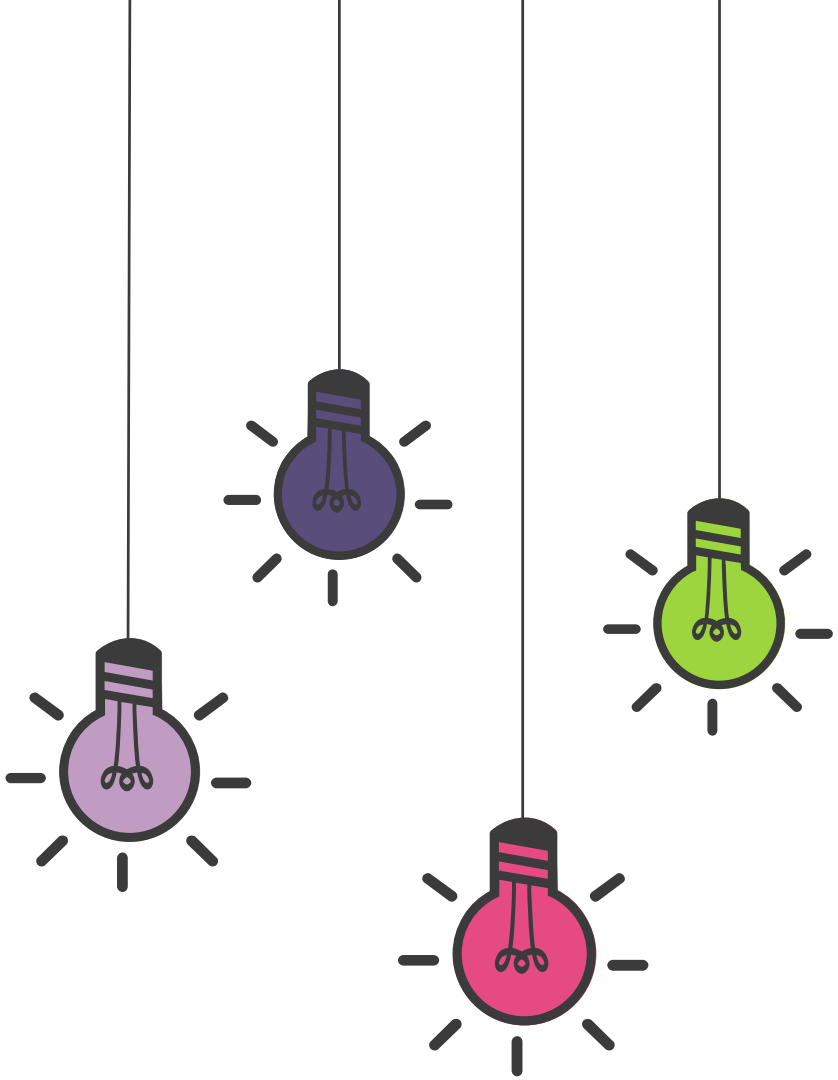


자바스크립트

J a v a S c r i p t

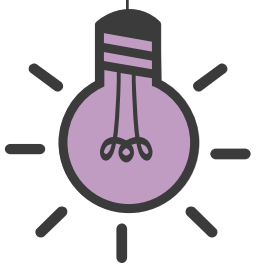
Copyright 2018.
안 효 인
All rights reserved.
무단 복제와 사용을 금합니다.

안 효 인
troment@nate.com

Index

CONTENTS

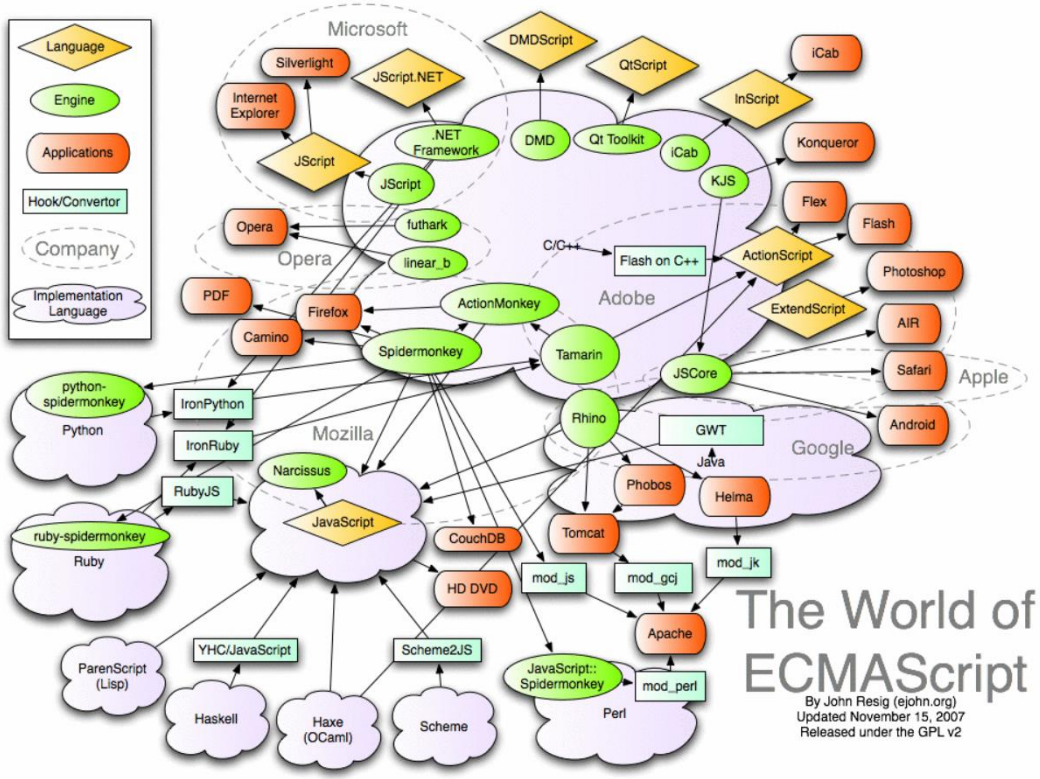
01 JavaScript 소개



1. JavaScript 소개

1-1. JavaScript 개요

- JavaScript는 프로토타입 기반의 스크립트 프로그래밍 언어로 객체지향 개념을 지원한다.
- 웹 브라우저가 JavaScript를 HTML과 함께 다운로드 하여 실행한다.
- 웹 브라우저가 HTML문서를 읽어 들이는 시점에 JavaScript Engine이 실행된다.
- 대부분의 JavaScript Engine은 ECMAScript 표준을 지원한다.

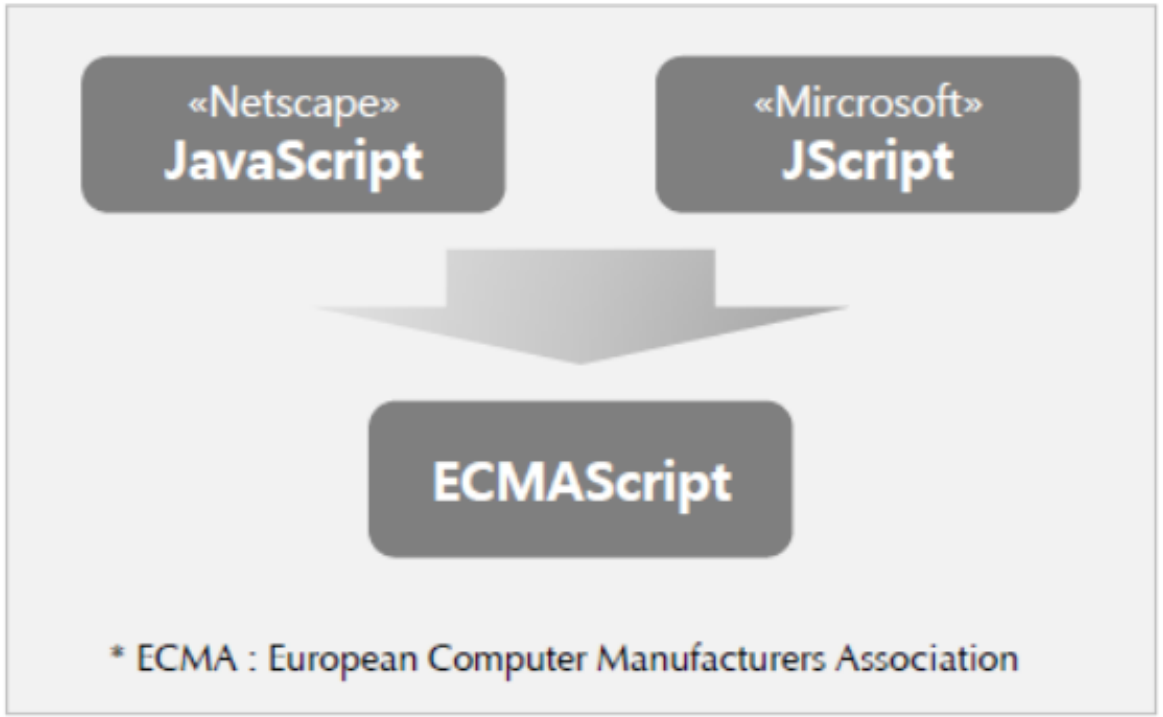


[ECMAScript 관련 기술 지도
via John Resig]

1. JavaScript 소개

1-2. ECMAScript(European Computer Manufacturers Association)

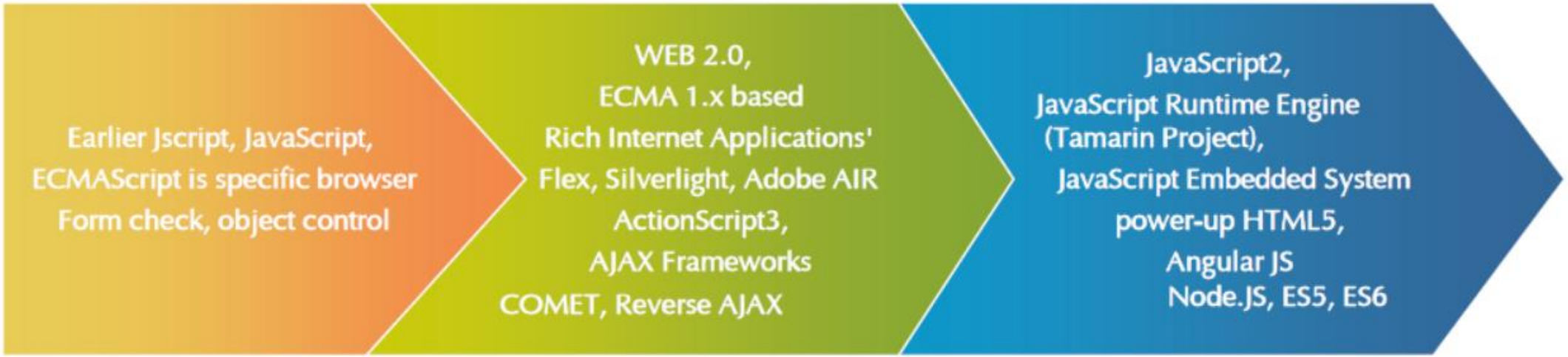
- 1996년 3월 Netscape는 'Netscape Navigator 2.0'을 출시하면서 JavaScript를 지원.
- Microsoft사는 웹에 호환되는 Jscript를 개발해서 1996년 8월 Internet Explore 3.0에 포함하여 출시.
- Netscape는 JavaScript 표준화를 위해 기술 규격을 ECMA에 제출.
- 1997년 6월 ECMA는 ECMA-262초안을 일반 회의에서 채택.



1. JavaScript 소개

1-3. JavaScript의 과거와 미래

- 1998년 ECMAScript(Edition 2)를 ISO 표준으로 개정.
- 1999년 정규 표현식, 문자열 처리, 새로운 제어문, 예외처리, 오류정의, 포매팅을 추가(Edition 3)
- Edition 4는 정치적인 문제로 폐기.
- Edition 5.1(2011년)을 거쳐 Edition 6(2015.06, Harmony)까지 표준화 됨.

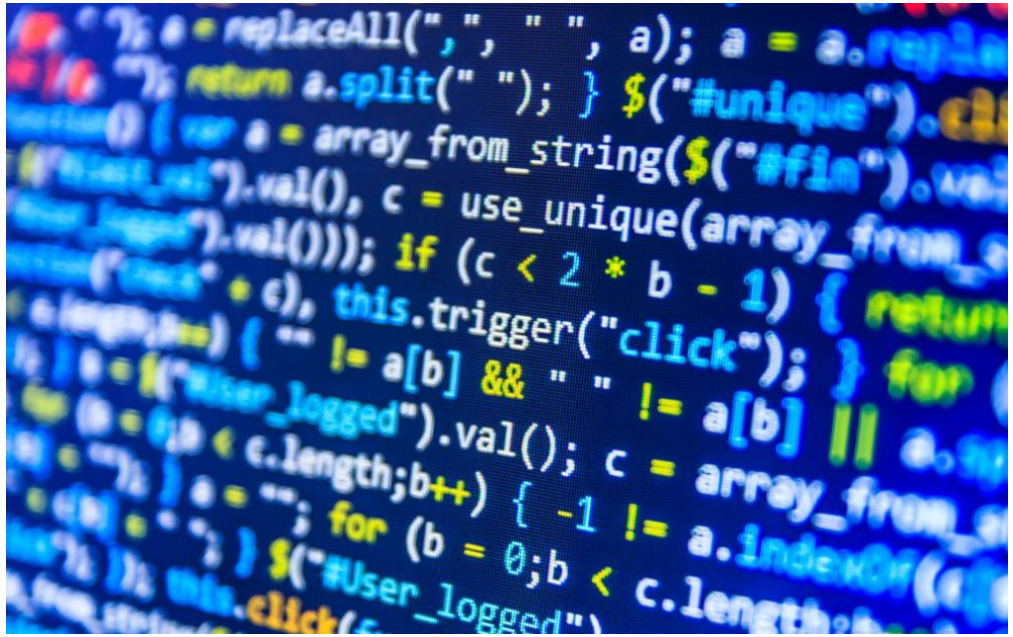


출처 : http://kangax.github.io/compat-table/es6

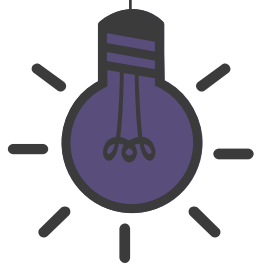
1. JavaScript 소개

1-4. 요약

- 웹브라우저는 JavaScript를 HTML과 함께 다운로드하고, 브라우저의 JavaScript Engine이 JavaScript를 실행한다.
- JavaScript는 클래스가 존재하지 않는 프로토타입 기반의 객체지향 언어이다. (Edition 6에서는 Class개념 지원)
- Netscape에서 처음 만들었으며, 이후 ECMA에서 ECMAScript라는 이름으로 표준화됨.
- 각 브라우저에서는 ECMAScript 스펙을 준수하는 방식으로 JavaScript를 지원한다.



02 JavaScript 시작



2. JavaScript 시작

2-1. JavaScript 선언(1/2)

- **HTML에서 JavaScript를 사용하려면 <script> 태그를 사용.**
- **<script> 태그는 'src'와 'type' 속성을 사용하여 JavaScript를 선언.(HTML5부터는 type속성 생략가능)**
- **src 속성은 외부의 JavaScript 파일을 HTML문서에 포함할 때 사용하며, 생략할 수 있다.**
- **Type 속성은 미디어 타입을 지정할 때 사용. JavaScript 코드는 'text/javascript'로 지정.**

JavaScript 코드를 HTML문서에 포함하는 방법

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JavaScript 선언</title>
<script type="text/javascript">
function hello(message) {
    alert('Hello ' + message);
}

hello('JavaScript !!!');
</script>
</head>
<body>

</body>
</html>
```

2-1.html

외부 JavaScript 파일을 HTML문서에 포함하는 방법

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JavaScript 선언</title>
<script src="hello.js" type="text/javascript"></script>
</head>
<body>

</body>
</html>
```

2-2.html

```
function hello(message) {
    alert('Hello ' + message);
}

hello('JavaScript !!!');
```

hello.js

2. JavaScript 시작

2-1. JavaScript 선언(2/2)

- **<script> 태그는 HTML 파일 내부의 <head>나 <body> 안 어느 곳에서나 선언 가능.**
- **하지만 <body> 안의 끝부분에 <script> 태그를 둘 것을 권장함.**
- **<head> 안에 위치한 JavaScript는 브라우저의 각종 입/출력 발생 이전에 초기화되므로 브라우저가 먼저 점검함.**
- **<body> 안에 위치하면 브라우저가 HTML부터 해석하여 화면에 그리기 때문에 사용자가 빠르다고 느낄 수 있음.**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>script tag의 위치</title>
        <script type="text/javascript">
        document.write("안녕하세요 자바스크립트입니다.");
        </script>
    </head>
    <body>
        아름다운 날이네요.
        <script type="text/javascript">
        document.write("날씨가 참 좋네요.");
        </script>
    </body>
</html>
```

<head>에 <script>태그를 두는 경우

<body>에 <script>태그를 두는 경우

2-3.html

2. JavaScript 시작

2-2. 구구단 출력

- 반복(Loop)문을 이용하여 브라우저 콘솔창에 구구단을 출력하는 예제.
- 변수(variable)를 선언할 때는 숫자형(type)이든 문자형(type)이든 모두 var keyword를 사용.
- for문을 사용해서 연산을 9번 반복 수행하는 코드 작성.
- 연산 결과를 선언한 변수에 저장한다.
- console.log() 함수를 이용하여 브라우저의 콘솔창에 결과 출력.

```html
<!DOCTYPE html>
<html>
    <head>
    <meta charset="UTF-8">
    <title>구구단</title>
</head>
    <body>
        <script type="text/javascript">
        //변수 선언.
        var dan = 7;
        var result = 0;

        console.log('** ' + dan + '단 ** ');
        //for문을 이용하여 9번 반복.
        for(var i=1;i<10;i++) {
            //곱셈 연산의 결과를 저장.
            result = dan * i;

            //콘솔창에 출력
            console.log(dan + ' * ' + i + ' = ' + result);
        }
        </script>
    </body>
</html>
```

2-4.html

Console창

```
** 7단 **
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
```

2. JavaScript 시작

2-3. 시계 구현

- JavaScript의 Date 내장 객체로 현재 시간을 출력.
- Date 객체의 toLocalTimeString() 함수를 호출하면 현재 시간을 return.
- 태그에 시간을 출력하기 위해 innerHTML 프로퍼티에 값을 할당.
- setInterval() 함수를 이용하여 1초마다 현재 시간을 업데이트.

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>시계</title>
        <script type="text/javascript">
        window.onload = function() {
            setInterval(showTime, 1000);
        };

        function showTime() {
            var view = document.getElementById("view");
            var date = new Date();
            view.innerHTML = date.toLocaleTimeString();
        }
        </script>
    </head>
    <body>
    현재 시간 : <span id="view"></span>
    </body>
</html>
```

2-5.html

실행결과

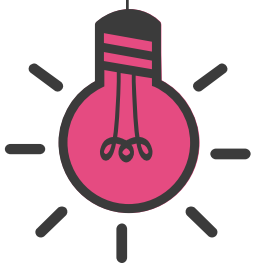
현재 시간 : 오후 4:13:44

2. JavaScript 시작

2-4. 요약

- HTML문서에서 JavaScript를 사용하려면 <script> 태그를 사용.
- JavaScript 코드는 HTML 파일 안에 두거나, 외부 JavaScript 파일(*.js)을 HTML 문서 안에 포함.
- <script> 태그는 HTML 문서의 어느 위치에서나 선언 가능하며, 일반적으로 <head>,<body> 내부에 위치.
- 웹 브라우저가 HTML 문서를 순차적으로 해석(parsing)하므로, script 위치에 따라 로드와 실행 시점이 달라진다.

03 JavaScript 기초



3. JavaScript 기초

3-1. 주석(comment)

- 주석은 JavaScript 코드에 대한 부연 설명이므로 실행 코드에 포함되지 않는다.
- JavaScript 주석은 한 줄 주석(Line Comment)과 블록 주석(Block Comment)이 있다.
- 한 줄 주석은 //code 로 표기하고, 블록 주석은 /* code */로 표기한다.
- 가능하면 블록 주석보다 라인 주석을 사용한다.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Insert title here</title>
        <script type="text/javascript">
        // 한줄 주석입니다.

        /*
        여러줄
        주석
        입니다.
        */
        </script>
    </head>
    <body>

    </body>
</html>
```

3-1.html

3. JavaScript 기초

3-2. 변수(variable)

- JavaScript는 변수를 선언할 때 타입을 명시하지 않고 var keyword를 사용하여 선언.
- 변수 이름은 함수 이름과 혼동되지 않도록 유일한 이름을 사용.(변수[형용사, 명사], 함수[동사] 사용)
- JavaScript는 ECMAScript 표준에 따라 낙타 표기법(Camel case)를 사용.
- 낙타 표기법이란 기본적으로 소문자로 작성하되 2개 이상의 단어일 경우 단어의 첫 글자는 대문자로 표기.

```
// 변수
var myName;
var myVariable;

// 적절
var listCount = 10;
var userName = "안효인";
var selected = false;

// 부적절 - 변수인지 함수인지 구별이 어렵다.
var getListCount = 10;
var isSelected = false;
```

3-1.html

3. JavaScript 기초

3-3. 자료형(data type) (1/6)

- 프로그램은 정적인 데이터 값을 동적으로 변환해 가면서 원하는 정보를 얻는다.
- 프로그램에서 다루는 데이터 값의 종류들을 자료 형(data type)이라 표현.
- JavaScript에서는 자료 형을 원시 타입(primitive type)과 객체 타입(object type)으로 분류.
- 원시 타입에는 숫자, 문자열, boolean, null, undefined와 같이 5가지가 있다. 이를 제외한 모든 값은 객체 타입이다.

자료 형	typeof 출력 값	설 명
숫자형	number	정수 또는 실수형.
문자열형	string	문자, single or double quotation으로 표기.
boolean형	boolean	참(true) or 거짓(false).
undefined	undefined	변수가 선언 되었지만 초기화가 되지 않을 경우.
null	object	값이 존재하지 않을 경우.

3. JavaScript 기초

3-3. 자료형(data type) (2/6) – 숫자(1/2)

- **JavaScript는 숫자를 정수와 실수로 나누어 구분하지 않는다.**
  - 모든 숫자를 8byte의 실수 형태로 처리.
  - 편의성을 위해 정수 리터럴과 실수 리터럴을 제공.
  - 숫자의 연산 처리시 실수 형태로 하기 때문에 특정 소수점을 정확하게 표현하지 못함.

- **기본 연산 기호는 Java나 C++과 같은 일반 프로그래밍 언어와 같다.(+, - , *, /, %)**

부동소수점 오류

```
var x = 0.3 - 0.2;
var y = 0.2 - 0.1;
console.log(x == y);     // false
console.log(x);          // 0.09999999999999998
console.log(y);          // 0.1
```

정확한 계산이 필요할 경우

```
var a = 0.3;
var b = 0.2;

var result = (a * 10 - b * 10) / 10;
console.log(result);     // 0.1
```

3-2.html

3. JavaScript 기초

3-3. 자료형(data type) (3/6) – 숫자(2/2)

- **JavaScript는 언더블로우, 오버플로우, 0으로 나누는 연산에 대해 예외를 발생 시키지 않는다.**
- **JavaScript에는 숫자와 관련된 특별한 상수가 존재한다.**
  - Infinity : 무한대를 나타내는 상수, 어떠한 수를 0으로 나누거나 Infinity를 어떠한 수로 사칙연산한 결과.
  - NaN(Not a Number) : 계산식의 결과가 숫자가 아님을 나타내는 상수.

```
// 언더플로우
console.log(0 / 100);        // 0
console.log(-0 / 100);       // -0

// 오버플로우
console.log(100 / 0);        // Infinity
console.log(-100 / 0);       // -Infinity
console.log(Infinity / 0);   // Infinity
console.log(-Infinity / 0); // -Infinity
```

실행결과

```
0
-0
Infinity
-Infinity
Infinity
-Infinity
```

3-3.html

```
console.log(0 / 0);              // Nan
console.log(parseInt('1A'));     // 1
console.log(parseInt('A'));      // NaN

console.log(new Number('1'));    // 1
console.log(new Number('1A'));   // NaN
```

실행결과

```
NaN
1
NaN
▼ Number i
  ▶ __proto__: Number
    [[PrimitiveValue]]: 1
▼ Number i
  ▶ __proto__: Number
    [[PrimitiveValue]]: NaN
```

3. JavaScript 기초

3-3. 자료형(data type) (4/6) - 문자열

- JavaScript에서 문자열은 16비트의 Unicode 문자를 사용.
- 문자 하나를 표현하는 char와 같은 문자형은 제공하지 않는다. ‘a’와 같은 한글자도 문자열로 표현.
- 작은 따옴표(‘, single quotes) 또는 큰 따옴표(“, double quotes) 둘다 사용 가능. 혼용 불가.
- 이스케이프 시퀀스(\)도 사용가능.

```
console.log("큰따옴표 문자열");                          // 큰따옴표 문자열
console.log('작은따옴표 문자열');                        // 작은따옴표 문자열
console.log("3.14");                                    // 3.14
console.log('문자열 안에 포함된 \'작은따옴표\' 표현');    // 문자열 안에 포함된 '작은따옴표' 표현
console.log("특수문자 사용\n줄바꿈 했다.");              // 특수문자 사용
                                                        // 줄바꿈 했다.
```

3-4.html

실행결과

```
큰따옴표 문자열
작은따옴표 문자열
3.14
문자열 안에 포함된 '작은따옴표' 표현
특수문자 사용
줄바꿈 했다.
```

3. JavaScript 기초

3-3. 자료형(data type) (5/6) – boolean, null과 undefined

- boolean은 비교 연산의 결과값으로 true 또는 false중 하나의 값을 갖는다.
- null은 값이 없거나 비어있음을 뜻하고, undefined는 값이 초기화 되지 않았음(정의되지 않음)을 의미.
- null과 undefined는 의미가 비슷하지만 값을 할당 하지 않은 변수는 undefined가 할당되고(시스템레벨), 코드에서 명시적으로 값이 없음을 나타낼 때(프로그램 레벨)는 null을 사용.

```
var name;
var id = null;

var n1 = 10;
var n2 = 20;

console.log(n1 == n2);   // false
console.log(name);       // undefined
console.log(id);         // null
```

실행결과

```
false

null
```

3-5.html

3. JavaScript 기초

3-3. 자료형(data type) (6/6) – 자동 형변환

- JavaScript는 Java나 C++등과 같은 언어와는 달리 자료 형에 대해 매우 느슨한 규칙이 적용.
- 어떤 자료 형이든 전달할 수 있고, 그 값을 필요에 따라 변환 가능.
- 서로 다른 자료 형의 연산이 가능.
- 모든 자료 형을 var로 선언하기 때문에 변수 선언은 쉽지만 이런 느슨한 규칙 때문에 혼란을 야기.

```
var msg = 40;

console.log("message : " + msg);          // message 40
msg = "hello javascript";
console.log(msg);                         // hello javascript

console.log("The message is " + 40);      // The message is 40
console.log(40 + " is The message");      // 40 is The message

console.log("40" - 5);                    // 35
console.log("40" + 5);                    // 405

console.log(parseInt("123.45") + 1);      // 124
console.log(parseFloat("123.45") + 1);    // 124.45

console.log("1.1" + "1.1");               // 1.11.1
console.log((+"1.1") + (+"1.1"));         // 2.2
```

실행결과

```
message : 40
hello javascript
The message is 40
40 is The message
35
405
124
124.45
1.11.1
2.2
```

3-6.html

3. JavaScript 기초

3-4. 상수(constant)

- **ECMAScript6 이전까지는 상수 표현을 지원하지 않음.**
  - 변수의 값을 변경하면 안 되는 상수와 일반 변수를 구분하고자 변수 명명 규칙을 다르게 하여 사용.
  - 상수의 표기법은 모든 문자를 대문자를 사용하고 단어 사이는 '_'로 표기.
- **ECMAScript6 에서는 const keyword가 추가되어 상수를 지원.**

3-7.html

ECMAScript6 이전

```
// 변수
var listCount = 0;
var selected = false;

// 상수 - ECMAScript6 이전 보편적 사용 방법
var LIST_COUNT = 10;
var SELECTED = false;

if(listCount < LIST_COUNT) {
    // todo
}
console.log('상수 초기값 : ' + LIST_COUNT);     // 10

LIST_COUNT = 20;
console.log('상수 변경값 : ' + LIST_COUNT);     // 20
```

```
상수 초기값 : 10
상수 변경값 : 20
```

ECMAScript6

```
// 변수
var listCount = 0;
var selected = false;

// 상수 - ECMAScript6
const LIST_COUNT = 10;
const SELECTED = false;

if(listCount < LIST_COUNT) {
    // todo
}
console.log('상수 초기값 : ' + LIST_COUNT);     // 10

LIST_COUNT = 20;                                //error 상수값 변경 불가능.
console.log('상수 변경값 : ' + LIST_COUNT);     // 10
```

```
상수 초기값 : 10
Uncaught TypeError: Assignment to constant variable.
```

3. JavaScript 기초

3-5. 연산자(operator)(1/4)

- **JavaScript에서 기본적으로 제공하는 약속된 문자의 표현 식을 연산자라 함.**
- **연산자는 산술 연산자, 비교 연산자, 논리 연산자, 기타 연산자 등을 제공.**
- **표현 식에서 2개 이상의 연산자를 동시에 사용했을 경우 우선순위별로 표현 식을 해석.**
- **괄호를 사용하여 우선순위 조절 가능.**

연산자	설명
++	선행 ++은 현재 값을 반환하고 값을 증가. 후행 ++은 값을 증가하고 결과값을 반환.
--	선행 --은 현재 값을 반환하고 값을 감소 후행 --은 값을 감소하고 결과값을 반환.
-	부호를 전환해서 결과값 반환.
+	숫자로 값을 반환.
~	비트단위 연산에서 사용하며 NOT 연산 수행.
!	논리연산에서 사용하며 boolean 결과를 반환.
delete	프로퍼티를 제거.
typeof	피연산자 타입을 리턴.
void	Undefined 값을 알려줌.
*, /, %	곱하기, 나누기, 나머지 연산결과 반환.

연산자	설명
+, -	값이 숫자일 때는 더하기, 빼기의 연산결과 반환.
+	값이 문자열이면 문자열을 서로 결합.
<<	비트연산자로 값을 왼쪽으로 이동.
>>	비트연산자로 값을 오른쪽으로 이동.
>>>	부호비트 확장 없이 값을 오른쪽으로 이동.
<, <=, >, >=	숫자의 대소비교.
instanceof	객체가 특정 객체의 타입인지를 확인.
in	프로퍼티가 존재하는지 확인.
==	동등 관계인지 확인.
!=	동등하지 않은 지 확인.

3. JavaScript 기초

3-5. 연산자(operator)(2/4)

- **연산자는 연산의 대상이 되는 값에 따라서 동작이 결정.**
- **'+' 연산자는 대상의 값이 모두 숫자인 경우 산술 연산을 수행.**
- **'+' 연산자는 대상 중에 문자열이 포함된 경우 모든 연산 대상을 문자열로 변환하고 문자열결합.**
- **연산자는 종류에 따라 1항 연산자, 2항 연산자, 3항 연산자로 구분.**

연산자	설명
===	값이 일치하는지 확인.
!==	값이 일치하지 않는지 확인.
&	비트 단위 연산자로 AND연산.
^	비트 단위 연산자로 XOR연산.
|	비트 단위 연산자로 OR연산.
&&	AND 연산.
||	OR 연산.
?:	조건에 해당하는 구문을 수행.
=	변수 또는 프로퍼티에 값을 할당.
,	1번째 구문은 버리고 다음 구문 값을 반환.

3. JavaScript 기초

3-5. 연산자(operator)(3/4) – 활용(1/2)

- a++와 a—는 각각 a = a + 1과 a = a – 1 연산의 축약 형태로써 증감연산자라 한다.
- 증감연산자가 앞에 오면 선증감 후실행, 뒤에오면 선실행 후증감.
- ! 연산은 NOT의 의미로 boolean형의 값을 반대로 반환.
- typeof는 해당 변수의 타입을 반환.

```
var num1 = 10;
var num2 = 20;
var bool = true;

console.log('num1++ : ' + num1++);              // 10
console.log('num1 : ' + num1);                  // 11
console.log('--num1 : ' + --num1);              // 10
console.log('!bool : ' + !bool);                // false

console.log('typeof bool : ' + typeof bool);    // boolean
console.log('typeof num1 : ' + typeof num1);    // number

console.log('num1 + num2 : ' + num1 + num2);    // 1020
console.log('num1 + num2 : ' + (num1 + num2));  // 30
console.log('num1 - num2 : ' + (num1 - num2));  // -10
console.log('num1 * num2 : ' + (num1 * num2));  // 200
console.log('num1 / num2 : ' + (num1 / num2));  // 0.5
```

실행결과

```
num1++ : 10
num1 : 11
--num1 : 10
!bool : false
typeof bool : boolean
typeof num1 : number
num1 + num2 : 1020
num1 + num2 : 30
num1 - num2 : -10
num1 * num2 : 200
num1 / num2 : 0.5
```

3-8.html

3. JavaScript 기초

3-5. 연산자(operator)(3/4) – 활용(2/2)

- 논리값을 비교하여 참(true), 거진(false)을 판단.
- 비교연산자 ==, ===의 차이점은 자료 형까지 비교하는지 아닌지의 여부.
- 비교연산자 &&는 둘 중 하나라도 거짓(false)이면 false, ||는 둘 중 하나라도 참(true)이면 true를 반환.
- 3항 연산자의 ? 앞 비교 값이 참(true)이면 : 앞의 값이 거짓(false)이면 : 뒤의 값이 반환.

3-9.html

```
var num = 10;
var str = "10";

console.log('num == str : ' + (num == str));         // true
console.log('num === str : ' + (num === str));       // false

str = "20";

console.log('num != str : ' + (num != str));         // true
console.log('num !== str : ' + (num !== str));       // true

console.log('true && true : ' + (true && true));     // true
console.log('true && false : ' + (true && false));   // false
console.log('true || true : ' + (true || true));     // true
console.log('true || false : ' + (true || false));   // true

console.log('num > num ? "ture" : "false" >> ' + (num > num ? "ture" : "false"));    // false
```

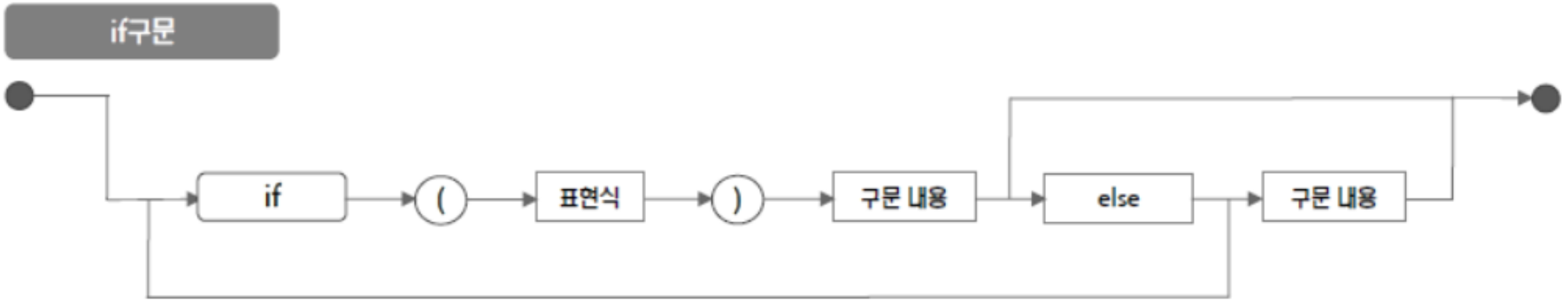
실행결과

```
num == str : true
num === str : false
num != str : true
num !== str : true
true && true : true
true && false : false
true || true : true
true || false : true
num > num ? "ture" : "false" >> false
```

3. JavaScript 기초

3-6. 조건문(conditional)(1/3) - if

- 표현 식의 값에 따라 특정 구문들을 실행 하거나 실행 하지 않도록 제어.
- 조건 구문을 분기 구문이라고도 표현하는데 2개 이상의 경로 중에서 특정 경로를 선택 가능.
- if 구문은 단순한 결정을 내리거나 정교한 구문들을 표현하는데 사용.
- if와 else의 표현 식 결과 값이 참(true)일 경우 구문을 실행. 모두 거짓(false)일 경우 else문 실행.

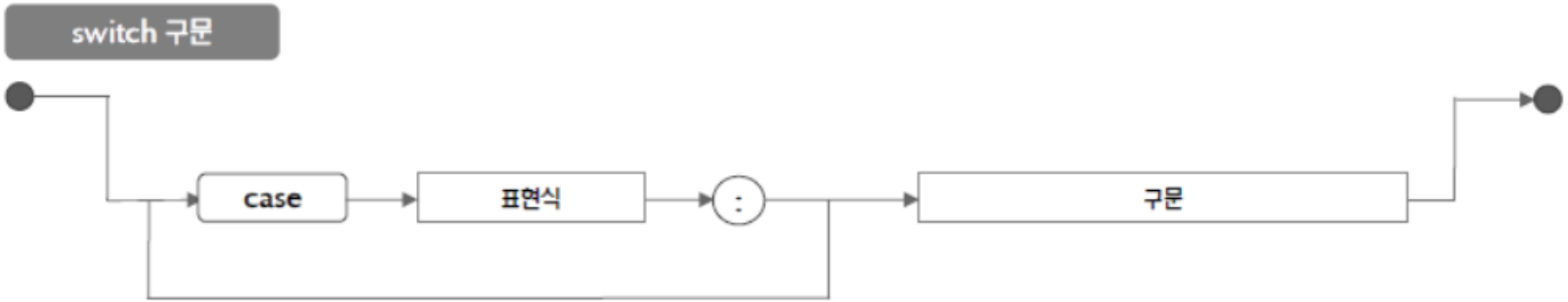



```
if(조건식1) {
    // 실행 구문1
} else if(조건식2) {
    // 실행 구문2
} else {
    // 실행 구문N
}
```

3. JavaScript 기초

3-6. 조건문(conditional)(2/3) - switch

- switch 구문은 동일한 표현식이 반복될 때 효과적인 구문.
- 값이 case와 동일 할 경우 해당 구문내용이 실행.
- break문은 switch 구문을 종료하며, break문이 없을 경우 다음 case를 실행.
- 동일한 case가 없을 경우 default 구문 실행.




```
switch(값) {
    case 값1 :
        // 실행 구문1;
    break;
    case 값12 :
        // 실행 구문2;
    break;
    default :
        // 실행 구문N;
}
```

3. JavaScript 기초

3-6. 조건문(conditional)(3/3) - 활용

- if 구문의 예제는 변수 score의 범위에 따라 해당하는 조건의 구문을 실행.
- if 조건이 거짓(false)이면 순차적으로 else if 조건을 확인하고, 모두 거짓일 경우 else문을 실행.
- switch 구문의 예제는 위 if예제를 switch 구문으로 변형한 예제.
- case문에 break를 빠뜨리지 않도록 주의.

```
var score = 95;

if(score >= 90) {
    console.log('A 학점입니다.'); // A 학점입니다. 출력됨.
} else if(score >= 80) {
    console.log('B 학점입니다.');
} else if(score >= 70) {
    console.log('C 학점입니다.');
} else {
    console.log('F 학점입니다.');
}
```

실행결과

```
A 학점입니다.
F 학점입니다.
```

3-10.html

```
var score2 = 78;

switch(parseInt(score2 / 10)) {
case 10 : case 9 : console.log('A 학점입니다.'); break;
case 8 : console.log('B 학점입니다.'); break;
case 7 : console.log('C 학점입니다.'); break; // C 학점입니다. 출력됨.
default : console.log('F 학점입니다.');
}
```

3. JavaScript 기초

3-7. 반복문(loop)(1/4) - while

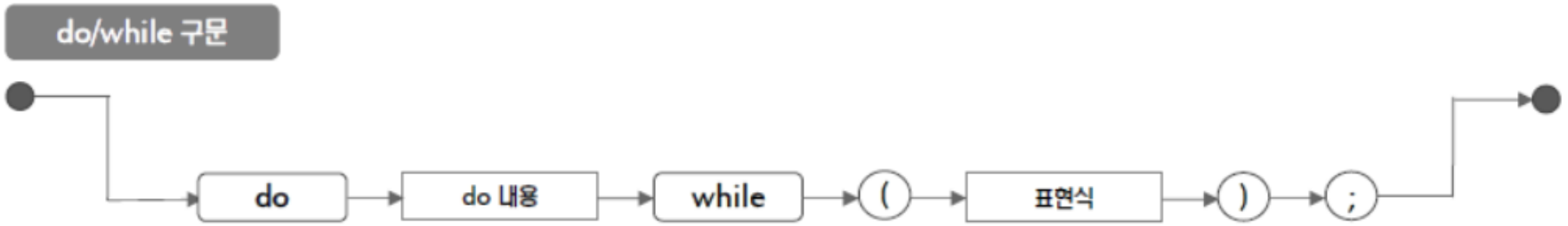
- 표현 식의 값이 참일 때 선언된 구문을 수행.
- 표현 식의 값이 거짓일 때 while문 종료.
- while 구문은 무한 반복에 빠지는 상황에 쉽게 노출.
- 표현 식의 값이 변수를 참조 하고 변수가 거짓이 될 수 있는 상황을 만들어 무한 반복 상황을 피해야함.


```
while(조건식) {
    // 실행 구문
}
```

3. JavaScript 기초

3-7. 반복문(loop)(2/4) – do/while

- 표현 식의 값을 확인하는 시점이 구문의 마지막이므로 최소 한번 이상 반복 구문을 수행.
- while과 문법적인 차이만 있을 뿐 수행 방법은 비슷함.
- 최초 1회 내용(body)을 수행하고 표현 식의 값이 참일 때 다음 내용을 수행.
- 표현 식의 값이 거짓일때 do문을 종료.




```
do {
    // 실행 구문
} while(조건식);
```

3. JavaScript 기초

3-7. 반복문(loop)(3/4) - for

- 프로그래밍 언어에서 가장 많이 사용하는 구문.
- for 구문은 카운터 변수를 사용하는 구문과 in keyword를 사용하는 구문으로 분류.
- 카운터 변수를 사용하는 for 구문은 카운터 변수가 표현 식에 명시된 조건이 거짓이 나오는 순간 반복 종료.
- in keyword를 사용하는 for 구문은 배열 또는 객체가 가진 프로퍼티를 순회하면서 키(index) 값을 조회


```
for(초기화;조건식;증감표현식) {
    // 실행 구문
}

for(var 변수 in 배열 or 객체) {
    // 실행 구문
    // 변수에는 배열의 인덱스
    // 객체의 프로퍼티 명이 담김
}
```

3. JavaScript 기초

3-7. 반복문(loop)(4/4) - 활용

- 1부터 10까지 숫자의 합. 반복문을 사용.
- 다른 반복 문과는 다르게 do – while문은 꼭 한번은 실행되어야 할 때 사용.
- break문을 만나면 조건식이 참이더라도 반복문 종료.
- continue문을 만나면 현재 반복문 블록의 아래 로직은 수행하지 않고 위로 올라가 다음 반복 실행.

```
var number = 1, sum = 0;

while(number < 11) {
    sum += number;
    number++;
}
console.log(sum);
```

```
sum = 0;

for(var number=1;number<11;number++) {
    sum += number;
}
console.log(sum);
```

```
number = 1;
sum = 0;

do{
    sum += number;
    number++;
} while(number < 11);
console.log(sum);
```

```
number = 1;
sum = 0;
while(number < 11) {
    sum += number;
    if(number == 5) {
        break;
    }
    number++
}
console.log(sum);
```

실행결과

```
55
55
55
15
```

3-11.html

3. JavaScript 기초

3-8. 함수(function)(1/3) – 선언, 호출

- JavaScript에서 함수는 일급(first-class) 객체이다.
- 함수를 변수, 객체, 배열 등에 저장할 수 있고 다른 함수에 전달하는 전달 인자 또는 리턴 값으로 사용 가능.
- 함수는 프로그램 실행 중에 동적으로 생성 가능.
- 함수 정의 방법은 함수 선언문, 함수 표현식, Function 생성자(constructor) 함수 세가지 방식 제공.

```
// 함수 선언문
function 함수이름(매개변수1, 매개변수2, … , 매개변수n) {
  //함수 내용
}
```

```
// 함수 표현식
var 함수이름 = function (매개변수1, 매개변수2, … , 매개변수n) {
  //함수 내용
}
```

```
// Function 생성자 함수
var 함수이름 = new Function("매개변수1", "매개변수2" , … , "매개변수n" , "함수내용");
```

```
// 함수호출
함수이름(매개변수1, 매개변수2, … , 매개변수n);
```

3. JavaScript 기초

3-8. 함수(function)(2/3) - 매개변수

- 함수의 정의 부분에 외부로부터 전달받을 변수를 매개변수(parameter)라 함.
- 함수를 호출할 때 전달하는 값을 전달인자(argument)라고 함.
- JavaScript에서 함수 정의 시 매개변수에 대한 타입은 명시하지 않는다.
- 함수를 호출할 때 정의된 매개변수와 전달인자의 개수가 일치하지 않더라도 호출 가능.

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

function printResult(x, y) {
    var result = sum(x, y);
    console.log(result);    // 15
}

printResult(10, 5, 7)
```

3-12.html

3. JavaScript 기초

3-8. 함수(function)(3/3) - 활용

- 1부터 매개변수 num까지의 합을 구하는 예제.
- 각각 함수 선언문과 함수 표현식, Function 생성자 함수 호출이다.
- 방식은 달라도 함수 호출 방식은 같다.

```
// 함수 선언문
function sum1(num) {
    var sum = 0;
    for(var i=1;i<=num;i++) {
        sum += i;
    }
    console.log(sum);   // 55
}
sum1(10);
```

```
// 함수 표현식
var sum2 = function(num) {
    var sum = 0;
    for(var i=1;i<=num;i++) {
        sum += i;
    }
    console.log(sum);   // 55
}
sum2(10);
```

```
// Function 생성자 함수
var sum3 = new Function("num",
    "var sum = 0;" +
    "for(var i=1;i<=num;i++) {" +
    "   sum += i;" +
    "} " +
    "console.log(sum);" );
sum3(10);
```

실행결과

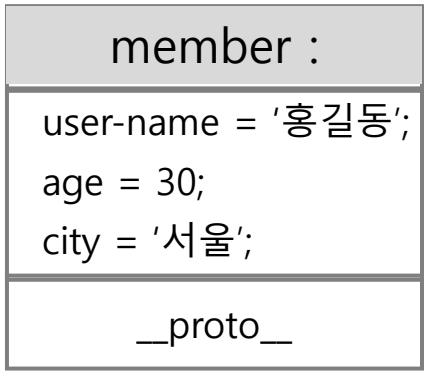
```
55
55
55
```

3-13.html

3. JavaScript 기초

3-9. 객체(object)(1/4) - 개요

- 객체는 이름과 값으로 구성된 프로퍼티의 집합.
- 문자열, 숫자, boolean, null, undefined를 제외한 모든 값은 객체.
- 전역 객체를 제외한 JavaScript 객체는 프로퍼티를 동적으로 추가하거나 삭제 가능.
- JavaScript 객체는 프로토타입(prototype)이라는 특별한 프로퍼티를 포함.



3. JavaScript 기초

3-9. 객체(object)(2/4) – 속성 값 조회

- 객체는 dot(.)을 사용하거나 대괄호([])를 사용해서 속성 값에 접근.
- 객체에 없는 속성에 접근하면 undefined를 반환.
- 객체 속성 값을 조회 할 때 || 연산자를 사용하는 방법도 가능.
  - 예 : var middle = employee['middle_name'] || 'none';

```
// 객체 리터럴
var empty_object = {};
var member = {
        "user-name" : "홍길동",
        "age" : 20,
        "city" : "서울"
};

// 객체의 속성에 접근
console.log(member.age);            // 1. dot 표기법
console.log(member["age"]);         // 2. [] 표기법

// 속성명에 연산자가 포함된 경우 [] 표기법만 접근 가능.
console.log(member["user-name"]);   // 홍길동
console.log(member.user-name);      // NaN
```

실행결과

```
20
20
홍길동
NaN
```

3-14.html

3. JavaScript 기초

3-9. 객체(object)(3/4) – 속성 값 변경

- **속성 값을 변경할 때는 dot(.)이나 대괄호([])를 사용.**
  - 예 : server['port'] = 80; or server.port = 80;
- **객체에 값을 할당하는 속성이 없을 경우, 그 속성은 추가됨.**
  - 예 : server['domain'] = 'korea.com';

```
// 객체 리터럴
var empty_object = {};
var member = {
        "user-name" : "홍길동",
        "age" : 20,
        "city" : "서울"
};

// 속성의 값 변경
member.age = 30;
member["city"] = "경기도";

console.log(member["age"]);      // 30
console.log(member.city);        // 경기도

// 속성 추가
member.tel = "010-1234-5678";
console.log(member.tel);         //010-1234-5678
```

실행결과

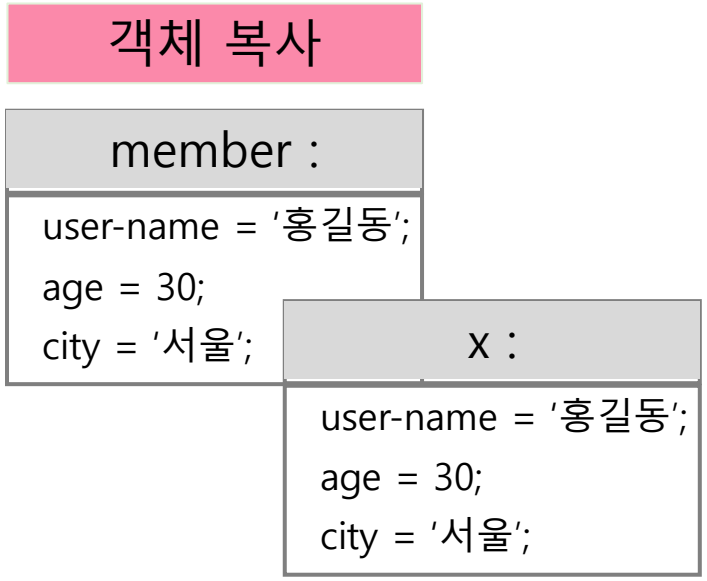
```
30
경기도
```

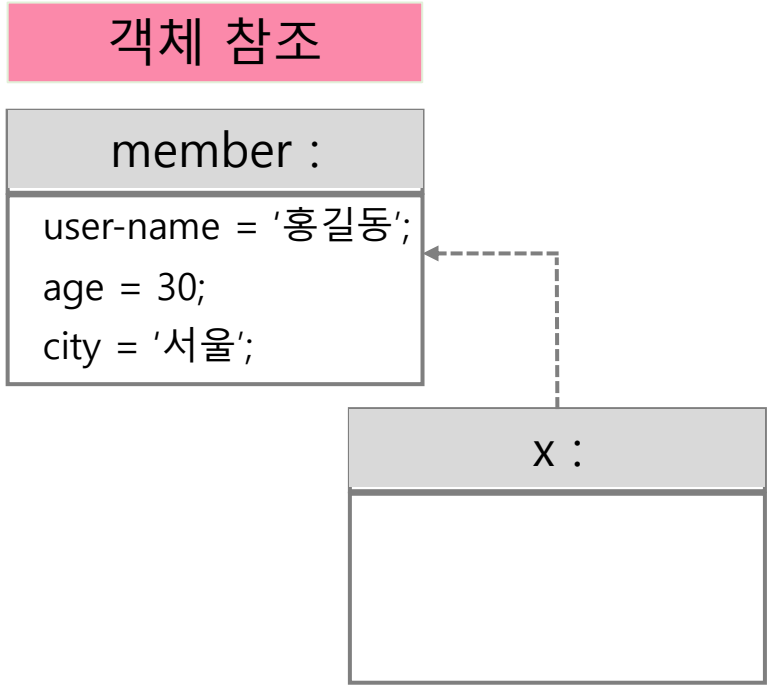
3-15.html

3. JavaScript 기초

3-9. 객체(object)(4/4) - 참조

- 객체는 복사되지 않고 참조된다.
- JavaScript에서 원시(Primitive) 데이터 타입이 아닌 모든 값은 참조 타입니다.
- 참조 타입은 Object, Array, Date, Error를 포함.
- 타입 확인 방법으로는 typeof 연산자가 있다.(null은 원시 타입이지만 typeof 연산자에서 object를 반환).






```
var x = member;
x.city = "제주도";
console.log(member.city);          //제주도
```

3-15.html

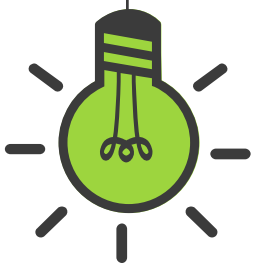
3. JavaScript 기초

3-10. 요약

- JavaScript의 변수는 var를 사용하여 선언하며 타입이 없다.
- 자료형은 원시타입(Primitive Type)과 객체타입(Object Type)으로 분류.
- 함수는 일급객체(First-class Object)로 변수에 저장하거나 함수의 전달인자 또는 반환 값으로 사용.
- 객체는 이름과 값으로 구성된 프로퍼티의 집합이며 프로퍼티를 동적으로 조작 가능.

```
for (var i = 0; i < 10; i += 1) {
        console.log(i);
}

// very common use case: looping ov
var pageNames = ['Home', 'About Us'
for (i = 0; i < pageNames.length; i
        if (document.title == pageN
                console.log('We ARE
        } else {
```

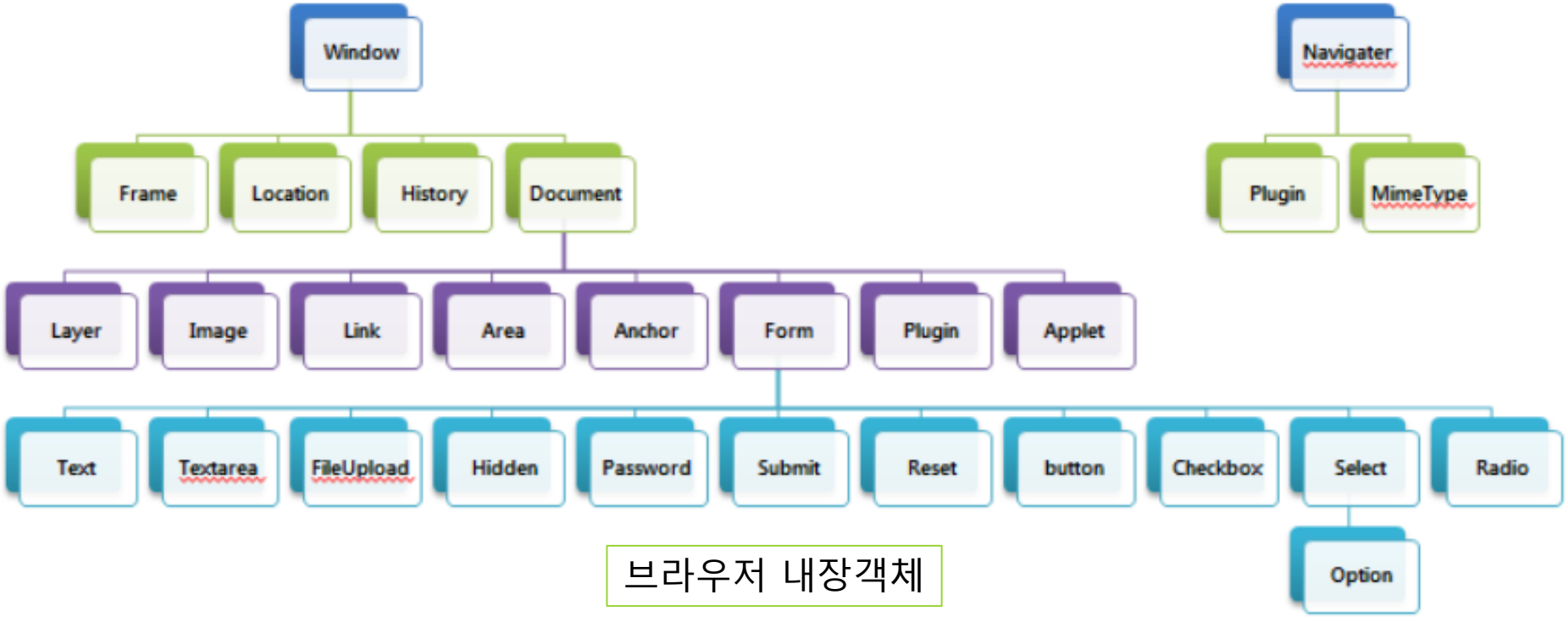
04 웹브라우저와 Window객체



4. 웹브라우저와 Window 객체

4-1. Window 객체 개요

- Window 객체는 웹 브라우저에서 작동하는 JavaScript의 최상위 전역객체이다.
- Window 객체에는 브라우저와 관련된 여러 객체와 속성, 함수가 있다.
- JavaScript에서 기본으로 제공하는 프로퍼티와 함수도 포함.(Number 객체, setInterval() 함수 등)
- BOM(Browser Object Model)으로 불기리도 함.



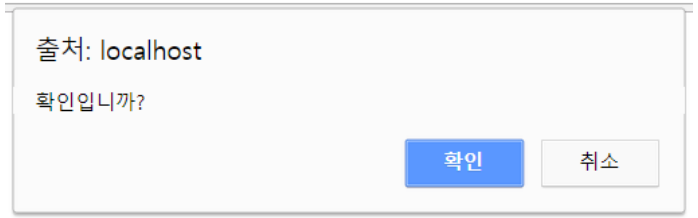
브라우저 내장객체

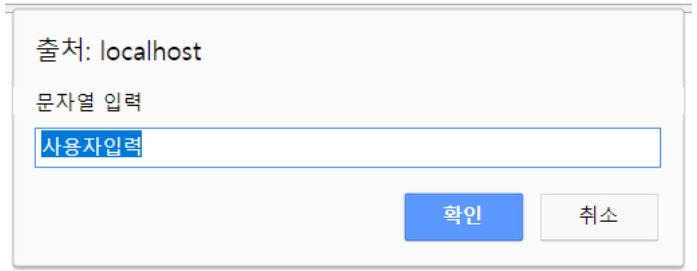
4. 웹브라우저와 Window 객체

4-2. Window 객체 사용(1/3) – alert, confirm, prompt

- window 객체의 함수를 호출하면 브라우저에서 제공하는 창을 open.
- alert()　: 브라우저의 알림창.
- confirm() : 브라우저의 확인/취소 선택창.
- prompt()　: 브라우저의 입력 창.

```
function openAlert() {
    alert("알림창입니다.");
}

function openConfirm() {
    if(confirm("확인입니까?")) {
        console.log("확인 눌렀네요.");
    } else {
        console.log("취소 눌렀네요.");
    }
}

function openPrompt() {
    var txt = prompt("문자열 입력", "사용자입력");
    console.log(txt);
}
```





확인 눌렀네요.
취소 눌렀네요.
안녕하세요

4-1.html

4. 웹브라우저와 Window 객체

4-2. Window 객체 사용(2/3) – navigator

- **navigator 객체는 브라우저의 정보가 내장된 객체.**
- **navigator의 정보로 서로 다른 브라우저를 구분할 수 있으며, 브라우저 별로 다르게 처리 가능.**
- **HTML5에서는 위치 정보를 알려주는 역할 가능.**

```
console.log("Browser CodeName    : " + navigator.appCodeName);
console.log("Browser Name        : " + navigator.appName);
console.log("Browser Version     : " + navigator.appVersion);
console.log("Browser Enabled     : " + navigator.cookieEnabled);
console.log("Platform        : " + navigator.platform);
console.log("User-Agent header  : " + navigator.userAgent);
```

4-2.html

chrome

```
Browser CodeName    : Mozilla
Browser Name        : Netscape
Browser Version     : 5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36
Browser Enabled     : true
Platform            : Win32
User-Agent header   : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36
```

Internet Explore

```
Browser CodeName    : Mozilla
Browser Name        : Netscape
Browser Version     : 5.0 (Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729; rv:11.0) like Gecko
Browser Enabled     : true
Platform            : Win32
User-Agent header   : Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729; rv:11.0) like Gecko
```

4. 웹브라우저와 Window 객체

4-2. Window 객체 사용(3/3) – location, history

- **location 객체를 이용하여 현재 페이지 주소(URL)와 관련 된 정보들을 알 수 있다.**
  - location.href : 프로퍼티에 값을 할당하지 않으면 현재 URL을 조회하고 값을 할당하면 할당 된 URL로 페이지 이동.
  - location.reload() : 현재 페이지를 새로고침.
- **history 객체는 브라우저의 페이지 이력을 담는 객체.**
  - history.back() / history.forward() : 브라우저의 뒤로 가기/앞으로 가기 버튼과 같은 동작.

```
console.log(location);
console.log(location.href);
location.href = "http://www.naver.com";
```

4-3.html

```
history.back();
history.forward();
```

4. 웹브라우저와 Window 객체

4-3. 새 창 열기(1/3)

- **window 객체의 open() 함수를 사용하면 새 창을 열 수 있다.**
- **window.open('페이지 URL', '창이름', '특성', 히스토리 대체여부);**
  - 창이름(string) : open 할 대상(_blank, _self등) 지정 혹은 창의 이름.
  - 특성(string) : 새로 열릴 창의 너비, 높이 등의 특성 지정.
  - 히스토리 대체여부(Boolean) : 현재 페이지 히스토리에 덮어 쓸지 여부(true/false)

창 특성			
width	픽셀	All	창의 너비
height	픽셀	All	창의 높이
top	픽셀	All	창의 세로(y) 좌표 위치
left	yes || no	All	창의 가로(x) 좌표 위치
menubar	yes || no	IE, Firefox, Opera	메뉴 표시줄
status	yes || no	IE, Firefox, Opera	상태 표시줄
scrollbars	yes || no	IE, Firefox, Opera	스크롤바
toolbar	yes || no	IE, Firefox	도구모음
resizable	yes || no	IE 전용	창 크기 조절 가능여부
location	yes || no	Opera 전용	주소 입력란

4. 웹브라우저와 Window 객체

4-3. 새 창 열기(2/3) – 창 열고 닫기

- **이벤트를 이용하여 특정 시점에 창을 열 수 있다.**
  - 페이지 로딩 완료 후 새 창 열기, 클릭할 때 새 창 열기등.
- **window 객체의 close() 함수로 현재 창을 닫을 수 있다.**
- **특히 브라우저에 내장 된 창이 아닌 JavaScript로 자체 구현한 팝업에서 필요.**

창 열기

4-4.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<button onclick="javascript:windowOpen();">
버튼 창열기
</button>
<a href="javascript:windowOpen();">링크 창열기</a>
<script>
function windowOpen() {
    window.open('./4-5.html', 'winname', 'width=300, height=200');
}
</script>
</body>
</html>
```

창 닫기

4-5.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<button onclick="javascript:windowClose();">
함수 이용해서 닫기
</button>
<a href="javascript:window.close();">
메소드 이용해서 닫기
</a>
<script>
function windowClose() {
    window.close();
}
</script>
</body>
</html>
```

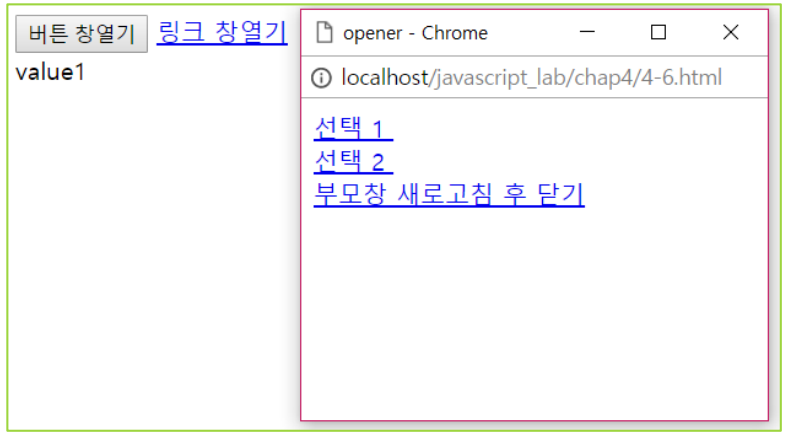
4. 웹브라우저와 Window 객체

4-3. 새 창 열기(3/3) – 부모 창 컨트롤

- **window 객체의 opener 속성을 이용하면 부모 창(새 창을 연 창)을 컨트롤 가능.**
  - 부모 창에 값 전달.
  - 부모 창을 새로 고침 하거나 페이지 이동.
- **opener 객체는 부모 창의 window 객체.**

4-6opener.html

```
<body>
<button onclick="javascript:windowOpen();">
버튼 창열기
</button>
<a href="javascript:windowOpen();">링크 창열기</a>
<div id="view"></div>
<script>
function windowOpen() {
    window.open('./4-6.html', 'winname', 'width=300, height=200');
}

function setData(data) {
    document.getElementById("view").innerHTML = data;
}
</script>
</body>
```



4-6.html

```
<script type="text/javascript">
// 부모 창에 값 전달 후 창 닫기
function setOpenerData(data) {
    window.opener.setData(data);
    window.close();
}
// 부모 창 새로고침 후 창 닫기
function reloadOpener() {
    opener.location.reload();
    self.close();
}
</script>
</head>
<body>
<a href="javascript:setOpenerData('value1');">
    선택 1
</a><br>
<a href="javascript:setOpenerData('value2');">
    선택 2
</a><br>
<a href="javascript:reloadOpener();">
    부모창 새로고침 후 닫기
</a>
</body>
```

4. 웹브라우저와 Window 객체

4-4. window 객체 프로퍼티

- **window 객체는 웹 브라우저에서는 구동 되는 JavaScript의 전역 객체.**
- **다음 장에서 살펴볼 document 객체는 HTML 문서와 관련된 객체로 가장 많이 사용하는 객체.**
- **screen 객체는 화면의 가로, 세로 크기 정보를 알 수 있다.**
- **pageYOffset등과 scroll() 함수를 이용하면 현재 화면의 크기를 계산하여 페이지 단위로 스크롤 제어가 가능.**

설 명			
self	현재 창 자신, window와 같음	statusbar	창의 상태 바
document	document 객체	toolbar	창의 툴 바
history	history 객체	personalbar	창의 퍼스널 바
location	location 객체	scrollbars	창의 스크롤 바
opener	open()으로 열린 창에서 볼 때 자기를 연 창	innerHeight	창 표시 영역의 높이(픽셀) (IE 지원 안함)
parent	프레임에서 현재프레임의 상위프레임	innerWidth	창 표시 영역의 너비(픽셀) (IE 지원 안함)
top	현재프레임의 최상위프레임	outerHeight	창 바깥쪽 둘레의 높이 (IE 지원 안함)
frames	창안의 모든 프레임에 대한 배열 정보	outerWidth	창 바깥쪽 둘레의 너비 (IE 지원 안함)
locationbar	location 바	pageXOffset	현재 나타나는 페이지의 X위치 (IE 지원 안함)
menubar	창 메뉴바	pageYOffset	현재 나타나는 페이지의 Y위치 (IE 지원 안함)

4. 웹브라우저와 Window 객체

4-5. window 객체 함수(1/2)

- **브라우저에서 버튼으로 제공하는 기능인 find, stop, print와 같은 함수도 있다.**
- **move 함수로 현재 열려 있는 창의 위치를 이동 가능.**

	설 명	
alert()	경고용 대화상자를 보여줌	
confirm()	확인, 취소를 선택할 수 있는 대화상자를 보여줌	
prompt()	입력창이 있는 대화 상자를 보여줌	
open()	새로운 창을 오픈	
scroll()	창을 스크롤 함	
find()	창안에 지정된 문자열이 있는지 확인. 있으면 true, 없으면 false	IE지원 안함
stop()	불러오기를 중지	IE지원 안함
print()	화면에 있는 내용을 프린터로 출력	
moveBy()	창을 상대적 좌표로 이동. 수평방향과 수직방향의 이동량을 픽셀로 지정.	
moveTo()	창을 절대적 좌표로 이동. 창의 왼쪽 상단 모서리를 기준으로 픽셀을 지정.	

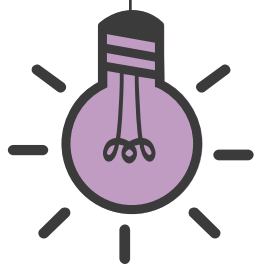
4. 웹브라우저와 Window 객체

4-5. window 객체 함수(2/2)

- **resize 함수로 현재 열려 있는 창의 크기 조절.**
- **window 객체에는 브라우저와 관련 된 함수 뿐만 아니라 순수 JavaScript에서 필요한 객체나 함수 존재.**
  - setTimeout() 함수와 setInterval() 함수로 함수를 특정 시간 후 혹은 특정 시간마다 호출가능.
  - eval() 함수는 문자열로 된 JavaScript 코드를 해석한 후 실행.

	설 명
resizeBy()	창의 크기를 상대적인 좌표로 재설정.
resizeTo()	창의 크기를 절대적인 좌표로 재설정. 창 크기를 픽셀로 지정.
scrollBy()	창을 상대적인 좌표로 스크롤. 창의 표시영역의 수평방향과 수직방향에 대해 픽셀로 지정.
scrollTo()	창을 절대적인 좌표를 스크롤 창의 왼쪽 상단 모서리를 기준으로 픽셀로 지정.
setTimeout()	지정한 밀리초 시간이 흐른 후에 함수 호출.
clearTimeout()	setTimeout 함수를 정지.
setInterval()	지정한 밀리초 주기마다 함수를 반복적으로 호출.
clearInterval()	setInterval 함수를 정지
eval()	문자열을 JavaScript 코드로 변환하여 실행.

05 HTML과 DOM



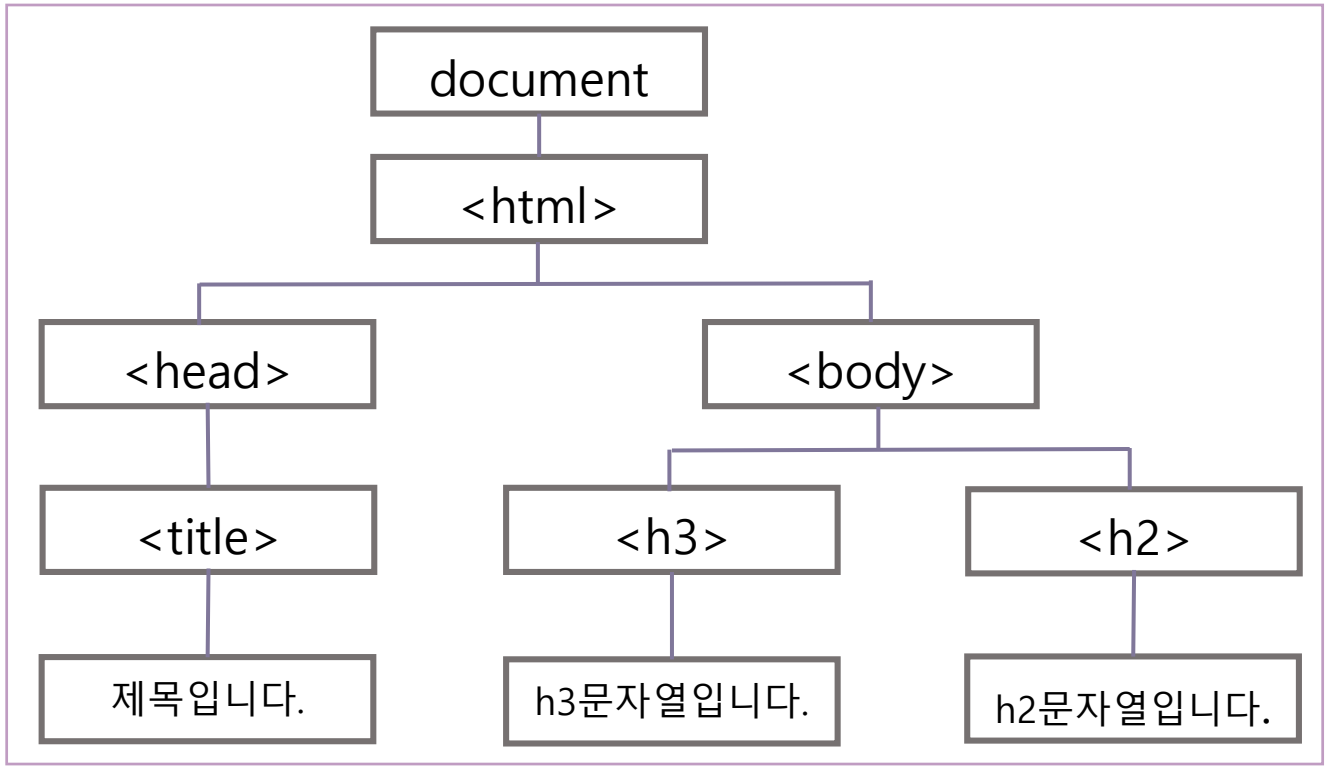
5. HTML과 DOM

5-1. DOM 개요(1/2)

- DOM(Document Object Model)은 HTML과 XML문서의 구조를 정의하고 API를 제공.
- DOM은 객체의 계층 구조로 HTML을 표현.
- HTML 계층 구조의 제일 위에는 document 노드가 있다.
- 그 아래로 HTML 태그나 요소(element)들을 표현하는 노드와 문자열을 표현하는 노드가 있다.

5-1.html

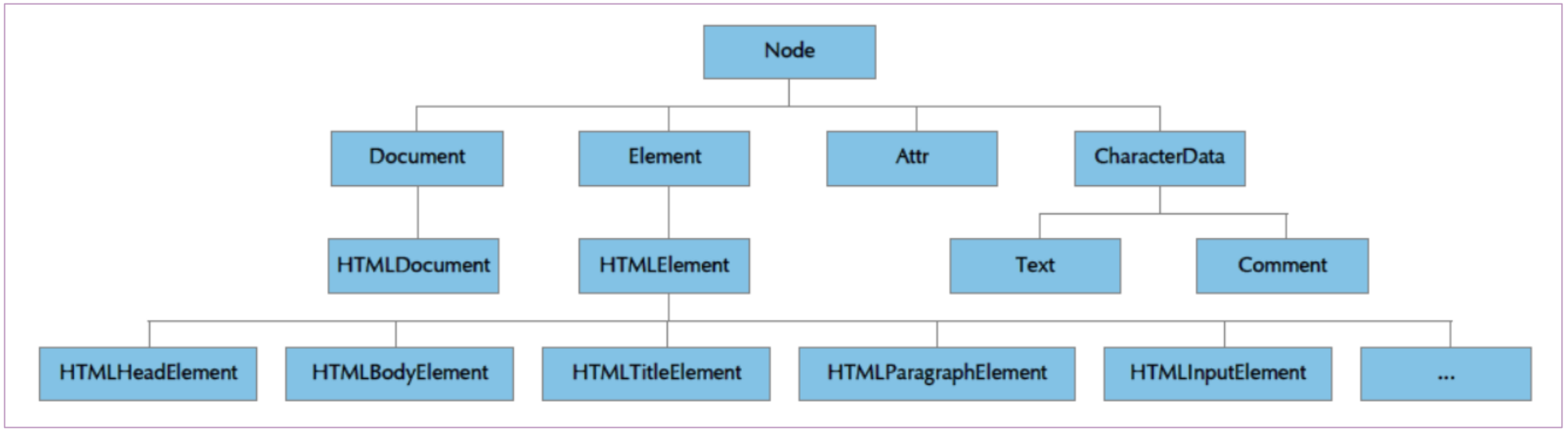
```
<!DOCTYPE html>
<html>
<head>
<title>제목입니다.</title>
</head>
<body>
    <h3>h3문자열입니다.</h3>
    <h2>h2문자열입니다.</h2>
</body>
</html>
```



5. HTML과 DOM

5-1. DOM 개요(2/2) - 문서계층구조

- Document는 HTML 또는 XML 문서를 표현.
- HTMLDocument는 HTML 문서와 요소만을 표현.
- HTMLElement의 하위 타입은 HTML 단일 요소나 요소 집합의 속성에 해당하는 JavaScript 프로퍼티를 정의.
- Comment 노드는 HTML이나 XML의 주석을 표현.

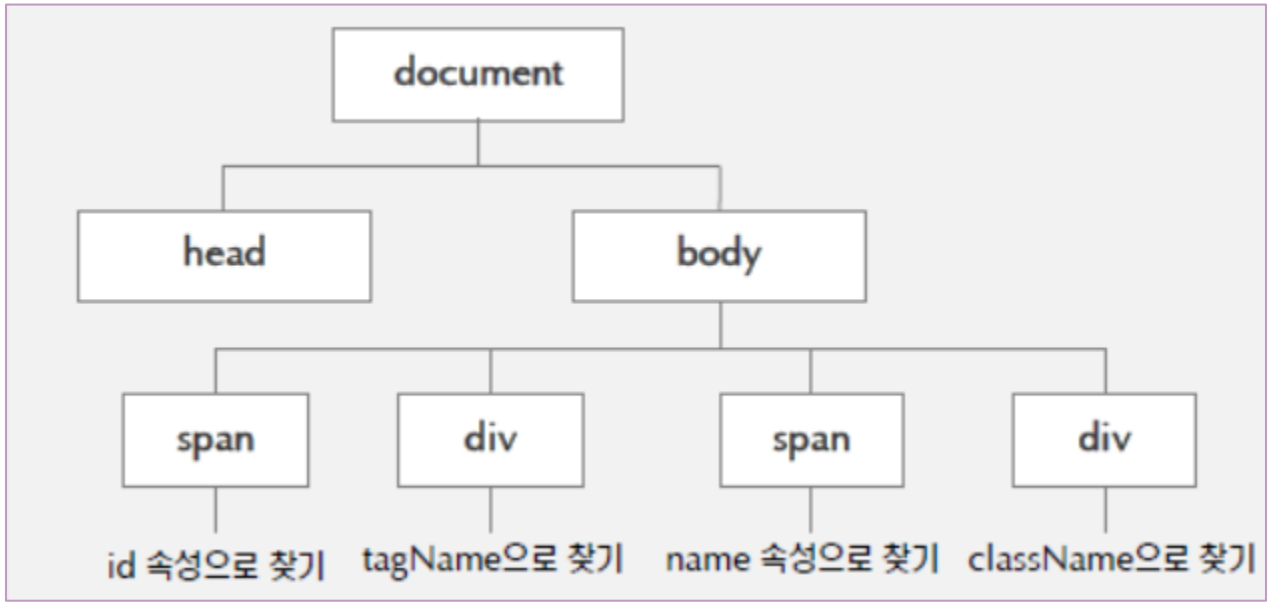


5. HTML과 DOM

5-2. DOM 제어(1/3)

- **웹 브라우저는 HTML(구조), CSS(표현), JavaScript(기능)의 협업을 통해 동적인 화면 표현.**
- **JavaScript를 사용하여 DOM을 검색하거나 제어할 수 있다.**
  - HTML 요소(element), HTML 요소의 속성, CSS 스타일 변경, 이벤트 제어 가능.
- **JavaScript는 id, name, HTML 태그이름, class등으로 문서 요소를 검색 가능.**

```
<body>
<span id="content1">content1</span>
<div>content2</div>
<span name="content3">content3</span>
<div class="content4">content4</div>

<script type="text/javascript">
// 1. id속성으로 찾기.
document.getElementById("content1");
// 2. tagName으로 찾기.
document.getElementsByTagName("div");
// 3. name 속성으로 찾기.
document.getElementsByName("content3");
// 4. className으로 찾기.
document.getElementsByClassName("content4");
</script>
</body>
```



5-2.html

5. HTML과 DOM

5-2. DOM 제어(2/3)

- DOM을 조회 할 때 id로 찾으면 가장 빨리 결과를 얻을 수 있다.
- HTML 표준 스펙 상 같은 HTML 문서 안에 중복되는 id가 있으면 안된다.
- 검색한 문서에 innerText나 innerHTML 속성으로 요소를 변경하거나 추가가능.

```
<body>
<div align="center">
    <div id="div1">초기 화면입니다.</div>
</div>

<script type="text/javascript">
var divEl = document.getElementById("div1");
divEl.innerHTML = '<font color="magenta" size="15">바뀐화면입니다.!!!!</font>';
</script>
</body>
```

5-3.html

실행화면

바뀐화면입니다.!!!!

5. HTML과 DOM

5-2. DOM 제어(3/3)

- JavaScript로 HTML 요소의 속성을 변경할 수 있다.
  - getAttribute('속성명');, setAttribute('속성명', '값');
- 문서 요소에서 style.color로 문서 요소(element)의 색을 변경 가능.
- className 속성으로 문서 요소의 CSS 클래스를 변경 가능.
- style.backgroundColor로 문서 요소의 배경색을 변경 가능.

```
<body>
<div align="center">
    <div id="div1">아이디가 div1인 div</div>

    <div>아이디가 없는 div</div>

    <script type="text/javascript">
    document.getElementById("div1").style.color = "cyan";
    </script>
</div>
</body>
```

5-4.html

실행화면

아이디가 div1인 div
아이디가 없는 div

5. HTML과 DOM

5-3. 이벤트 개요(1/6)

- **웹 페이지에서 여러 종류의 상호작용이 있을 때 마다 이벤트가 발생.**
- **사용자가 마우스를 클릭 하였을 때, 키보드를 눌렀을 때등과 같이 다양한 종류의 이벤트가 존재.**
- **JavaScript를 사용하여 DOM에서 발생하는 이벤트를 감지하여 이벤트에 대응하는 여러 작업 수행.**

이벤트 발생 종류
웹 페이지가 로딩되었을 때
페이지를 스크롤 했을 때
브라우저 창의 크기를 조절 했을 때
마우스를 클릭 했을때
키보드로 키를 입력 했을 때
Form이 Submit 되었을 때
Input의 내용이 변경 되었을 때
마우스를 움직여서 Element를 이동할 때

5. HTML과 DOM

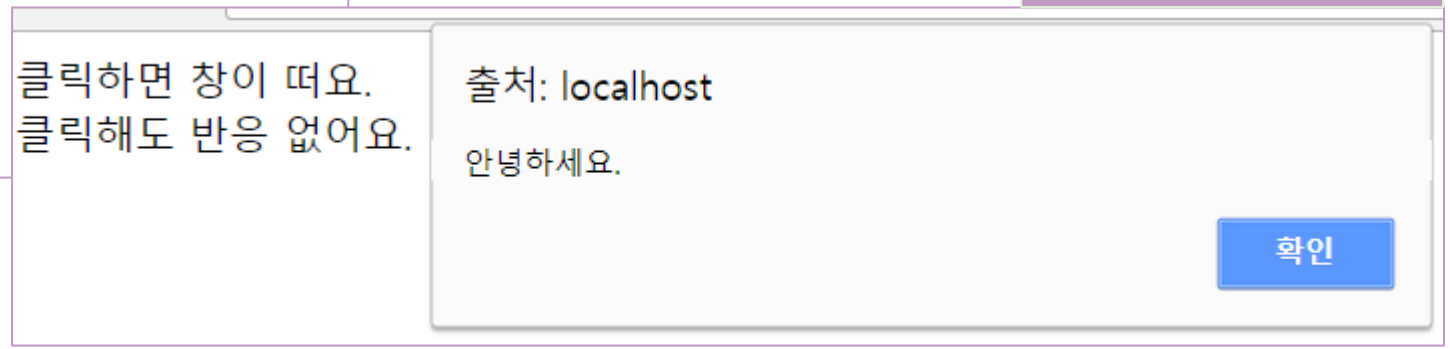
5-3. 이벤트 개요(2/6) - 예제

- Click 이벤트는 사용자가 마우스를 클릭 했을 때 발생.
- 특정 DOM 요소에 한하여 click 이벤트를 제어할 수 있다.
- “Click Me”라는 문자열을 담고 있는 요소 영역을 클릭할 경우에만 “Hello” 알림 창이 표시.

```
<body>
<div>
<div onclick="javacript:openHello();">클릭하면 창이 떠요.</div>
<div>클릭해도 반응 없어요.</div>
</div>
<script type="text/javascript">
function openHello() {
    alert("안녕하세요.");
}
</script>
</body>
</html>
```

5-5.html

실행화면



5. HTML과 DOM

5-3. 이벤트 개요(3/6) – 마우스 이벤트

- **웹 초기에는 load, click등 소수의 이벤트만 사용.**
- **마우스 이벤트는 웹 어플리케이션에서 가장 많이 사용하는 이벤트.**
- **마우스 이벤트 핸들러에 전달되는 이벤트 객체에는 마우스 위치와 버튼 상태 등의 정보를 담고 있다.**

이벤트	설명
onclick	마우스로 Element를 클릭 했을 때 발생.
ondbclick	마우스로 Element를 더블 클릭 했을 때 발생.
onmouseup	마우스로 Element에서 마우스 버튼을 올렸을 때 발생.
onmousedown	마우스로 Element에서 마우스 버튼을 눌렀을 때 발생.
onmouseenter	마우스를 움직여서 Element 밖에서 안으로 들어 올 때 발생.
onmouseleave	마우스를 움직여서 Element 안에서 밖으로 나갈 때 발생.

5. HTML과 DOM

5-3. 이벤트 개요(4/6) – 키보드 이벤트

- **키보드의 커서가 웹 브라우저에 나타나는 지점에서 키보드를 조작할 때 이벤트 발생.**
- **키보드 조작은 운영체제에 영향을 받으므로 특정 키가 이벤트 핸들러에게 전달되지 않을 수 있다.**
- **키보드 커서가 나타내는 요소가 없다면 document에서 이벤트가 발생.**

이벤트	설명
onkeypress	키보드가 눌려 졌을 때 발생. (키를 눌렀다가 떼면 발생)
onkeydown	키보드를 누르는 순간 발생.
onkeyup	키보드 키가 올려 질 때 발생.

5. HTML과 DOM

5-3. 이벤트 개요(5/6) – Frame 이벤트

- **Frame 관련 이벤트는 특정 DOM 문서에 관련된 이벤트가 아니라 Frame 자체에 대한 이벤트.**
- **Frame 이벤트 중에서는 load 이벤트가 가장 많이 사용.**
- **Load는 문서 및 자원들이 모두 웹 브라우저에 탑재되면 이벤트를 수행.**
- **unload는 사용자가 브라우저를 떠날 때 이벤트가 발생하지만, 사용자가 브라우저를 떠나는 것을 막을 수는 없다.**

이벤트	설명
onload	document, image, frame등이 모두 로딩 되었을 때 발생.
onabort	이미지 등의 내용을 로딩하는 도중 취소 등으로 중단 되었을 때 발생.
onerror	이미지 등의 내용을 로딩 중 오류가 발생 했을 때 발생.
onresize	document, element의 크기가 변경 되었을 경우 발생.
onscroll	document, element가 스크롤 되었을 때 발생.

5. HTML과 DOM

5-3. 이벤트 개요(6/6) – 폼(Form) 이벤트

- **Form 관련 이벤트는 웹 초기부터 지원되어 여러 웹 브라우저에서 가장 안정적으로 동작하는 이벤트.**
- **자주 사용되는 이벤트로 form이 전송될 때에는 submit 이벤트가 발생.**
- **Form을 초기화할 때는 reset 이벤트가 발생.**
- **Submit과 reset은 이벤트 핸들러에서 취소할 수 있다.**

이벤트	설명
onsubmit	form이 전송될 때 발생.
onreset	입력 form이 reset될 때 발생.
onblur	input과 같은 요소 등에서 입력 포커스가 다른 곳으로 이동할 때 발생.
onchange	입력 내용이 변경 되었을 때 발생.
onfocus	input과 같은 요소에 입력 포커스가 들어 올 때 발생.
onselect	input, textarea에 입력 값 중 일부가 마우스 등으로 선택될 때 발생.

5. HTML과 DOM

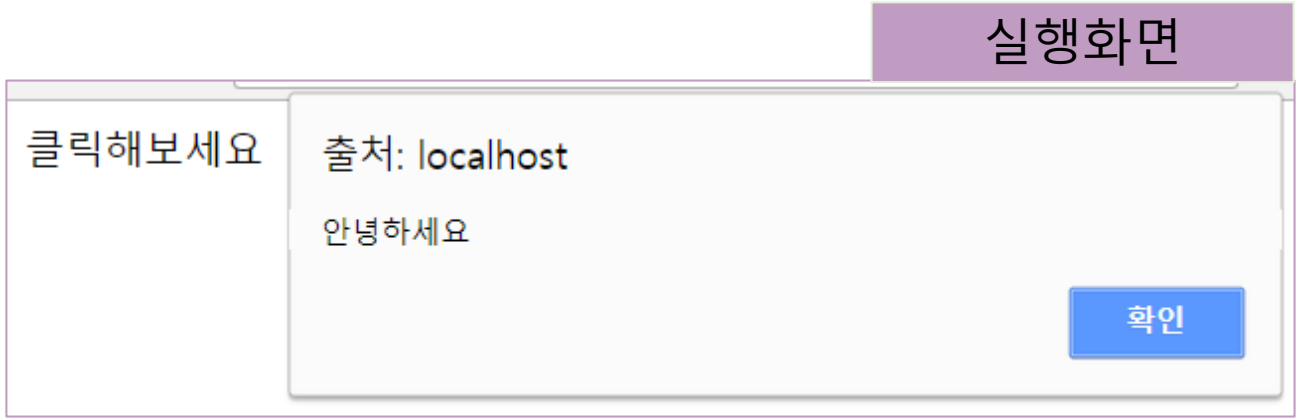
5-4. 이벤트 처리(1/4) – 등록(1/3)

- 이벤트를 감지하고 대응하는 작업을 등록하는 방법은 여러가지 제공.
- 어떤 이벤트를 처리할 작업을 등록하는 것을 '이벤트 핸들러(혹은 리스너)를 등록한다 ' 하고 표현.
- JavaScript의 초기에는 HTML 요소의 내부에서 직접 이벤트 핸들러를 등록.
- 이러한 방식은 HTML 코드를 JavaScript 코드가 침범한다는 문제가 있음.

```
<body>

    <div onclick="javascript:alert('안녕하세요');">클릭해보세요</div>

</body>
```

5-6.html

실행화면



5. HTML과 DOM

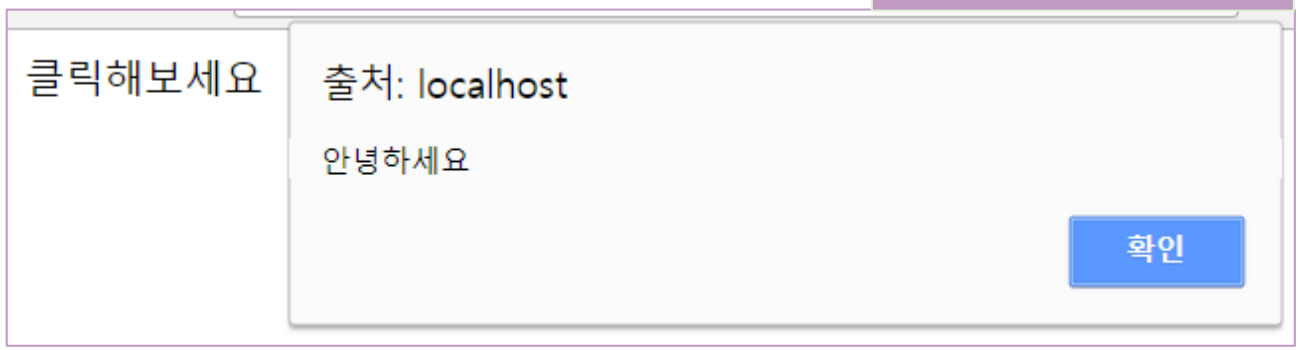
5-4. 이벤트 처리(2/4) – 등록(2/3)

- HTML에 직접 이벤트 핸들러를 등록하는 방법 대신에 JavaScript에서 이벤트 핸들러를 등록하는 방법이 있다.
- JavaScript에서 이벤트 핸들러를 등록함으로써 HTML코드와 JavaScript 코드를 분리할 수 있다.
- 이벤트 대상이 되는 특정 DOM을 선택하고 이벤트 핸들러를 등록.
- 'div1' 요소(element)에 클릭 이벤트가 발생하면 핸들러로 등록한 함수가 실행.

```
<body>
    <div id="div1">클릭해보세요</div>

    <script type="text/javascript">
    document.getElementById("div1").onclick = function() {
        alert('안녕하세요');
    };
    </script>
</body>
```

5-7.html

실행화면



5. HTML과 DOM

5-4. 이벤트 처리(3/4) – 등록(3/3)

- 2000년에 발표된 DOM 레벨2 이벤트 명세의 addEventListener(arg1, arg2, arg3)를 이용하여 좀더 세밀한 이벤트 제어가 가능.
- 전달 인자의 첫 번째에는 이벤트 이름, 두 번째에는 이벤트 핸들러, 세 번째에는 캡쳐링 여부를 사용.
- 첫 번째 전달인자의 이벤트 이름에는 'on'을 제거한 이벤트 이름을 사용.
- [주의!!] 이 방식은 Internet Explorer 9 이전 버전에서는 사용 불가.

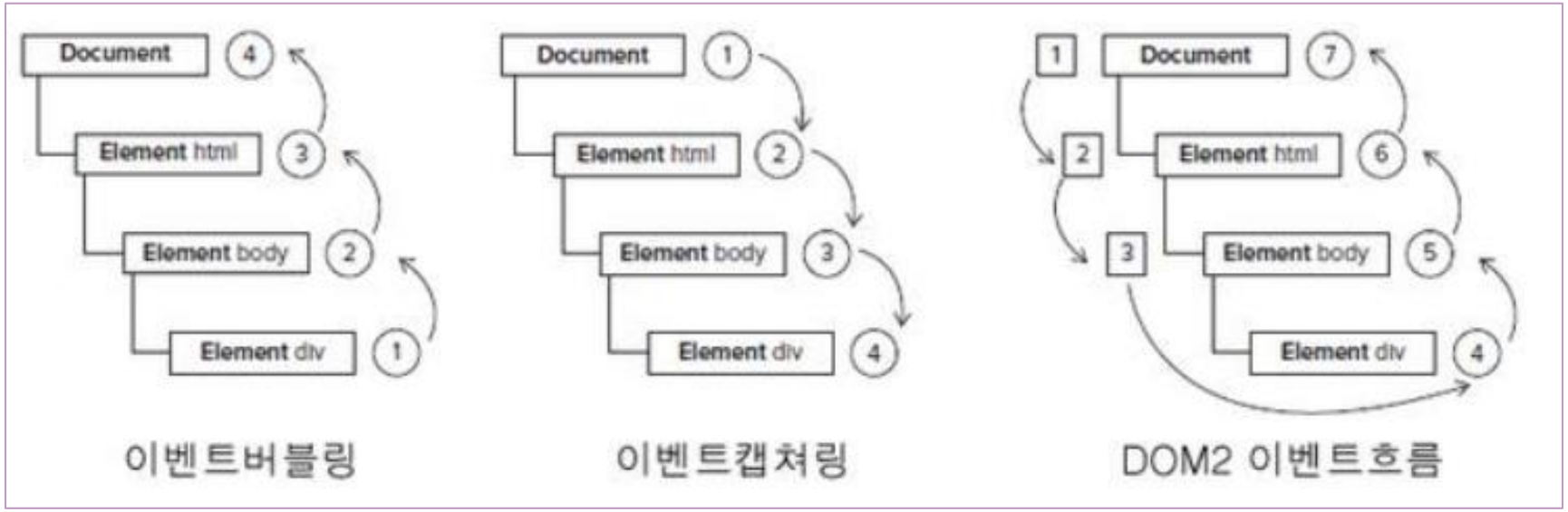
```
<body>
    <div id="div1">클릭하세요.</div>

    <script type="text/javascript">
    // DOM 레벨2 이벤트 명세 방법
    // 이벤트명에 on을 생략(Internet Exploler 9 이전버전 사용불가)
    document.getElementById("div1").addEventListener("click", openAlert, false);

    // Internet Exploler 9 이전 버전에서 사용하기위한 방법
    //document.getElementById("div1").attachEvent("onclick", openAlert);

    function openAlert() {
        alert("안녕하세요.");
    }
    </script>
</body>
```

5-8.html

5. HTML과 DOM

5-4. 이벤트 처리(4/4) – 버블링과 캡쳐링(1/2)

- **이벤트가 발생한 요소를 포함하는 부모 HTML로부터 이벤트 근원지인 자식요소까지 검사하는 것을 캡쳐링이라 한다.**
  - 이벤트 캡쳐링에서 캡쳐속성의 이벤트 핸들러가 등록되어 있으면 수행.
- **이벤트 발생 요소부터 요소를 포함하는 부모요소까지 올라가면서 이벤트를 검사하는 것을 이벤트 버블링이라 한다.**
  - 이벤트 버블링에서 버블속성의 이벤트 핸들러가 등록되어 있으면 수행.

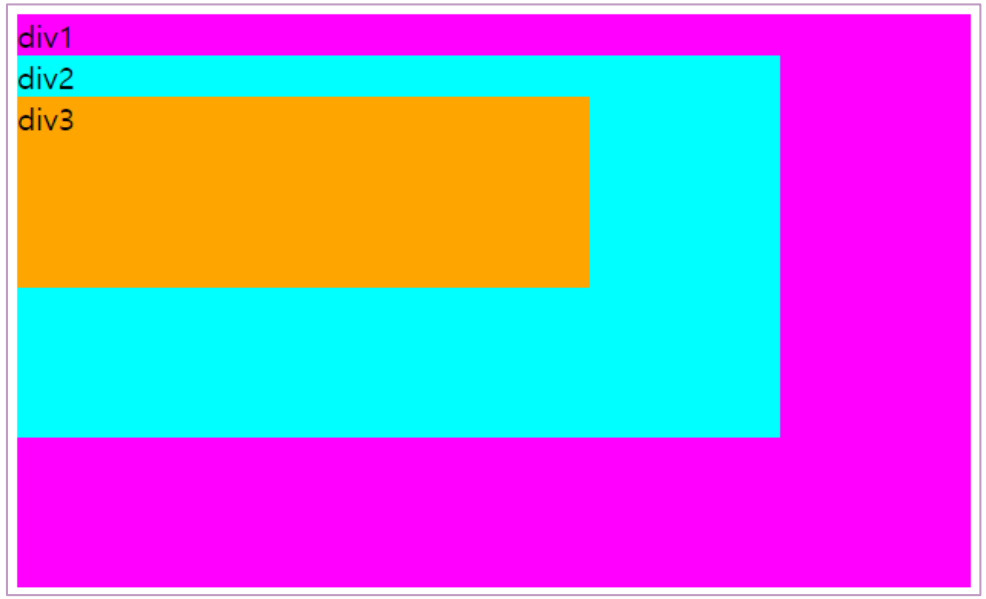


5. HTML과 DOM

5-4. 이벤트 처리(4/4) – 버블링과 캡쳐링(2/2)

- 함수의 세번째인자값이 true면 캡쳐링, false면 버블링
- div3을 클릭했을 경우 결과.
  - True(캡쳐링) : div1 > div2 > div3
  - false(버블링) : div3 > div2 > div1

```
<body>
<div id="div1" style="width:500px; height:300px; background:magenta;">
    div1
    <div id="div2" style="width:400px; height:200px; background:cyan;">
        div2
        <div id="div3" style="width:300px; height:100px; background:orange;">
            div3
        </div>
    </div>
</div>
<script type="text/javascript">
var div1 = document.getElementById("div1");
var div2 = document.getElementById("div2");
var div3 = document.getElementById("div3");

div1.addEventListener("click", function(e) {
    alert("div1");
}, true);

div2.addEventListener("click", function(e) {
    alert("div2");
}, true);

div3.addEventListener("click", function(e) {
    alert("div3");
}, true);
//true이면 캡쳐, false이면 버블
</script>
```



5-9.html

5. HTML과 DOM

5-5. 이벤트 활용

- 하나의 DOM 엘리먼트에 복수의 이벤트 핸들러를 등록할 수 있다.
- 마우스가 특정 DOM 엘리먼트 영역 안으로 들어온 경우 mouseenter 이벤트가 발생.
- 반대로 마우스가 특정 DOM 엘리먼트 영역 밖으로 나간 경우 mouseleave 이벤트가 발생.
- 태그에 mouseenter, mouseleave 2가지 이벤트 핸들러를 등록.

```
<body>
    <span id="span1" style="background:black;color:white;">span입니다.</span>

    <script type="text/javascript">
    var span1 = document.getElementById("span1");

    // mouseenter 이벤트 핸들러 등록
    span1.addEventListener("mouseenter", mouseIn, false);
    // mouseleave 이벤트 핸들러 등록
    span1.addEventListener("mouseleave", mouseOut, false);

    // mouseenter 핸들러 함수
    function mouseIn() {
        span1.style.background = "magenta";
    }
    // mouseleave 핸들러 함수
    function mouseOut() {
        span1.style.background = "orange";
    }
    </script>
</body>
```

초기화면

mouseenter

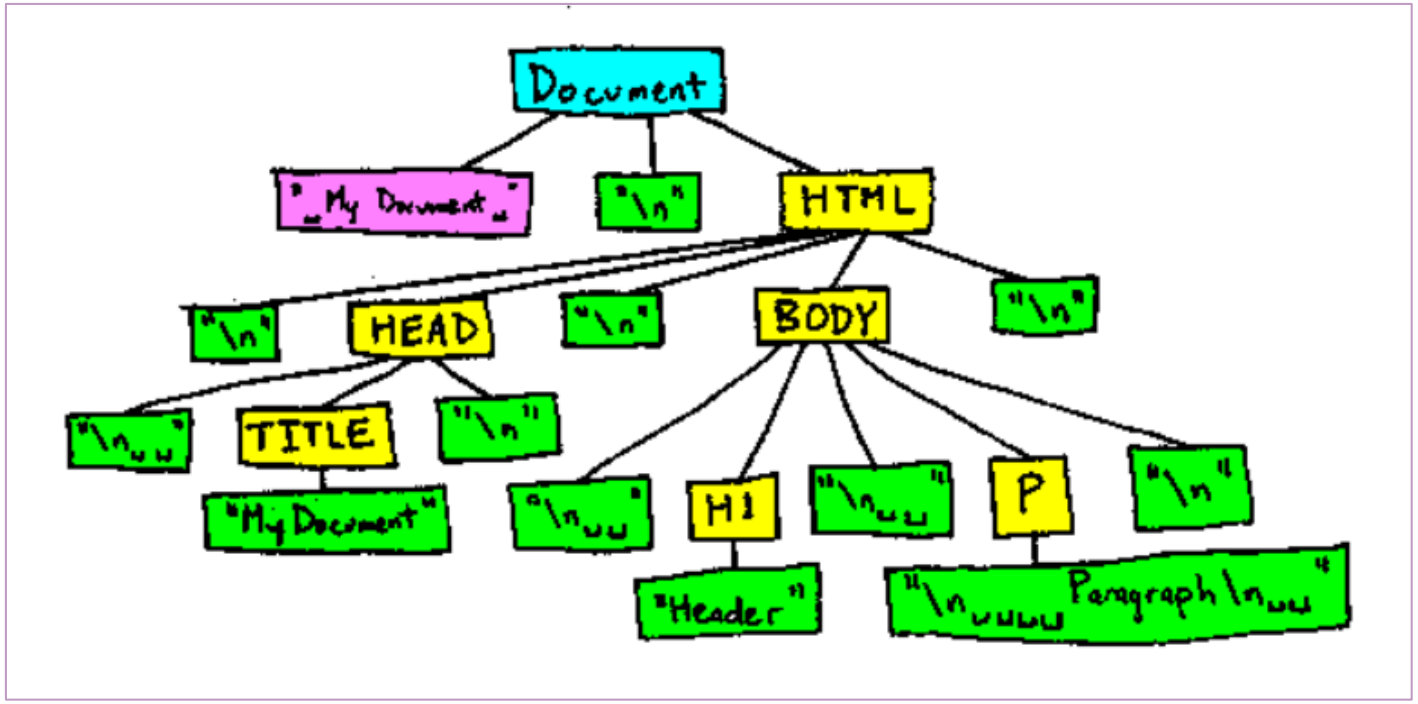
mouseleave

5-10.html

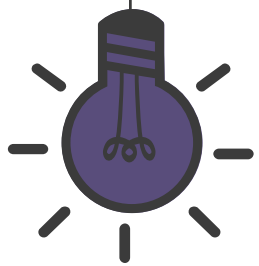
5. HTML과 DOM

5-6. 요약

- DOM은 HTML 문서의 내용을 조작할 수 있는 API로 HTML을 계층구조 형식의 객체로 표현.
- DOM으로 HTML 문서의 검색과 조작(추가, 수정, 삭제)을 할 수 있다.
- DOM에서 발생되는 이벤트에 대한 핸들러(리스너)를 등록하여 특정 이벤트에 대응하는 작업을 할 수 있다.
- 핸들러 등록은 HTML태그의 on 속성에 명시하는 방법과 JavaScript에서 DOM 검색 후 등록하는 방법이 있다.



06 JavaScript 활용



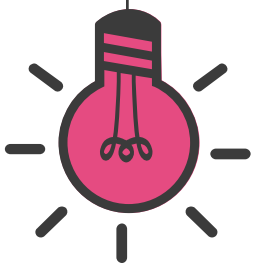
6. JavaScript 활용

6-1.

6. JavaScript 활용

6-1.

07 JavaScript 기타

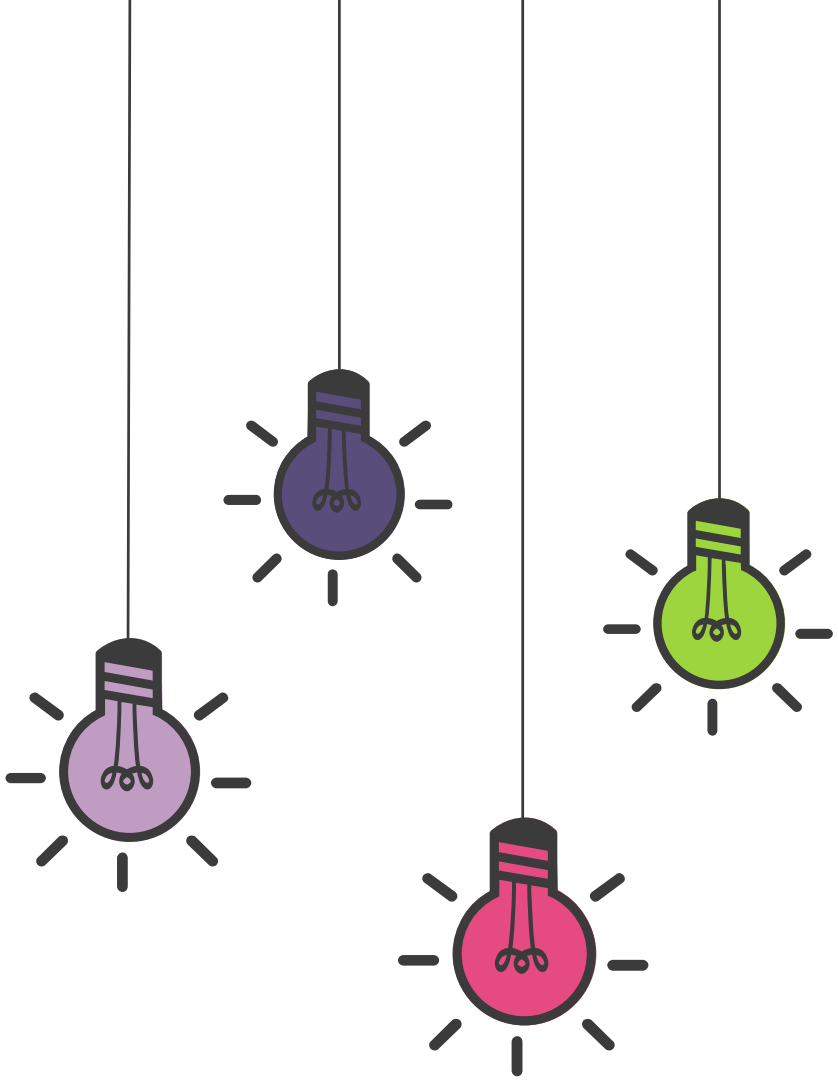


7. JavaScript 기타

7-1.

7. JavaScript 기타

7-1.



감사합니다

THANK YOU FOR WATCHING

안 효 인
troment@nate.com