>
        <td>가입을 신청합니다.</td>
        <td>강*준</td>
      </tr>
    </tbody>
  </table>
</section>
```

```
window.addEventListener("load", function() {
  var notices=[
    {id:5, title:"가입인사2", content:"안녕하세요2", writer:"김*성"},
    {id:6, title:"가입인사3", content:"안녕하세요3", writer:"한*국"},
  ];
  var section = document.querySelector("#section");
  var noticeList = document.querySelector(".notice-list");
  var cloneButton = section.querySelector(".clone-button");
  var inputButton = section.querySelector(".input-button");
  var tempButton = section.querySelector(".temp-button");

  inputButton.onclick = function() {
    var tbodyNode = noticeList.querySelector("tbody");
    var trNode = noticeList.querySelector("tbody tr");
    var cloneNode = trNode.cloneNode(true);
    var tds = cloneNode.querySelectorAll("td");
    // 속해있는 td에 대해서 배열로 처리됨
    console.log(tds);
    tds[0].textContent = notices[0].id;
    tds[1].textContent = notices[0].title;
    tds[2].textContent = notices[0].content;
    tds[3].textContent = notices[0].writer;

    tbodyNode.appendChild(cloneNode);
  };
});
```

1. 복제
2. 업데이트

3.9.4. DOM 모델 실습 – NODE 삽입 (Table 예제) -3

- 자바스크립트로 Table의 데이터 템플릿 기준 삽입
- 초기 데이터 없을 경우 활용

DOM 모델 – NODE 삽입 (템플릿)

```
<div>
  <input class="clone-button" type="button" value="복사">
  <input class="input-button" type="button" value="입력">
  <input class="temp-button" type="button" value="템플릿">
</div>
<!-- 템플릿은 출력안됨. -->
<template>
  <tr>
    <td></td>
    <td></td>
    <td></td>
    <td></td>
  </tr>
</template>
<table border="1" class="notice-list">
  <thead>
    <tr>
      <td>번호</td>
      <td>제목</td>
      <td>내용</td>
      <td>작성자</td>
    </tr>
  </thead>
  <tbody>
    <!-- <tr>
      <td>1</td>
      <td>가입신청</td>
      <td>가입을 신청합니다.</td>
      <td>김*준</td>
    </tr> -->
  </tbody>
</table>
```

2. 템플릿 만든 후.
3. 해당데이터를 복제하여
4. 업데이트

1. 초기 데이터 없음

```
window.addEventListener("load", function() {
  var notices=[
    {id:5, title:"가입인사2", content:"안녕하세요2", writer:"김*성"},
    {id:6, title:"가입인사3", content:"안녕하세요3", writer:"한*국"},
  ];
  var section = document.querySelector("#section");
  var noticeList = document.querySelector(".notice-list");
  var cloneButton = section.querySelector(".clone-button");
  var inputButton = section.querySelector(".input-button");
  var tempButton = section.querySelector(".temp-button");

  tempButton.onclick = function() {
    var tempNode = section.querySelector("template");
    console.log(tempNode);

    // 복제할 노드는 tempNode에서 import 함.
    var cloneNode = document.importNode(tempNode.content, true);
    var tds = cloneNode.querySelectorAll("td");
    // 속해있는 td에 대해서 배열로 처리됨
    console.log(tds);
    tds[0].textContent = notices[0].id;
    tds[1].textContent = notices[0].title;
    tds[2].textContent = notices[0].content;
    tds[3].textContent = notices[0].writer;

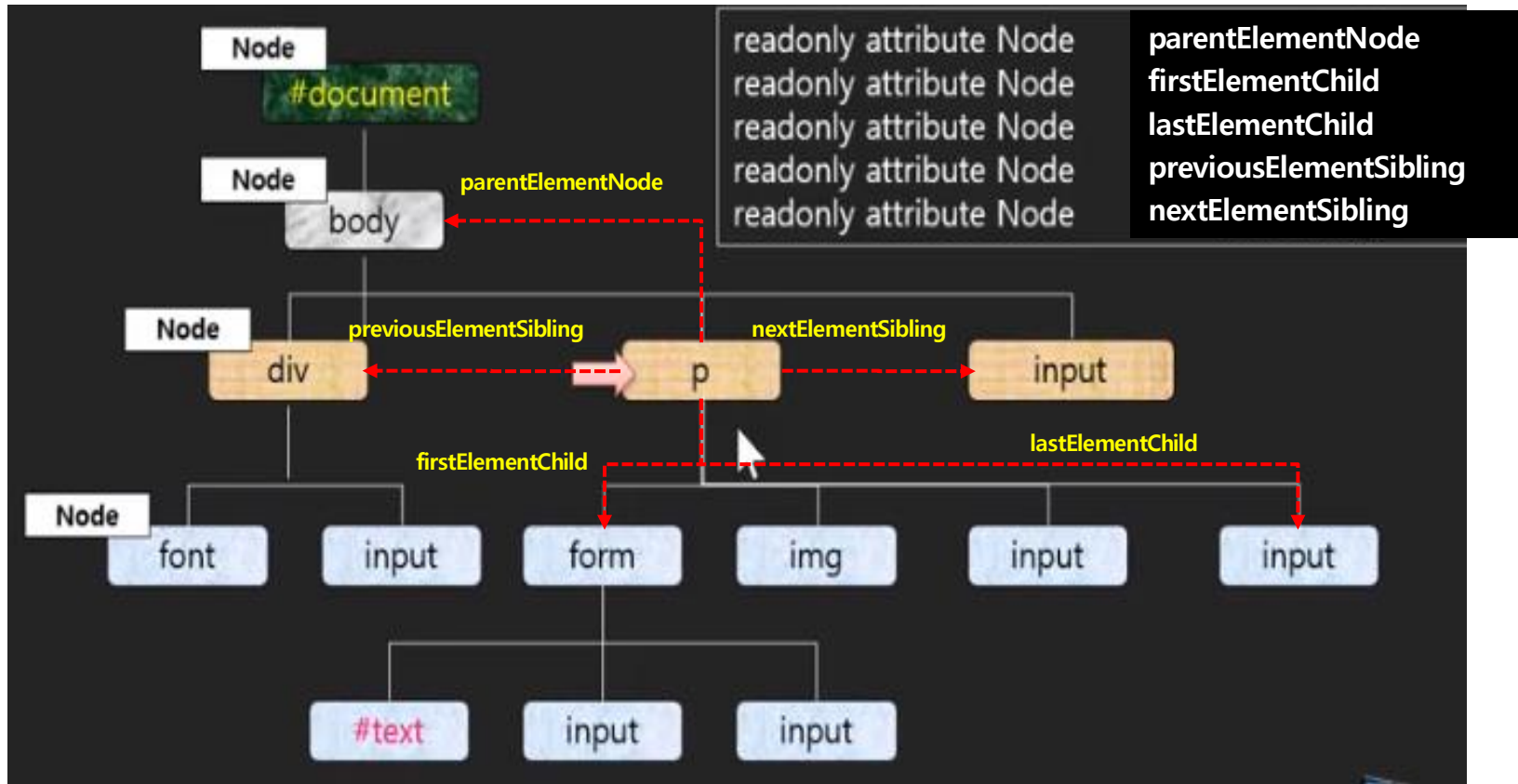
    var tbodyNode = noticeList.querySelector("tbody");
    tbodyNode.appendChild(cloneNode);
  };
});
```

1. 템플릿 복제
2. 실제데이터 업데이트

3.9.5. DOM 모델 실습 – Node 관리 (삽입, 순회, 제거 등)

- Node는 메서드를 통해서 관리할 수 있음
- Node 관리 메서드 : `parentNode`, `firstElementChild`, `lastElementChild`
`previousElementSibling`, `nextElementSibling`

DOM 모델 – NODE 삽입 (템플릿)



3.9.5. DOM 모델 실습 – Node 관리 (삽입, 순회, 제거 등)

- Node는 메서드를 통해서 관리할 수 있음
- Node 관리 메서드 : parentNode, firstElementChild, lastElementChild
previousElementSibling, nextElementSibling

DOM 모델 – NODE 순회

```
<body>
  <section id = "section">
    <div>
      <input class="up-button" type="button" value="위로">
      <input class="down-button" type="button" value="아래로">
    </div>
    <table border="1" class="notice-list">
      <thead>
        <tr>
          <th>번호</th>
          <th>제목</th>
          <th>내용</th>
          <th>작성자</th>
        </tr>
      </thead>
      <tbody>
        <tr style="background-color: lightblue">
          <td>1</td>
          <td>가입신청</td>
          <td>가입을 신청합니다.</td>
          <td>강*준</td>
        </tr>
        <tr>
          <td>2</td>
          <td>가입신청2</td>
          <td>가입을 신청2.</td>
          <td>성*혁</td>
        </tr>
      </tbody>
    </table>
  </section>
</body>
```

번호	제목	내용	작성자
1	가입신청	가입을 신청합니다.	강*준
2	가입신청2	가입을 신청2.	성*혁
3	가입신청3	가입을 신청3.	한*연

```
window.addEventListener("load", function() {
  var section = document.querySelector("#section");
  var noticeList = section.querySelector(".notice-list");
  var tbodyNode = noticeList.querySelector("tbody");
  var upButton = section.querySelector(".up-button");
  var downButton = section.querySelector(".down-button");
```

우선 첫번째인 TR을 선택. 아래도 가능함.

```
var currentNode = tbodyNode.children[0];
var currentNode = tbodyNode.firstElementChild;
```

```
upButton.onclick = function() {
  var nextNode = currentNode.nextElementSibling;
  if (nextNode == null) {
    alert("더이상 이동할 수 없습니다.");
    return;
  }
  // next를 지우고, insertBefore (대상, 기준)
  // next를 지우고, next 앞에 current를 추가함
  tbodyNode.insertBefore(nextNode, currentNode);
};
```

```
downButton.onclick = function() {
  var prevNode = currentNode.previousElementSibling;
  if (prevNode == null) {
    alert("더이상 이동할 수 없습니다.");
    return;
  }
  // current를 지우고, insertBefore (대상, 기준)
  // next를 지우고, current 앞에 previous를 추가함
  tbodyNode.insertBefore(currentNode, prevNode);
};
```

3.9.6. DOM 모델 실습 – checkBox (1)

- checkBox도 자바스크립트를 통해서 관리 가능
- 모든 버튼을 선택하는 방법

DOM 모델 – checkBox

```
<section id = "section">
  <div>
    <input class="del-button" type="button" value="일괄삭제">
    <input class="swap-button" type="button" value="선택노드 바꾸기">
  </div>
  <table border="1" class="notice-list">
    <thead>
      <tr>
        <td><input type="checkbox" class="overall-checkbox"></td>
        <td>번호</td>
        <td>제목</td>
        <td>내용</td>
        <td>작성자</td>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td><input type="checkbox"></td>
        <td>1</td>
        <td>가입신청</td>
        <td>가입을 신청합니다.</td>
        <td>강*준</td>
      </tr>
      <tr>
        <td><input type="checkbox"></td>
        <td>2</td>
        <td>가입신청2</td>
        <td>가입을 신청2.</td>
        <td>성*혁</td>
      </tr>
    </tbody>
  </table>
</section>
```

<input type="checkbox"/>	번호	제목	내용	작성자
<input type="checkbox"/>	1	가입신청	가입을 신청합니다.	강*준
<input type="checkbox"/>	2	가입신청2	가입을 신청2.	성*혁
<input type="checkbox"/>	3	가입신청3	가입을 신청3.	한*연

```
indow.addEventListener ("load", function() {
  var section = document.querySelector ("#section");
  var noticeList = section.querySelector (".notice-list");
  var tbodyNode = noticeList.querySelector ("tbody");
  var allCheckBox = section.querySelector (".overall-checkbox");
  var delButton = section.querySelector (".del-button");
  var swapButton = section.querySelector (".swap-button");

  allCheckBox.onchange = function() {
    var inputs=tbodyNode.querySelectorAll("input[type='checkbox']");
    console.log(inputs);
    for (var i=0 ; i<inputs.length ; i++) {
      // all체크박스와 동일하게 모든 input 박스를 만들.
      inputs[i].checked = allCheckBox.checked;
    }
  };
});
```

3.9.6. DOM 모델 실습 – checkBox (2)

- checkBox도 자바스크립트를 통해서 관리 가능
- 선택된 항목 일괄 삭제

DOM 모델 – checkBox

```
<section id = "section">
  <div>
    <input class="del-button" type="button" value="일괄삭제">
    <input class="swap-button" type="button" value="선택노드 바꾸기">
  </div>
  <table border="1" class="notice-list">
    <thead>
      <tr>
        <td><input type="checkbox" class="overall-checkbox"></td>
        <td>번호</td>
        <td>제목</td>
        <td>내용</td>
        <td>작성자</td>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td><input type="checkbox"></td>
        <td>1</td>
        <td>가입신청</td>
        <td>가입을 신청합니다.</td>
        <td>강*준</td>
      </tr>
      <tr>
        <td><input type="checkbox"></td>
        <td>2</td>
        <td>가입신청2</td>
        <td>가입을 신청2.</td>
        <td>성*혁</td>
      </tr>
    </tbody>
  </table>
</section>
```

일괄삭제	선택노드 바꾸기			
<input type="checkbox"/>	번호	제목	내용	작성자
<input type="checkbox"/>	1	가입신청	가입을 신청합니다.	강*준
<input type="checkbox"/>	2	가입신청2	가입을 신청2.	성*혁
<input type="checkbox"/>	3	가입신청3	가입을 신청3.	한*연

```
window.addEventListener ("load", function() {
  var section = document.querySelector ("#section");
  var noticeList = section.querySelector (".notice-list");
  var tbodyNode = noticeList.querySelector ("tbody");
  var allCheckBox = section.querySelector (".overall-checkbox");
  var delButton = section.querySelector (".del-button");
  var swapButton = section.querySelector (".swap-button");

  delButton.onclick = function() {
    // input type이 checkbox이면서 체크된 것만 배열로 가져옴
    var inputs=tbodyNode.querySelectorAll("input[type='checkbox']:checked");
    console.log(inputs);

    for (var i=0 ; inputs.length ; i++){
      inputs[i].parentElement.parentElement.remove();
    }
  } ;
});
```

3.9.6. DOM 모델 실습 – checkBox (3)

- checkBox도 자바스크립트를 통해서 관리 가능
- 선택항목 SWAP

DOM 모델 – checkBox

```
section id = "section">
  <div>
    <input class="del-button" type="button" value="일괄삭제">
    <input class="swap-button" type="button" value="선택노드 바꾸기">
  </div>
  <table border="1" class="notice-list">
    <thead>
      <tr>
        <td><input type="checkbox" class="overall">
        <td>번호</td>
        <td>제목</td>
        <td>내용</td>
        <td>작성자</td>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td><input type="checkbox"></td>
        <td>1</td>
        <td>가입신청</td>
        <td>가입을 신청합니다.</td>
        <td>강*준</td>
      </tr>
      <tr>
        <td><input type="checkbox"></td>
        <td>2</td>
        <td>가입신청2</td>
        <td>가입을 신청2.</td>
        <td>성*혁</td>
      </tr>
    </tbody>
  </table>
```

```
window.addEventListener("load", function() {
  var section = document.querySelector("#section");
  var noticeList = section.querySelector(".notice-list");
  var tbodyNode = noticeList.querySelector("tbody");
  var allCheckBox = section.querySelector(".overall-checkbox");
  var delButton = section.querySelector(".del-button");
  var swapButton = section.querySelector(".swap-button");

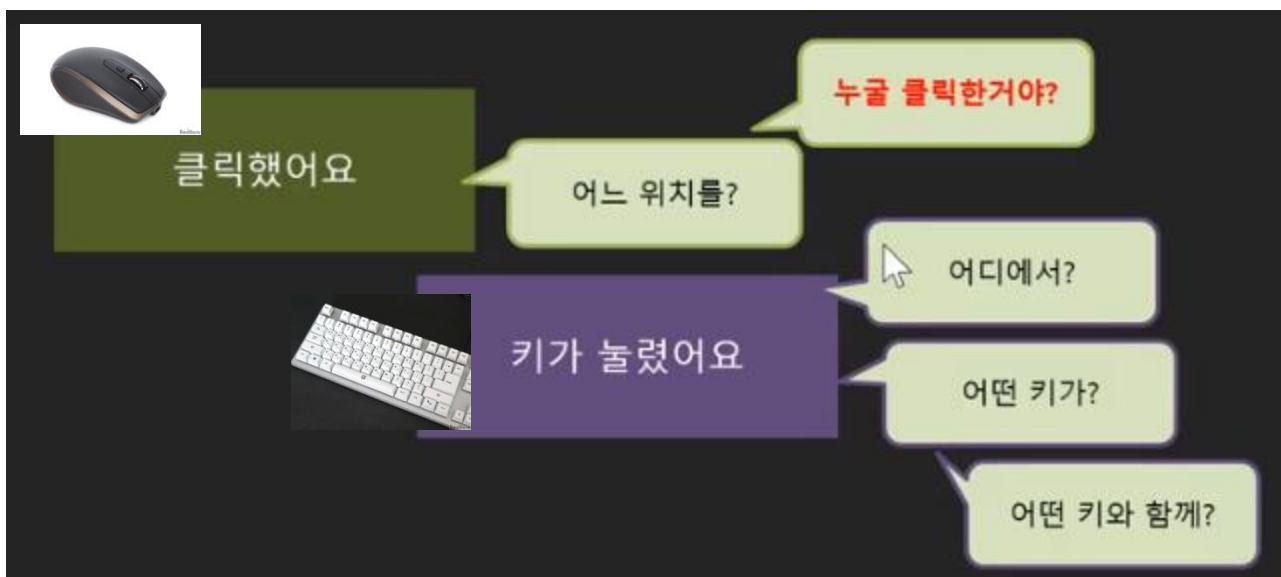
  swapButton.onclick = function() {
    var inputs = tbodyNode.querySelectorAll("input[type='checkbox']:checked");
    if(inputs.length != 2) {
      alert("2개만 체크해주세요");
      return;
    }
    var trs = [];
    for (var i=0; i<inputs.length; i++)
      trs.push(inputs[i].parentElement.parentElement);
    // 자손까지 Copy하여 복제해놓음
    var cloneNode = trs[0].cloneNode(true);
    // copy 문과 1과 위치 교환
    trs[1].replaceWith(cloneNode);
    // 1과 2의 위치 교환
    trs[0].replaceWith(trs[1]);
  };
});
```

순서가 이해가 안됨

3.10.1. 이벤트 객체

- Keyboard 및 Mouse 등의 입력장치에서는 이벤트가 발생함
- 발생한 Event의 속성값들을 대상으로 문서의 변경을 활용할 수 있음

이벤트 객체 종류 및 객체별 속성



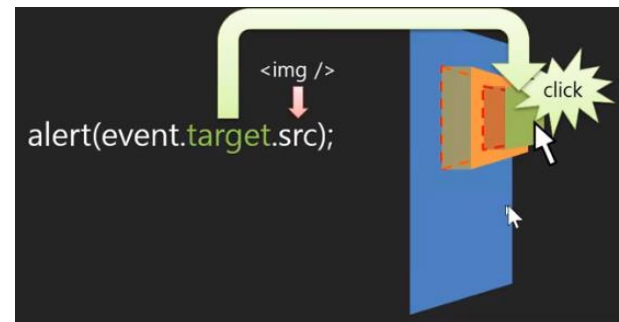
선택된 객체의 속성들을 사용할 수 있음

0. 마우스 이벤트의 종류

1. click, mousedown, mouseup
2. dblclick
3. mousemove
4. mouseover, mouseout
5. mouseenter, mouseleave
6. mouseover, mouseout와 mouseenter, mouseleave 차이점
7. contextmenu

- 키보드 이벤트의 종류 -

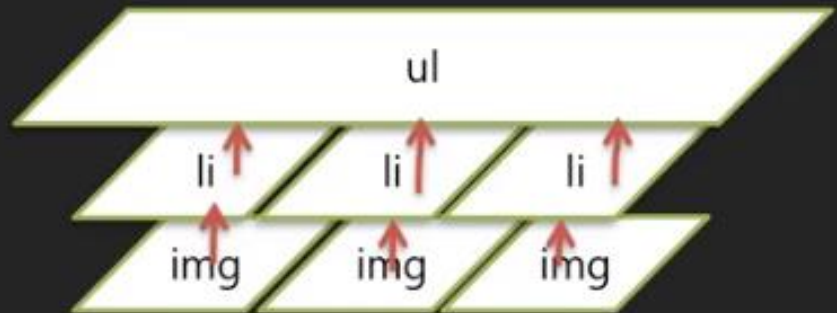
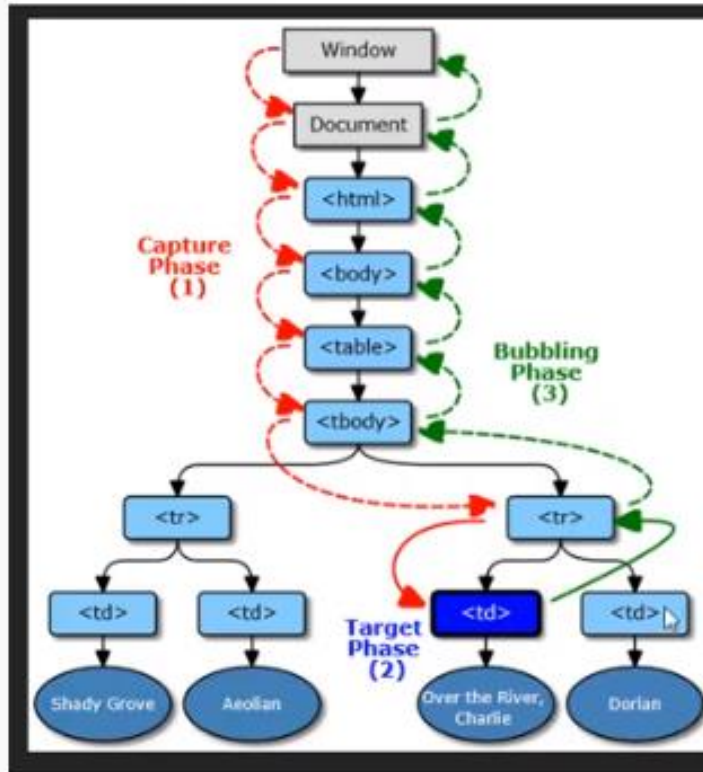
- keydown: 키를 누를 때 발생
- keypress: 키를 누를 때 발생 (keyup: 키를 뗄 때 발생)



3.10.2. 이벤트 관리

- 이벤트는 Window라는 객체가 모든 객체를 대상으로 관리하고 있음.
- 최하단의 객체의 이벤트는, 상위로 이벤트를 발생시키고 있음. (상위 객체에서 관리 가능함)

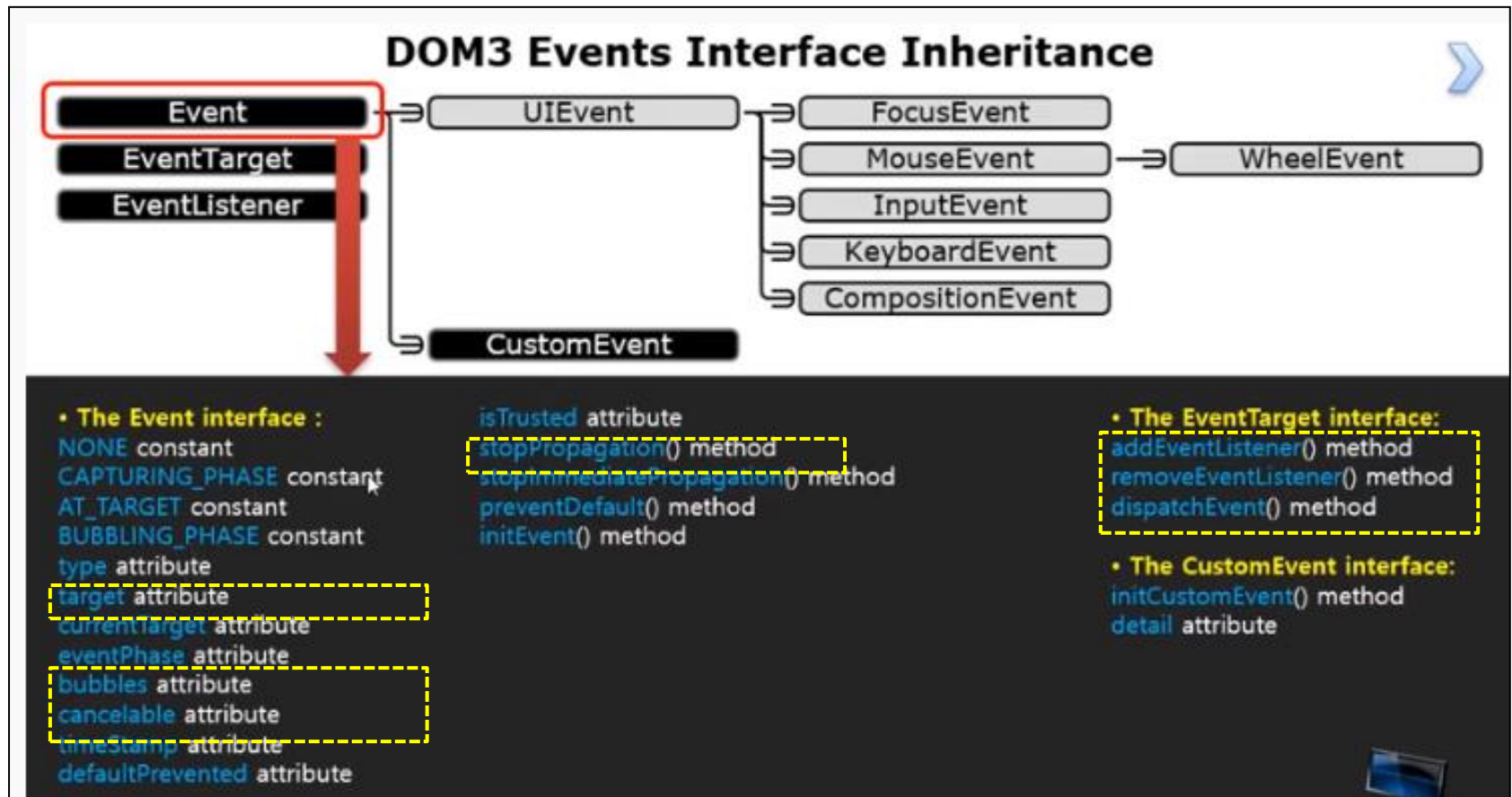
버블링 개념 정리



3.10.3. 이벤트 객체의 메서드 (1)

- EVENT 인터페이스가 최상단이며, UI > 키보드, 마우스(휠 포함) Event 인터페이스가 있음

Event 객체의 메서드

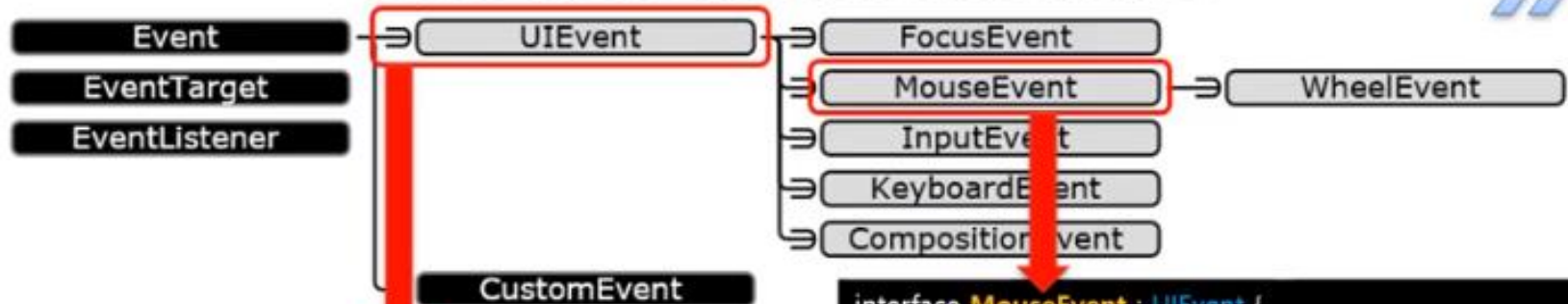


3.10.3. 이벤트 객체의 메서드 (2)

- EVENT 인터페이스가 최상단이며, UI > 키보드, 마우스(휠 포함) Event 인터페이스가 있음
- Mouse는 X,Y의 좌표를 확인할 수 있도록 인터페이스 제공함

Mouse객체의 메서드

DOM3 Events Interface Inheritance



```
interface UIEvent : Event {  
    readonly attribute WindowProxy? view;  
    readonly attribute long detail;  
};
```

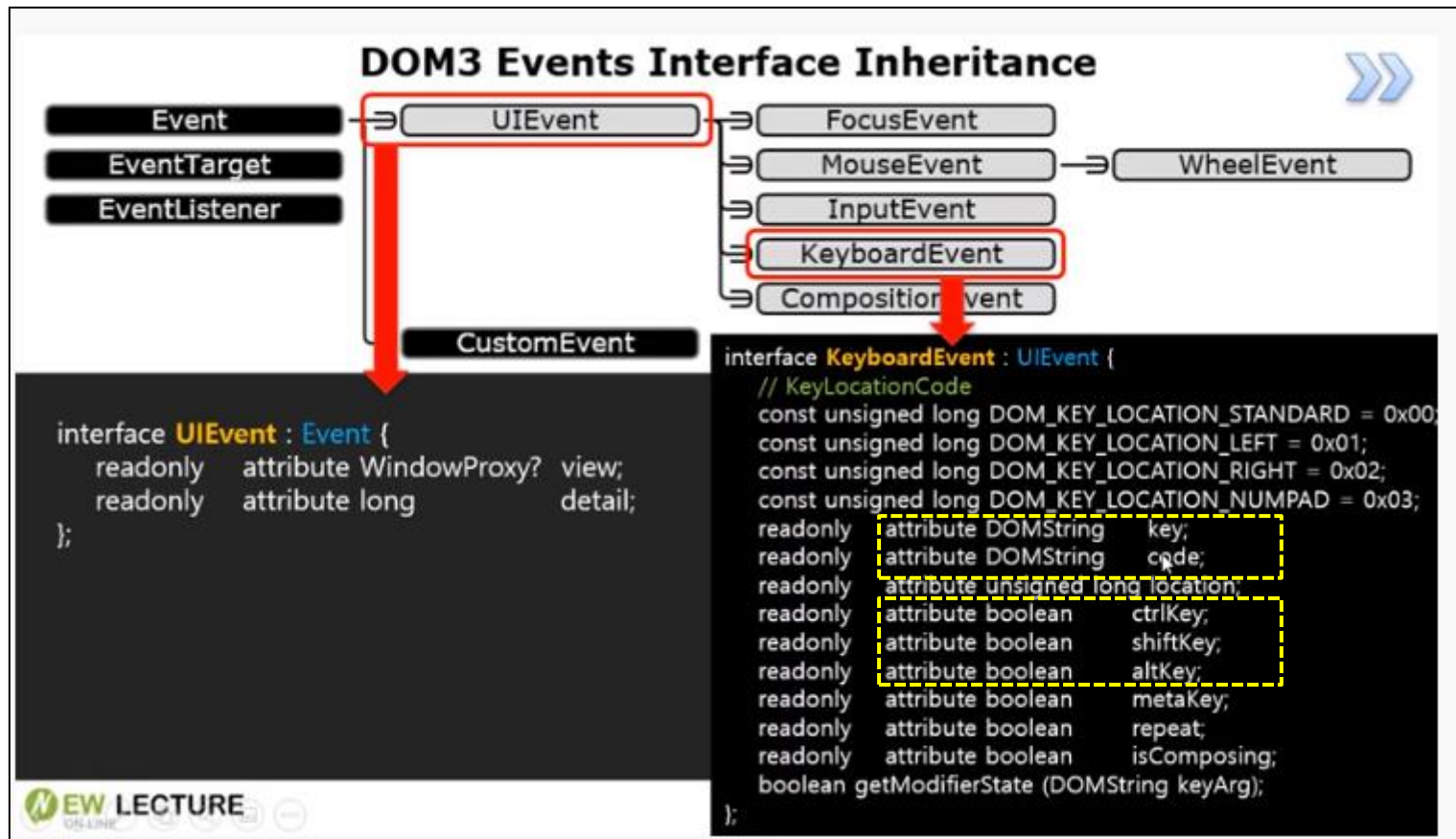
detail of type long, readonly
Specifies some detail information about the Event, depending on the type of event.
The un-initialized value of this attribute MUST be 0.
view of type WindowProxy, readonly, nullable
The view attribute identifies the Window from which the event was generated.
The un-initialized value of this attribute MUST be null.

```
interface MouseEvent : UIEvent {  
    readonly attribute long screenX;  
    readonly attribute long screenY;  
    readonly attribute long clientX;  
    readonly attribute long clientY;  
    readonly attribute boolean ctrlKey;  
    readonly attribute boolean shiftKey;  
    readonly attribute boolean altKey;  
    readonly attribute boolean metaKey;  
    readonly attribute short button;  
    readonly attribute EventTarget? relatedTarget;  
    // Introduced in DOM Level 3  
    readonly attribute unsigned short buttons;  
    boolean getModifierState (DOMString keyArg);  
};
```

3.10.3. 이벤트 객체의 메서드 (3)

- EVENT 인터페이스가 최상단이며, UI > 키보드, 마우스(휠 포함) Event 인터페이스가 있음
- Keyboard는 눌러진 Key 및 ALT/SHIFT 등을 확인할 수 있도록 인터페이스 제공함

KeyBoard 객체의 메서드



3.10.4. 마우스 이벤트 객체 실습 - 1

- 마우스 이벤트는 e.target이란 객체로 속성값들을 사용할 수 있음



마우스 이벤트 객체 실습

```
<!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">
  <title>Document</title>
  <script src="node_05_01.js"></script>
</head>
<body>
  <section id = "section">
    <div>
      
      
      
    </div>
    <div>
      
    </div>
  </section>
</body>
</html>
```

```
window.addEventListener("load", function() {
  var section = document.querySelector("#section");
  var imgs = section.querySelectorAll(".img");
  var currentImg = section.querySelector(".current-img");

  // imgs[0].onclick = function(e) {
  //   console.log(e.target.src);
  //   currentImg.src = e.target.src;
  // };
  // imgs[1].onclick = function(e) {
  //   console.log(e.target);
  //   currentImg.src = e.target.src;
  // };
  // imgs[2].onclick = function(e) {
  //   console.log(e.target);
  //   currentImg.src = e.target.src;
  // };
  // 위의 내용을 단순화 함.
  for (var i=0 ; imgs.length; i++) {
    imgs[i].onclick = function(e) {
      // console.log(e.target);
      currentImg.src = e.target.src;
    };
  }
});
```

e.target이 image임
Image 관련 객체 사용가능

3.10.4. 마우스 이벤트 객체 실습 - 2

- 마우스 이벤트는 e.target이란 객체로 속성값들을 사용할 수 있음

마우스 이벤트 객체 실습

```
<!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">
  <title>Document</title>
  <script src="node_05_02.js"></script>
</head>
<body>
  <section id = "section">
    <div class="img-list">
      
      
      
    </div>
    <div>
      
    </div>
  </section>
</body>
</html>
```

```
window.addEventListener ("load", function() {
  var section = document.querySelector ("#section");
  var imgList = section.querySelector(".img-list");
  var currentImg = section.querySelector(".current-img");

  // for (var i=0 ; imgs.length; i++) {
  //   imgs[i].onclick = function(e) {
  //     // console.log(e.target);
  //     currentImg.src = e.target.src;
  //   };
  // }

  imgList.onclick = function(e){
    // 이미지 영역밖에서도 있는 문제
    // 왜냐하면 DIV이기 때문에
    console.log(e.target.className);
    if (e.target.className != 'img') return;
    currentImg.src = e.target.src;
  }
});
```

해당되지 않는 객체도 반복
하면서 호출하는 것은
문제가 있음.

상위객체에서 관리하되,
img가 아닌 것은 제외함.
버블링 효과로 인해
이벤트 체크가 가능함.

3.10.5. 마우스 이벤트 객체 실습 - Trigger

- 특정 Element 이벤트는 다른 Element 이벤트를 호출할 수 있음 (트리거)
- Span tag의 클릭이 file 선택하는 input Tag를 클릭하게 함.

Trigger 실습

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, in
  <title>Document</title>
  <script src="node_06_01.js"></script>
</head>
<body>
  <section id = "section">
    <style>
      .file-button{
        display:none;
      }
      .file-rriget-button{
        background-color: green;
        border: 1px solid lightgreen;
        border-radius: 5px;
        padding: 5px 10px;
        color: white;
        cursor: pointer;
      }
      .file-rriget-button:hover{
        background-color: lightgreen;
      }
    </style>
    <input type="file" class="file-button">
    <span class="file-rriget-button">파일선택</span>
  </section>
</body>
```

Input Tag는 숨김
사용자 선택 불가함

Span Tag의 클릭 유도

```
window.addEventListener ("load", function() {
  var section = document.querySelector ("#section");
  var fileButton = section.querySelector(".file-button");
  var fileTriggerButton = section.querySelector(".file-rriget-button");

  fileButton.onclick = function(e){
  };
  // fileTriggerButton은 span tag로 파일을 선택하는 기능이 없음
  fileTriggerButton.onclick = function(e){
    // 마우스 이벤트 객체 받아옴
    // 크롬에서만 진행됨.
    var event = new MouseEvent("click",{
      'view':window,
      'bubbles':true, // 버블링이 가능한지?
      'cancelable':true // cancel 가능한지?
    });
    // fileTriggerButton을 통해서 fileButton을 trigger 함
    fileButton.dispatchEvent(event);
  };
});
```

Filebutton은 코드 없음

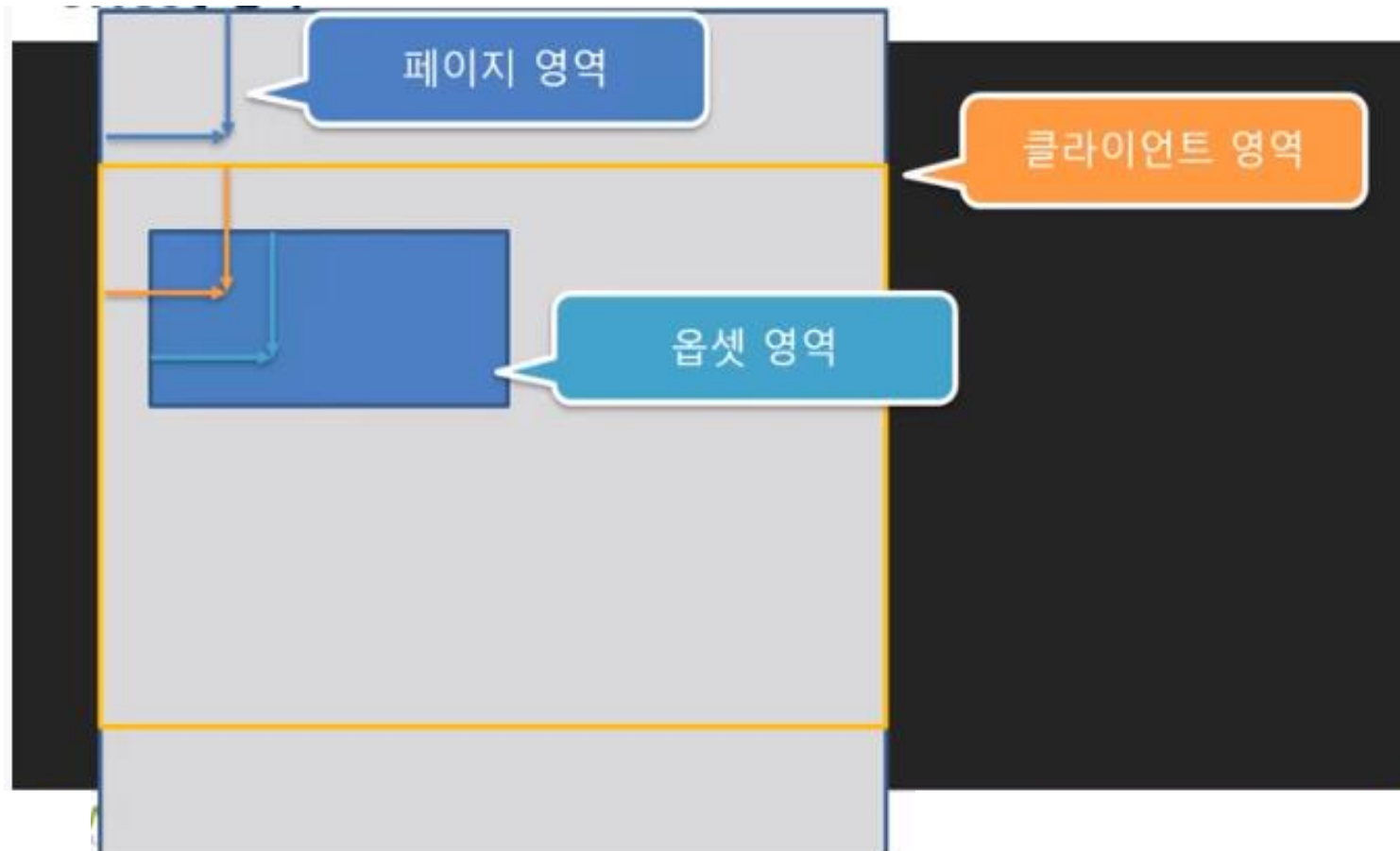
Trigger 발생

3.10.6. 마우스 이벤트 객체 실습 - X,Y 좌표 이용 (1)

- 마우스 이벤트 발생시 X,Y 좌표를 활용할 수 있음
- X,Y 좌표는 여러가지의 형태로 존재함.



마우스 X, Y 좌표 개념



3.10.6. 마우스 이벤트 객체 실습 - X,Y 좌표 이용 (2)

- 마우스 이벤트 발생시 X,Y 좌표를 활용할 수 있음
- X,Y 좌표는 여러가지의 형태로 존재함.



마우스 클릭에 반응하는 BOX

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="width=device-width, initial-scale=1">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Document</title>
  <script src="node_06_02.js"></script>
</head>
<body>
  <section id = "section">
    <style>
      .container{
        width:800px;
        height:400px;
        border: 1px solid gray;
      }
      .box{
        width:100px;
        height:100px;
        border: 1px solid blue;
        background-color: blue;
      }
    </style>
    <div class="container">
      <div class="box"></div>
    </div>
  </section>
</body>
</html>
```

```
✓ window.addEventListener ("load", function() {
  var section = document.querySelector ("#section");
  var container = section.querySelector(".container");
  var box = section.querySelector(".box");
  ✓ container.onclick = function(e){
    console.log("(x,y) = " + e.x, e.y);
    console.log("client = " + e.clientX, e.clientY);
    console.log("page = " + e.pageX, e.pageY);
    console.log("offset = " + e.offsetX, e.offsetY);
    box.style.position = "absolute";
    box.style.left = e.x + "px";
    box.style.top = e.y + "px";
  };
});
```


3.10.6. 마우스 이벤트 객체 실습 - X,Y 좌표 이용 (3)

- 마우스 이벤트 발생시 X,Y 좌표를 활용할 수 있음
- X,Y 좌표는 여러가지의 형태로 존재함.

마우스 DRAG

```
<section id = "section">
  <style>
    .container{
      width:800px;
      height:400px;
      border: 1px solid gray;
    }
    .box{
      width:100px;
      height:100px;
      border: 1px solid blue;
      background-color: blue;
      position: absolute;
    }
  </style>
  <div class="container">
    <div class="box"></div>
  </div>
</section>
```

```
window.addEventListener ("load", function() {
  var section = document.querySelector ("#section");
  var container = section.querySelector(".container");
  var box = section.querySelector(".box");
  var dragging = false;
  var offset = {x:0, y:0};

  container.onmousedown = function(e){
    // 선택된 것이 박스일때만 drag 가능
    if(e.target == box)
      dragging = true;
  };
  container.onmousemove = function(e){
    if (dragging) {
      // box.style.left = e.x + "px";
      // box.style.top = e.y + "px";
      //마우스 클릭 위치 지정됨.
      box.style.left = e.pageX - offset.x + "px";
      box.style.top = e.pageY - offset.y + "px";
    }
  };
  container.onmouseup = function(e){
    dragging = false;
  };
  box.onmousedown = function(e) {
    offset.x = e.offsetX;
    offset.y = e.offsetY;
  }
});
```

Offset 위치 참조