

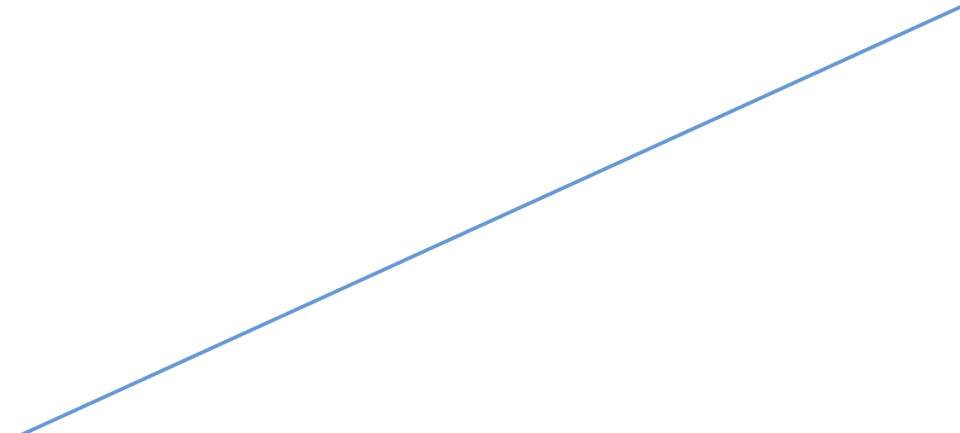
JavaScript

Front-End 개발자 양성 과정



DinoWorks
DIGITAL NOMAD

JavaScript

- JavaScript Core
 - JavaScript BOM
 - JavaScript DOM
- 

➤ 웹의 3대 구성 요소

- HTML : 웹 페이지의 콘텐츠를 정의
- CSS : 웹 페이지의 레이아웃(디자인) 구성
- JavaScript : 웹 페이지의 동작을 프로그래밍

➤ JavaScript로 할 수 있는 일

- HTML 콘텐츠를 수정
- HTML 속성(Attribute)을 수정
- HTML의 Style(CSS)을 수정
- 자료의 유효성 검사(Validation) 등

```
document.getElementById("demo").innerHTML = "Hello JavaScript";  
var image = document.getElementById('myImage');  
image.src = "pic_bulboff.gif";  
document.getElementById("demo").style.fontSize = "25px";
```

- HTML 문서의 <body>, <head> 내부에 작성
- HTML 문서에서 작성하는 경우에는 <script></script> 태그 사이에 작성

```
<script>  
document.getElementById("demo").innerHTML = "My First JavaScript";  
</script>
```

- 시작 태그를 <script type="text/javascript">와 같은 형태로 쓰기도 함
 - HTML 문서에서 기본 스크립팅 언어가 JavaScript이므로 type 속성은 생략 가능
- JavaScript는 <head>, <body> 태그 내에 아무 곳에서도나 사용 가능
 - 가능하면 한 곳에 모아서 작성하는 것이 좋음.
- HTML 요소(element)내 이벤트 속성 값으로 사용

```
<button onclick="document.write(5 + 6)">Try it</button>
```

➤ JavaScript in <head>

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>
<h1>My Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
</body>
</html>
```

➤ JavaScript in <body>

```
<!DOCTYPE html>
<html>
<body>
<h1>My Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</body>
</html>
```

➤ 외부 JavaScript

```
<!DOCTYPE html>
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

- 외부 JavaScript 파일도 <head>, <body> 태그 어디서나 링크할 수 있으며, 링크한 위치에 해당 스크립트 파일이 위치하게 됨

➤ 외부 JavaScript 파일의 장점

- HTML 코드와 분리
- 파일의 가독성을 높이고 HTML과 JavaScript 파일의 유지보수가 용이
- JavaScript 파일이 캐쉬되며 페이지의 로딩속도를 높일 수 있음

➤ JavaScript의 출력 기능

- alert box 출력 : `window.alert()`
- HTML 문서 출력 : `document.write()`
- HTML 요소(element) 내부 출력 : `innerHTML()`
- 브라우저 콘솔(console) 출력 : `console.log()`

```
<script>
```

```
window.alert(5 + 6);
```

```
document.write(5 + 6);
```

```
document.getElementById("demo").innerHTML = 5 + 6;
```

```
console.log(5 + 6);
```

```
</script>
```

```
<button onclick="document.write(5 + 6)">Try it</button>
```


➤ 문장 (Statements)

- JavaScript는 문장으로 구성되며, 각 문장은 세미콜론(;)으로 구분
- 문장은 값 (Values), 연산자 (Operators), 표현식 (Expressions), 예약어 (Keyword), 주석 (Comments)으로 구성

➤ 값 (Values)

- 리터럴 (Fixed value), 변수 (Variable values)
- 리터럴 (Literals) : 숫자 (Numbers), 문자열 (Strings)
 - 숫자는 소수점을 포함하거나 포함하지 않는 형태로, 문자열은 외따옴표나 쌍따옴표 내부에 작성
- 변수 (Variables) : 데이터 값을 저장하기 위해 사용하며 var 키워드 사용

➤ 연산자 (Operators)

- 대입 연산자 (=), 사칙연산 (+, -, *, /) 등

➤ 표현식 (Expressions)

- 표현식은 값을 계산하기 위한 값, 변수, 연산자의 조합

```
5 * 10
```

```
x * 10
```

```
"John" + " " + "Doe"
```

➤ 예약어 (Keywords)

- 수행되어야 할 동작을 식별하기 위한 예약어

➤ 주석 (Comments)

- // (한줄 주석), /* */ (여러 줄 주석)

➤ 식별자 (Identifiers)

- 변수나 함수 등의 이름을 지정하기 위해 사용
- 첫 글자는 문자, _(underscore), \$ 만 허용되며 나머지는 문자, 숫자, _, \$ 사용
 - 첫 글자에는 숫자를 사용할 수 없음 (식별자와 숫자를 구분하기 위함)

➤ 대소문자 구분

- lastName 과 LastName 은 서로 다름

➤ 표기법

- Hyphens : first-name, last-name
- Underscore : first_name, last_name
- Camel Case : FirstName, LastName
- JavaScript에서는 첫 글자가 소문자인 Camel Case 방식 사용 : firstName, lastName



➤ JavaScript Statements

- HTML에서 자바스크립트 문장은 웹 브라우저가 실행해야하는 지시문
- 자바스크립트 프로그램을 구성하는 기본단위로 세미콜론(;)으로 구분
 - 일부의 경우 세미콜론은 생략이 가능하나 무조건 사용할 것을 권장

```
document.getElementById("demo").innerHTML = "Hello Dolly.";
```

➤ JavaScript Programs

- 한 줄에 여러 개의 문장을 사용할 수 있음.
- White Space : 여러 개의 공백은 하나로 간주
- 가독성을 위해 연산자 주변에 공백 사용을 권장

```
var x = y + z;
```

- 1줄의 코드라인은 보통 80 글자 미만으로 사용할 것을 권장

➤ JavaScript Code Blocks

- 자바스크립트 문장은 함께 실행되어야 하는 코드블럭으로 묶일 수 있음
- 코드블럭은 중괄호({ }, curly brackets)로 묶임
- 들여쓰기 : 일반적으로 Space 4칸

```
function myFunction() {  
    document.getElementById("demo").innerHTML = "Hello Dolly.";  
    document.getElementById("myDIV").innerHTML = "How are you?";  
}
```

➤ JavaScript Keyword

- 자바스크립트 문장은 특정 행동을 수행하게 하는 keyword로 시작함.
- break, if, for, do... while, switch, function 등

➤ 한줄주석

```
// Change heading:  
document.getElementById("myH").innerHTML = "My First Page";  
var x = 5;      // Declare x, give it the value of 5  
var y = x + 2;  // Declare y, give it the value of x + 2
```

➤ 여러줄 주석

```
/*  
The code below will change  
the heading with id = "myH"  
and the paragraph with id = "myP"  
in my web page:  
*/  
document.getElementById("myH").innerHTML = "My First Page";  
document.getElementById("myP").innerHTML = "My first paragraph.";
```

➤ 변수 (Variables)

- 데이터 값을 저장하는 공간

```
var price1 = 5;  
var price2 = 6;  
var total = price1 + price2;
```

➤ 식별자 (identifiers)

- 문자, 숫자, _(underscore), \$(dollar sign)만 포함
- 문자나 _, \$로 시작
- 대소문자를 가림(case sensitive)
- 예약어(reserved words)는 사용할 수 없음

➤ 할당 연산자 (=)

- 같다는 의미는 == 로 사용

➤ 데이터 형태

- 숫자, 문자열 등 다양한 형태의 값 대입 가능

```
var pi = 3.14;  
var person = "John Doe";  
var answer = 'Yes I am!';
```

➤ 변수 선언

```
<p id="demo"></p>  
<script>  
var carName = "Volvo";  
document.getElementById("demo").innerHTML = carName;  
var person = "John Doe", carName = "Volvo", price = 200;  
// 한번에 다수의 변수 선언 가능  
</script>
```


➤ 산술 연산자 (Arithmetic Operators)

- +, -, *, /, %, ++, --

➤ 할당 연산자 (Assignment Operators)

- =, +=, -=, *=, /=, % =

➤ 문자열 연산자 (String Operators)

- +, +=

➤ 비교 및 논리 연산자 (Comparison and Logical Operators)

- ==, ===, !=, !==, >, <, >=, <=

JavaScript 데이터 타입(Data Types)

➤ 데이터 타입

- 숫자(numbers), 문자열(strings), 배열(arrays), 객체(objects) 등

```
var length = 16;           // Number
var lastName = "Johnson"; // String
var cars = ["Saab", "Volvo", "BMW"]; // Array
var x = {firstName:"John", lastName:"Doe"}; // Object
```

➤ 문자열(Strings)

```
var carName = "Volvo XC60"; // Using double quotes
var carName = 'Volvo XC60'; // Using single quotes
var answer = "It's alright"; // Single quote inside double quotes
var answer = "He is called 'Johnny'"; // Single quotes inside double quotes
var answer = 'He is called "Johnny"'; // Double quotes inside single quotes
```

JavaScript 데이터 타입 (Data Types)

3. JavaScript

➤ 숫자 (numbers)

```
var x1 = 34.00;    // Written with decimals
var x2 = 34;       // Written without decimals
var y = 123e5;     // 12300000
var z = 123e-5;    // 0.00123
```

➤ 불리언 (booleans)

```
var x = true;
var y = false;
```

➤ 배열 (Arrays)

➤ 객체 (Objects)

➤ Undefined, empty (""), Null

➤ 함수

- 특정 기능을 수행하기 위해 설계된 코드 블록
- 호출된 경우에만 실행

```
var x = myFunction(4, 3);           // Function is called, return value will end up in x

function myFunction(a, b) {
    return a * b;                   // Function returns the product of a and b
}
```

- parameter/arguments, return
- 코드를 재사용 할 수 있음

➤ 범위 (Scope)

- 접근 가능한 변수(객체, 함수 포함)의 집합
- 함수 범위를 가짐 : 함수의 내부에서 범위가 변경됨

➤ 지역변수(Local Variables)

- 함수 내부에서 선언된 변수는 함수 내부에서만 접근 가능
- 지역변수는 함수가 시작될 때 생성되고, 함수가 종료될 때 삭제됨

```
// code here can not use carName
function myFunction() {
    var carName = "Volvo";
    // code here can use carName
}
```

➤ 전역변수(Global Variables)

- 함수 외부에서 선언된 변수로 웹 페이지에 존재하는 모든 스크립트와 함수에서 접근 가능

```
var carName = " Volvo";  
// code here can use carName  
function myFunction() {  
    // code here can use      carName  
}
```

➤ 자동 전역변수

- 선언되지 않은 변수가 사용되는 경우, 자동적으로 전역변수로 등록

```
// code here can use carName  
function myFunction() {  
    carName = "Volvo";  
    // code here can use carName  
}
```

JavaScript 배열 (Arrays)

➤ 한 변수에 다수의 변수를 저장하기 위해 사용

➤ 배열의 선언

```
var cars = ["Saab", "Volvo", "BMW"];    // literal 방식  
var cars = new Array("Saab", "Volvo", "BMW"); // 객체 방식
```

- 위 2가지 모두 사용가능하나, 리터럴 방식이 보다 간편하고 가독성, 실행속도 측면에서 유리

➤ 배열 요소 접근

```
var name = cars[0];    // 값 사용  
cars[0] = "Opel";    // 값 변경
```

➤ 연관 (Associative) 배열

- 자바스크립트는 연관배열을 지원하지 않음
- String 타입의 이름을 붙이길 원하는 경우, 객체(object) 사용

➤ if, else, else if

```
if (time < 10) {  
    greeting = "Good morning";  
} else if (time < 20) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}
```


➤ switch, case

```
switch (new Date().getDay()) {  
    case 6:  
        text = "Today is Saturday";  
        break;  
    case 0:  
        text = "Today is Sunday";  
        break;  
    default:  
        text = "Looking forward to the Weekend";  
}
```

➤ For

```
for (i = 0; i < 5; i++) {  
    text += "The number is " + i + "<br>";  
}
```

➤ For / in

```
var person = {fname:"John", lname:"Doe", age:25};  
  
var text = "";  
var x;  
for (x in person) {  
    text += person[x];  
}
```

➤ While

```
while (i < 10) {  
    text += "The number is " + i;  
    i++;  
}
```

➤ Do / While

```
do {  
    text += "The number is " + i;  
    i++;  
}  
while (i < 10);
```

➤ break

```
for (i = 0; i < 10; i++) {  
    if (i === 3) { break; }  
    text += "The number is " + i + "<br>";  
}
```

➤ continue

```
for (i = 0; i < 10; i++) {  
    if (i === 3) { continue; }  
    text += "The number is " + i + "<br>";  
}
```

➤ 특징

- JavaScript에서 거의 대부분이 객체임
- 객체의 값은 이름(name):값(value)의 순서 없는 쌍으로 이루어 짐

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

- 객체의 생성

```
var person = new Object();  
person.firstName = "John";  
person.lastName = "Doe";  
person.age = 50;  
person.eyeColor = "blue";  
  
var x1 = {};    // literal 방식 (더 선호)
```

➤ 객체의 프로퍼티 (property)

```
var person = {fname:"John", lname:"Doe", age:25};  
person.nationality = "English";  
delete person.age;    // or delete person["age"];  
for (x in person) {  
    txt += person[x];  
}
```

➤ 객체의 메소드 (method)

```
var person = {  
    fname:"John",  
    lname:"Doe",  
    age:25,  
    fullName: function() {return this.firstName + " " + this.lastName;}  
};  
var x = person.fullName();
```

➤ 함수 선언

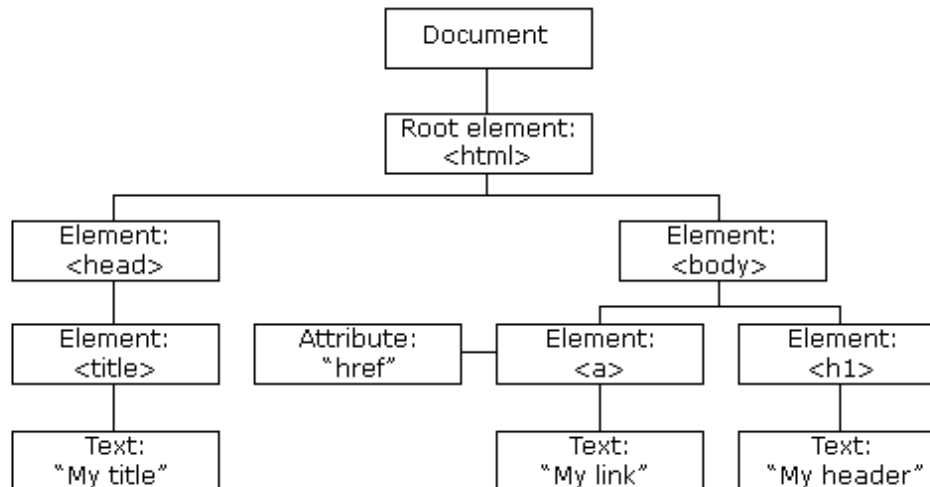
```
myFunction(5);  
function myFunction(y) {  
    return y * y;  
}  
  
(function () {  
    var x = "Hello!!";    // I will invoke myself  
})();  
  
function myFunction(a, b) {  
    return a * b;  
}  
var x = myFunction(4, 3) * 2;
```

➤ 클로저 (closer)

```
function add() {  
    var counter = 0;  
    counter += 1;  
}  
add();  
add();  
add();  
// counter dilemma  
  
var add = ( () {  
    var counter = 0;  
    return function () {return counter += 1;}  
})();  
add();  
add();  
add();  
// closer can solve this
```


➤ HTML DOM (Document Object Model)

- HTML DOM : HTML 문서의 객체 모델 및 프로그래밍 인터페이스의 표준
- HTML 요소(Object), HTML 요소의 프로퍼티와 메소드, 이벤트를 정의
 - HTML 요소의 값을 얻고, 변경하고, 추가하고, 삭제할 수 있는 표준을 제공
- DOM Tree of Objects



➤ The DOM Programming Interface

```
<html>
<body>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
</body>
</html>
```

- property : 가져오거나(get) 변경(set)할 수 있는 값(value)
 - innerHTML
- method : 수행할 수 있는 동작(action)
 - getElementById()

➤ HTML DOM Document Object

- HTML 문서에 접근하기 위해서는 항상 document 객체로 시작해야 함

➤ Finding HTML Elements

- `document.getElementById(id)`
- `document.getElementsByTagName(name)`
- `document.getElementsByClassName(name)`

➤ Changing HTML Elements

- `element.innerHTML = new html content`
- `element.attribute = new value`
- `element.setAttribute(attribute, value)`
- `element.style.property = new style`

➤ Adding and Deleting Elements

- `document.createElement(element)`
- `document.removeChild(element)`
- `document.appendChild(element)`
- `document.replaceChild(element)`
- `document.write(text)`

➤ Adding Events Handlers

- `document.getElementById(id).onclick = function() {code}`

➤ HTML Objects

- 참조 : http://www.w3schools.com/js/js_htmlDOM_document.asp

➤ HTML 요소 선택

```
var myElement = document.getElementById("intro");
var x = document.getElementsByTagName("p");
var x = document.getElementById("main");
var y = x.getElementsByTagName("p");
var x = document.getElementsByClassName("intro");
var x = document.getElementsByName("user_id");
var x = document.querySelector("p"); // IE8 이하에서 미동작
var x = document.querySelectorAll("p.intro"); // IE8 이하에서 미동작
var x = document.forms["frm1"];
var text = "";
var i;
for (i = 0; i < x.length; i++) {
    text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

➤ HTML / 속성(attribute) 변경

```
<html>
<body>
<p id="p1">Hello World!</p>


<script>
document.getElementById("p1").innerHTML = "New text!";
// document.write(Date());
document.getElementById("myImage").src = "landscape.jpg";
</script>

</body>
</html>
```

➤ CSS 변경

```
<body>

<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color = "blue";
</script>

<p>The paragraph above was changed by a script.</p>

</body>
</html>
```

➤ 이벤트 연결

```
<!DOCTYPE html>
<html>
<body onload="checkCookies()" >

<h1 onclick="changeText(this)">Click on this text!</h1>

<script>
function changeText(id) {
    id.innerHTML = "Ooops!";
}
document.getElementById("myBtn").onclick = displayDate;
</script>

</body>
</html>
```


➤ addEventListener

```
element.addEventListener("click", function(){ alert("Hello World!"); });
element.addEventListener("click", myFunction);
function myFunction() {
    alert ("Hello World!");
}

var x = document.getElementById("myBtn");
if (x.addEventListener) {           // For all major browsers, except IE 8 and earlier
    x.addEventListener("click", myFunction);
} else if (x.attachEvent) {        // For IE 8 and earlier versions
    x.attachEvent("onclick", myFunction);
}
```

➤ BOM : Browser Object Model

- 브라우저와 상호 연계할 수 있는 인터페이스 제공
- 브라우저별 일부 호환성 문제 발생

➤ Window 객체

- 자바스크립트의 모든 global 객체, 함수, 변수 등은 자동으로 window 객체의 멤버
- document 객체도 window 객체의 프로퍼티

```
window.document.getElementById("header");  
document.getElementById("header");
```

➤ Window 프로퍼티 , 메소드

- innerHeight/innerWidth
 - IE8 이하 : document.documentElement.clientHeight, document.body.clientHeight
- open(), close(), moveTo(), resizeTo()

➤ window.screen 객체

- 사용자 스크린에 대한 정보를 포함
- window를 생략하고 사용 가능 (screen.width)

```
document.getElementById("demo").innerHTML = "Screen Width: " + screen.width;  
document.getElementById("demo").innerHTML = "Screen Height: " + screen.height;  
document.getElementById("demo").innerHTML = "Available Screen Width: " +  
screen.availWidth;  
document.getElementById("demo").innerHTML = "Available Screen Height: " +  
screen.availHeight;  
document.getElementById("demo").innerHTML = "Screen Color Depth: " + screen.colorDepth;  
document.getElementById("demo").innerHTML = "Screen Pixel Depth: " + screen.pixelDepth;
```

➤ window.location 객체

- 현재 페이지의 URL 정보를 얻거나, 다른 페이지로 이동하는데 사용

```
document.getElementById("demo").innerHTML = "Page location is " + window.location.href;  
document.getElementById("demo").innerHTML = "Page hostname is " +  
window.location.hostname;  
document.getElementById("demo").innerHTML = "Page path is " + window.location.pathname;  
document.getElementById("demo").innerHTML = "Page protocol is " +  
window.location.protocol;  
window.location.assign("http://www.w3schools.com");  
window.location.href = "http://www.w3schools.com";  
window.location.reload();  
window.location.reload(true);
```

- window.history 객체
 - 브라우저의 히스토리 정보를 포함

```
<html>
<head>
<script>
function goBack() {
    window.history.back();
}
function goForward() {
    window.history.forward();
}
</script>
</head>
<body>

<input type="button" value="Back" onclick="goBack()">

</body>
</html>
```

➤ window.navigator 객체

○ 사용자의 브라우저에 대한 정보를 포함

- navigator의 정보는 정확하지 않거나 사용자에게 의해 변경될 수 있으므로 사용시 주의가 필요

```
document.getElementById("demo").innerHTML = "Cookies Enabled is " +  
navigator.cookieEnabled;  
document.getElementById("demo").innerHTML = "Name is " + navigator.appName + ". Code name  
is " + navigator.appCodeName;  
document.getElementById("demo").innerHTML = navigator.product;  
document.getElementById("demo").innerHTML = navigator.appVersion;  
document.getElementById("demo").innerHTML = navigator.userAgent;  
document.getElementById("demo").innerHTML = navigator.platform;  
document.getElementById("demo").innerHTML = navigator.language;  
document.getElementById("demo").innerHTML = navigator.javaEnabled();
```

➤ 3가지 종류의 팝업 박스가 존재

- alert box, confirm box, prompt box

```
alert("I am an alert box!");  
alert("Hello\nHow are you?");           // line break  
var r = confirm("Press a button");  
if (r == true) {  
    x = "You pressed OK!";  
} else {  
    x = "You pressed Cancel!";  
}  
var person = prompt("Please enter your name", "Harry Potter");  
if (person != null) {  
    document.getElementById("demo").innerHTML =  
        "Hello " + person + "! How are you today?";  
}
```

- window 객체는 코드는 특정 시간 간격으로 실행할 수 있게 함
 - setTimeout, setInterval

```
// setTimeout
<button onclick="myVar=setTimeout(myFunction, 3000)">Try it</button>
<button onclick="clearTimeout(myVar)">Stop it</button>

// setInterval
<p id="demo"></p>
<button onclick="clearInterval(myVar)">Stop time</button>
<script>
var myVar = setInterval(myTimer, 1000);
function myTimer() {
    var d = new Date();
    document.getElementById("demo").innerHTML = d.toLocaleTimeString();
}
</script>
```


➤ 쿠키는 웹페이지에 사용자의 정보를 저장할 수 있게 함

○ 사용자의 컴퓨터에 저장되는 작은 크기의 텍스트 데이터

```
document.cookie="username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path="/";
var x = document.cookie;
document.cookie="username=John Smith; expires=Thu, 18 Dec 2013 12:00:00 UTC; path="/";
document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC";

function setCookie(cname, cvalue, exdays) {
    var d = new Date();
    d.setTime(d.getTime() + (exdays*24*60*60*1000));
    var expires = "expires="+d.toUTCString();
    document.cookie = cname + "=" + cvalue + "; " + expires;
}

function getCookie(cname) {
    var name = cname + "=";
    var ca = document.cookie.split(';');
    for(var i=0; i<ca.length; i++) {
        var c = ca[i];
        while (c.charAt(0)==' ') c = c.substring(1);
        if (c.indexOf(name) == 0) return c.substring(name.length,c.length);
    }
    return "";
}
```

➤ 코딩 스타일 가이드

- 변수/함수명 : 소문자로 시작하는 camelCase 방식
- 연산자 주변에 공백
- 들여쓰기 : 스페이스바 4칸
- 문장(Statement) 작성 규칙
 - 문장은 항상 세미콜론(;)으로 마침
 - 코드 블록이 시작되는 중괄호는 첫번째 줄에 작성, 중괄호 앞에 공백 사용
 - 코드 블록 종료되는 중괄호는 마지막 줄의 가장 앞에 사용
 - 코드 블록 종료 시, 세미콜론 생략
- 객체(Object) 작성 규칙
 - 객체가 시작되는 중괄호는 첫 줄에 위치
 - 콜론(:) 이후에 공백
 - 문자열 값에는 따옴표(""), 숫자는 따옴표 없이 표시
 - 마지막 프로퍼티-값에는 콤마(,) 생략
 - 객체 종료 중괄호는 마지막 줄의 가장 앞에 표시하며, 세미콜론(;)으로 종료

JavaScript Coding Conventions

```
firstName = "John";
lastName = "Doe";

var x = y + z;
var values = ["Volvo", "Saab", "Fiat"];

function toCelsius(fahrenheit) {
    return (5 / 9) * (fahrenheit - 32);
}

var values = ["Volvo", "Saab", "Fiat"];
var person = {
    firstName: "John",
    lastName: "Doe",
    age: 50,
    eyeColor: "blue"
};
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

for (i = 0; i < 5; i++) {
    x += i;
}
```

➤ 코딩 스타일 가이드

- 한 줄은 되도록 80글자 이하로 작성
- 네이밍 규칙
 - 변수/함수명 : camelCase 방식 (Hyphens, Underscore, PascalCase, camelCase)
 - 전역변수(Global Variable) : 대문자
 - 상수(Constants) : 대문자
 - 첫문자를 \$로 시작하지 말 것 (다른 라이브러리와 혼동 우려)
- 외부 자바스크립트 링크시 간단한 문장 사용(type 속성 생략)
- HTML의 id나 class, name 등의 속성도 javascript와 동일한 네이밍 규칙 사용

- 전역변수(Global Variables)를 되도록 사용하지 않음
 - 되도록 지역변수를 사용하고, 필요한 경우에는 클로저(closer) 사용
- 모든 선언문은 상단에 위치
- 변수 초기화
- 숫자, 문자열, 불리언 객체를 생성하지 말 것
- new Object() 사용하지 말 것
- 자동 형 변환에 유의할 것
- 비교 시 === 사용
- 함수의 파라미터의 기본값을 지정할 것
- switch 문은 default 문을 항상 사용할 것
- eval() 함수 사용하지 말 것

```
// Declare at the beginning
var firstName, lastName, price, discount, fullPrice;
// Use later
firstName = "John";
lastName = "Doe";

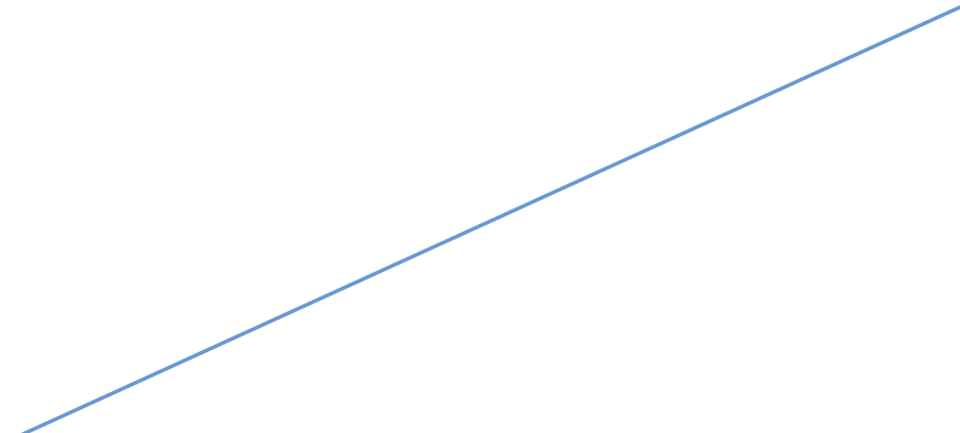
// Declare and initiate at the beginning
var firstName = "",
    lastName = "",
    price = 0,
    discount = 0,
    fullPrice = 0,
    myArray = [],
    myObject = {};

// 객체를 사용한 변수 선언 회피 (리터럴 방식 사용)
var x1 = {};           // new object
var x2 = "";           // new primitive string
var x3 = 0;            // new primitive number
var x4 = false;        // new primitive boolean
var x5 = [];           // new array object
var x6 = /()/;         // new regexp object
var x7 = function(){}; // new function object
```

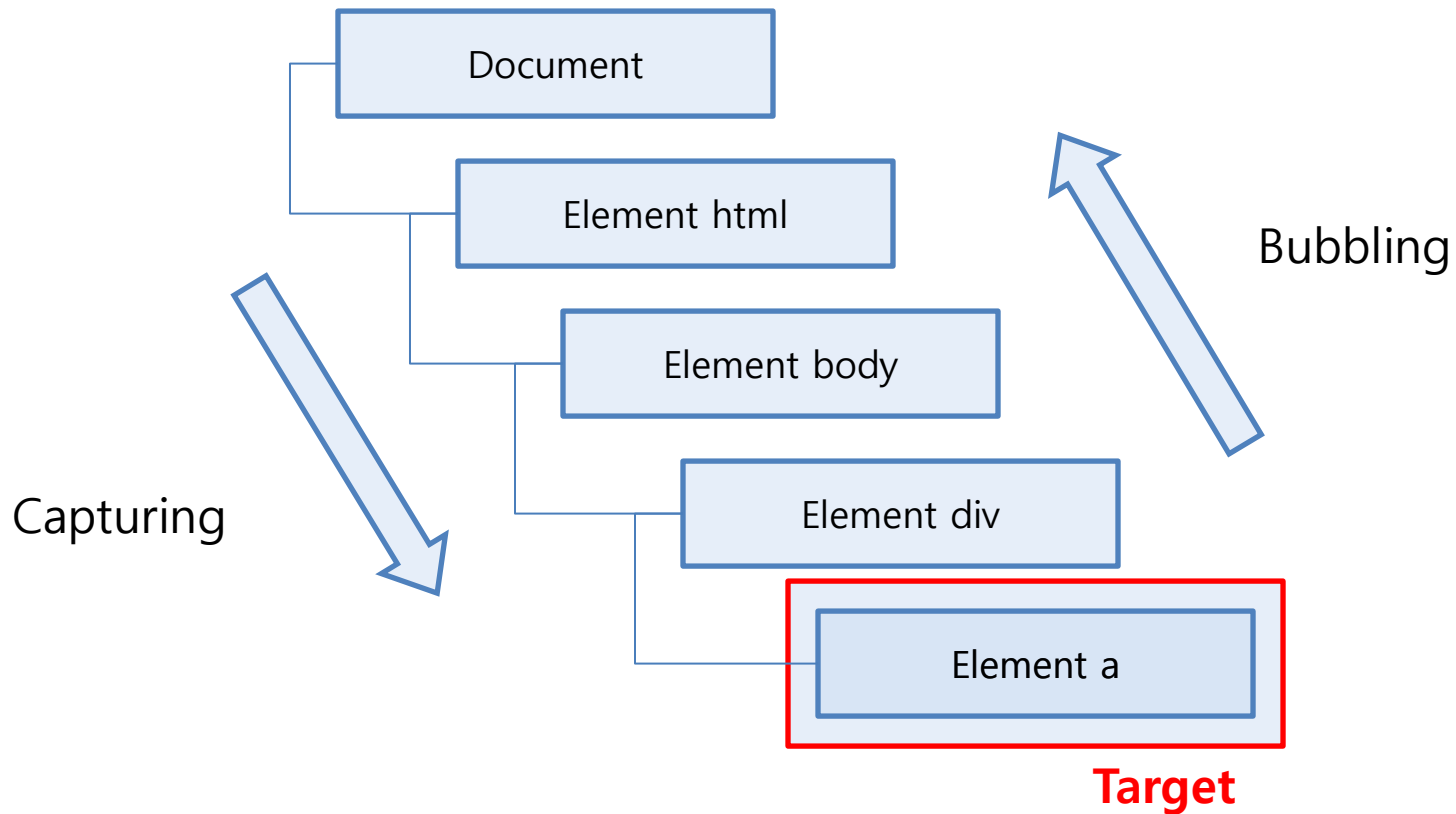
```
// Use parameter default
function myFunction(x, y) {
  if (y === undefined) {
    y = 0;
  }
}

switch (new Date().getDay()) {
  case 0:
    day = "Sunday"; break;
  case 1:
    day = "Monday"; break;
  case 2:
    day = "Tuesday"; break;
  case 3:
    day = "Wednesday"; break;
  case 4:
    day = "Thursday"; break;
  case 5:
    day = "Friday"; break;
  case 6:
    day = "Saturday"; break;
  default:
    day = "Unknown";
}
```

이벤트 (Events)

- 이벤트 흐름
 - 이벤트 핸들러
 - 이벤트 객체
 - 이벤트 타입
- 

- 이벤트 버블링 (Bubbling)
- 이벤트 캡처링 (Capturing)



➤ HTML 이벤트 핸들러

```
<input type="button" value="Click Me" onclick="alert('clicked')" />
```

➤ DOM 레벨 0 이벤트 핸들러

```
var btn = document.getElementById("myBtn");  
btn.onclick = function(){  
    alert("clicked");  
}  
  
btn.onclick = null; // 이벤트 제거
```

➤ DOM 레벨 2 이벤트 핸들러(표준)

```
var btn = document.getElementById("myBtn");  
btn.addEventListener("click", function(){  
    alert(this.id);  
}, false);    // 이벤트 제거는 못함  
  
btn.addEventListener("click", handler, false);  
btn.removeEventListener("click", handler, false); // 이벤트 제거  
var handler = function(){  
    alert("Clicked");  
}
```

➤ IE 이벤트 핸들러 (IE8 이하)

```
var btn = document.getElementById("myBtn");  
btn.attachEvent("onclick", function(){  
    alert(this.id);  
});    // IE8 이하에서는 버블링(false)만 지원  
  
btn.attachEvent("onclick", handler, false);  
btn.detachEvent("onclick", handler, false); // 이벤트 제거  
var handler = function(){  
    alert("Clicked");  
}
```

크로스브라우징 : jQuery를 쓰는 이유

➤ DOM 이벤트 객체

- bubbles
- cancelable
- **currentTarget**
- defaultPrevented
- detail
- eventPhase
- **preventDefault()**
- stopImmediatePropagation()
- **stopPropagation()**
- **target**
- trusted
- **type**
- view

e

event

➤ 이벤트 객체 (jQuery)

```
$("#button1").on("click", function(event){  
    event.preventDefault();  
    alert(event.type);  
});
```

```
$('a[href="#"]').on("click", function(e){  
    e.preventDefault();  
});
```

➤ UI 이벤트

○ load

- 페이지를 완전히 불러왔을 때 **window**에서, 모든 프레임을 완전히 불러왔을 때 프레임셋에서, 이미지나 객체를 완전히 불러왔을 때 **** 요소나 **<object>** 요소에서 발생

○ unload

- 페이지를 완전히 종료했을 때 **window**에서, 모든 프레임을 완전히 종료했을 때 프레임셋에서, 객체를 완전히 종료했을 때 **<object>** 요소에서 발생

○ abort

- **<object>** 요소의 콘텐츠를 완전히 내려받기 전에 사용자가 취소했을 때 해당 요소에서 발생

○ error

- 자바스크립트 에러가 발생했을 때 **window**에서, 이미지를 불러올 수 없을 때 해당 **** 요소에서, **<object>** 요소 콘텐츠를 불러올 수 없을 때 해당 요소에서, 프레임을 불러올 수 없을 때 해당 프레임셋에서 발생

○ select

- 사용자가 텍스트 박스(**<input>**)이나 **<textarea>**)에서 글자를 선택할 때 발생

○ resize

- **window**나 프레임의 크기를 바꿀 때 발생

○ scroll

- 사용자가 **스크롤바 있는 요소**를 스크롤할 때 발생. **<body>** 요소에는 페이지 전체에 대한 스크롤바가 있음

➤ Focus 이벤트

- blur
 - 요소가 포커스를 잃을 때 발생. 이 이벤트는 **버블링 되지 않음** (모든 브라우저 지원)
- focus
 - 요소가 포커스를 받을 때 발생. 이 이벤트는 **버블링 되지 않음** (모든 브라우저 지원)
- focusin
 - 요소가 포커스를 받을 때 발생. HTML 이벤트 focus의 **버블링 버전**
- focusout
 - 요소가 포커스를 잃을 때 발생. HTML 이벤트 blur의 **버블링 버전**

➤ Mouse/Wheel 이벤트

○ click

- 사용자가 주요 마우스 버튼(일반적으로 왼쪽 버튼)을 클릭하거나 엔터키를 누를때 발생. **키보드와 마우스에 모두 반응하므로 접근성 구현에 중요** (mousedown → mouseup → click)

○ dblclick

- 사용자가 주요 마우스 버튼(일반적으로 왼쪽 버튼)을 더블클릭할 때 발생. (DOM 레벨 3 이벤트에서 표준화)

○ mousedown

- 사용자가 마우스 버튼을 누를때 발생

○ mouseenter

- 마우스 커서가 요소 밖에서 요소 경계 안으로 처음 이동할 때 발생. 이 이벤트는 **버블링되지 않으며** 커서가 **자손 요소 위에 올라갈 때 발생하지도 않음**. (DOM 레벨 3 이벤트에서 추가)

○ mouseleave

- 마우스 커서가 요소 위에 있다가 요소 경계 밖으로 이동할 때 발생. **버블링되지 않으며** 커서가 **자손 요소 위에 올라갈 때 발생하지도 않음**. (DOM 레벨 3 이벤트에서 추가)

○ mousemove

- 마우스 커서가 요소 위를 이동하는 동안 **계속** 발생.

○ mouseout

- 마우스 커서가 요소 위에 있다가 다른 요소(경계 밖의 요소 혹은 자식 요소) 위로 이동할 때 발생.

○ mouseover

- 마우스 커서가 요소 바깥에 있다가 요소 경계 안으로 이동할 때 발생

○ mouseup

- 사용자가 마우스 버튼을 누르고 있다가 놓을 때 발생.

○ mousewheel (DOMMouseScroll)

- 사용자가 마우스 휠을 세로 방향으로 움직일 때 발생.

➤ 키보드/텍스트 이벤트

- keydown

- 사용자가 키를 처음 누를 때 발생하며 누르고 있는 동안 계속 발생

- keypress

- 사용자가 키를 누른 결과로 문자가 입력되었을 때 처음 발생하며 누르고 있는 동안 계속 발생. ESC 키에서도 발생. (DOM 레벨 3에서는 **keypress 이벤트를 폐기**했으며, textInput 이벤트를 권장)

- keyup

- 사용자가 키에서 손을 떼 때 발생

- textInput

- 텍스트가 텍스트 박스에 삽입되기 직전에 발생. (문자 입력전에 이벤트를 가로챌 수 있음)
- **DOM 레벨 3**에서 추가

➤ 조합(Composition) 이벤트

- compositionstart

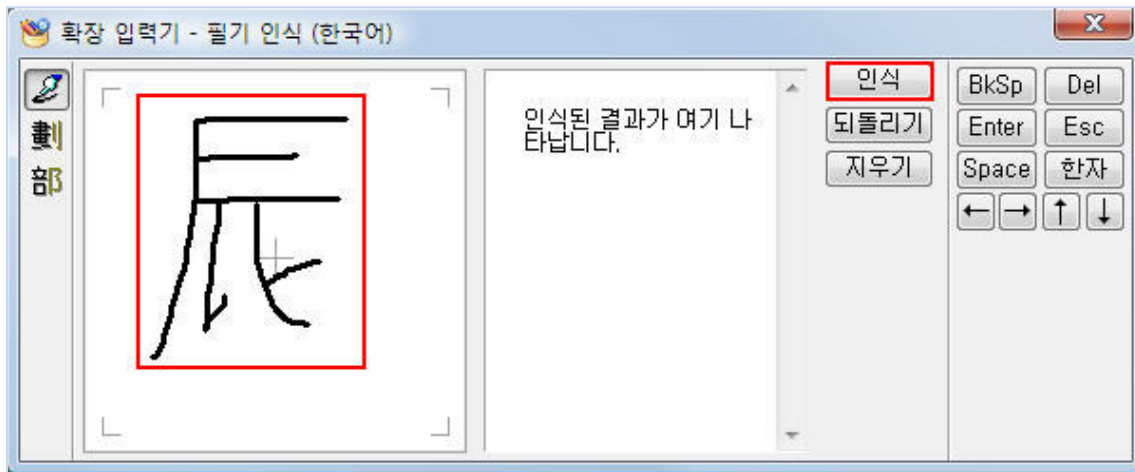
- IME의 텍스트 조합 시스템이 열리는 순간 발생하며 곧 문자가 입력될 것임을 나타냄.

- compositionupdate

- 입력 필드에 새 문자가 삽입될 때 발생.

- compositionend

- 텍스트 조합 시스템이 닫힐 때 발생하며 일반적인 키보드 입력으로 돌아갈 것을 나타냄.



➤ mutation(변경) 이벤트

- DOMSubtreeModified
 - DOM 구조가 어떻게든 바뀌었을 때 발생. 이벤트는 다른 이벤트가 모두 발생한 후에 발생. (DOM 레벨 2)
- DOMNodeInserted
 - 노드가 다른 노드의 자식으로 삽입될 때 발생.
- DOMNodeRemoved
 - 노드가 부모 노드로부터 제거될 때 발생.

➤ HTML5 이벤트

○ contextmenu

- **컨텍스트 메뉴**가 표시되려는 순간에 발생. 버블링되어 document 이벤트 핸들러 하나만 할당해서 페이지에서 발생하는 contextmenu 이벤트를 모두 처리 가능. (마우스 이벤트로 간주)

○ beforeunload

- window에서 발생. 브라우저가 페이지를 unload하기 직전에 발생하며 페이지를 언로드하겠다고 확인하지 않으면 계속 페이지에 머무르게 함.

○ DOMContentLoaded

- DOM 트리가 완전히 구성되는 즉시 발생.
- window의 load 이벤트는 페이지를 완전히 불러와야 발생하므로 외부 자원(이미지, JS, CSS 등)이 많이 포함된 페이지에서는 시간이 걸릴 수 있음.

○ readystatechange


- 문서나 요소를 불러오는 상황에 대한 정보를 제공
- uninitialized, loading, loaded, interactive, complete

○ pageshow, pagehide

- 페이지가 표시될 때 발생(pageshow) : load 이벤트 보다 늦게 발생(**cache에서 가져온 경우에도 발생**)
- 페이지가 사라질 때 발생(pagehide) : unload 이벤트 직전에 발생

○ hashchange

- URL 해시(URL에서 #기호 다음 전부)가 바뀔 때 발생

 <https://mail.google.com/mail/u/0/#inbox>

➤ 장치 이벤트

- orientationchange
 - 사용자가 장치를 가로 모드나 세로 모드로 바꿀 때 발생
- deviceorientation
 - 가속도계가 부착된 장치에서 관련 있는 동작을 감지했을 때 window에서 발생
 - x, y, z 축의 변화 정도(각도)를 감지
- devicemotion
 - 장치가 실제로 움직이고 있을 때 발생
 - 장치가 떨어지고 있거나 누군가가 장치를 걷고 있는지 판단할 때 유용
 - x, y, z 축의 가속도를 감지(중력 감안여부 선택 가능)

➤ 터치와 제스처 이벤트

- touchstart

- 손가락으로 화면을 터치할 때 발생. 이미 다른 손가락을 화면에 대고 있어도 다른 손가락을 대면 또 발생

- touchmove

- 손가락을 화면에서 움직일 때 계속 발생.
- 이 이벤트가 일어나는 동안 preventDefault()를 호출하면 스크롤을 막을 수 있음.

- touchend

- 손가락을 화면에서 뺄 때 발생.

- touchcancel

- 시스템에서 터치를 더 이상 추적하지 않을 때 발생. (명확하게 문서화되지는 않음)

- gesturestart

- 한 손가락을 화면에 얹은 채 다른 손가락으로 화면을 터치할 때 발생.

- gesturechange

- 화면에서 두 손가락 중 하나의 위치가 바뀔 때 발생.

- gestureend

- 두 손가락 중 하나를 화면에서 뺄 때 발생.

- 가상 포커스의 동작(voice over / iOS)
 - 요소선택 더블탭
 - 이벤트 생성(➔ click)
 - 더블탭 종료 후 ➔ touchstart ➔ touchend ➔ mousedown ➔ focusout() ➔ blur() ➔ mouseup ➔ click
 - 가상포커스와 실제 웹 브라우저의 포커스는 다른 형태의 존재임
 - 이벤트 설계시 복잡하지 않도록 구현(click 위주)