



JavaScript

❖ Javascript

- ◆ 크로스-플랫폼, 객체지향 스크립트 언어
- ◆ 작고 가벼운 언어
- ◆ 웹브라우저(호스트 환경) 내에서 주로 사용 - JavaScript는 프로그램 제어를 제공
- ◆ 다른 응용 프로그램의 내장 객체에도 접근할 수 있는 기능을 가지고 있다.
- ◆ Node.js와 같은 런타임 환경과 같이 서버 사이드 네트워크 프로그래밍에도 사용



❖ Javascript

- ◆ 넷스케이프 커뮤니케이션즈 코퍼레이션의 브렌던 아이크(Brendan Eich)가 처음에는 모카(Mocha)라는 이름으로, 나중에는 라이브스크립트(LiveScript)라는 이름으로 개발하였으며, 최종적으로 자바스크립트(자바와 닮았다해서...)가 되었다.
자바스크립트가 썬 마이크로시스템즈의 **자바와 구문(syntax)이 유사한 점도 있지만**, 이는 사실 **두 언어 모두 C 언어의 기본 구문을 바탕**했기 때문이고,
자바와 자바스크립트는 직접적인 관련성이 없다.

❖ Java와 비교한 Javascript

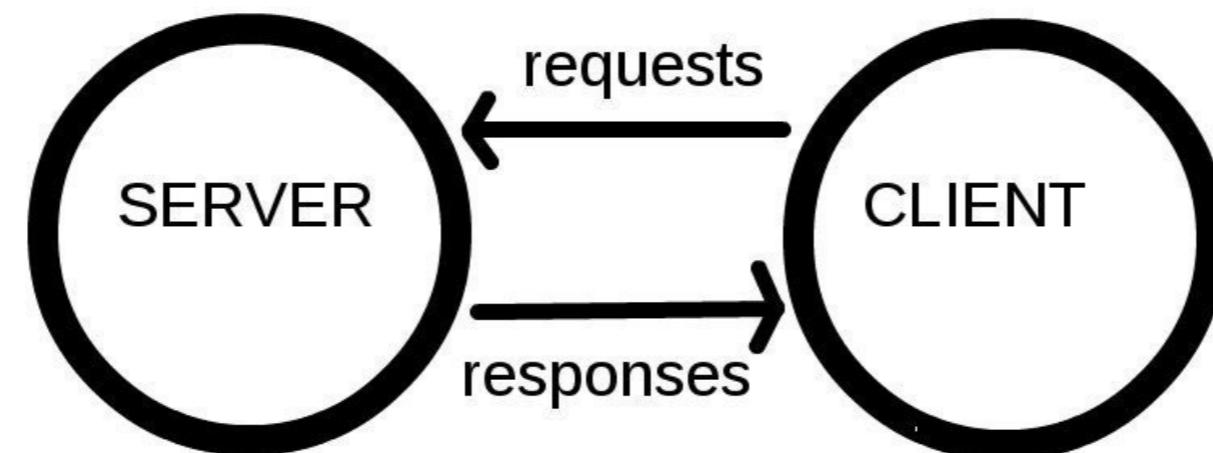
JavaScript	Java
객체 지향. 객체의 형 간에 차이 없음. 프로토타입 메커니즘을 통한 상속, 속성과 메서드는 어떤 객체든 동적으로 추가될 수 있음.	클래스 기반. 객체는 클래스 계층구조를 통한 모든 상속과 함께 클래스와 인스턴스로 나뉨. 클래스와 인스턴스는 동적으로 추가된 속성이나 메소드를 가질 수 없음.
변수 자료형이 선언되지 않음 (동적 형지정, dynamic typing).	변수 자료형은 반드시 선언되어야 함 (정적 형지정, static typing).
하드 디스크에 자동으로 작성 불가.	하드 디스크에 자동으로 작성 가능.

- ❖ 최근 버전 : ECMAScript(자바스크립트의 표준화된 버전) 2016 (2016년 6월 17일)
- ❖ 브라우저마다 지원되는 버전이 다르며, 가장 범용적으로 지원되는 버전은 1.5이다.
- ❖ 상표는 오라클 소유 /
넷스케이프 커뮤니케이션스가 발명, 구현한 기술 및
모질라 재단과 같은 독립 기관의 라이선스 하에 사용된다.
- ❖ 유용한 링크
 - <https://developer.mozilla.org/ko/docs/Web/JavaScript>
 - <https://www.w3schools.com/js/default.asp>

❖ 웹의 동작 원리

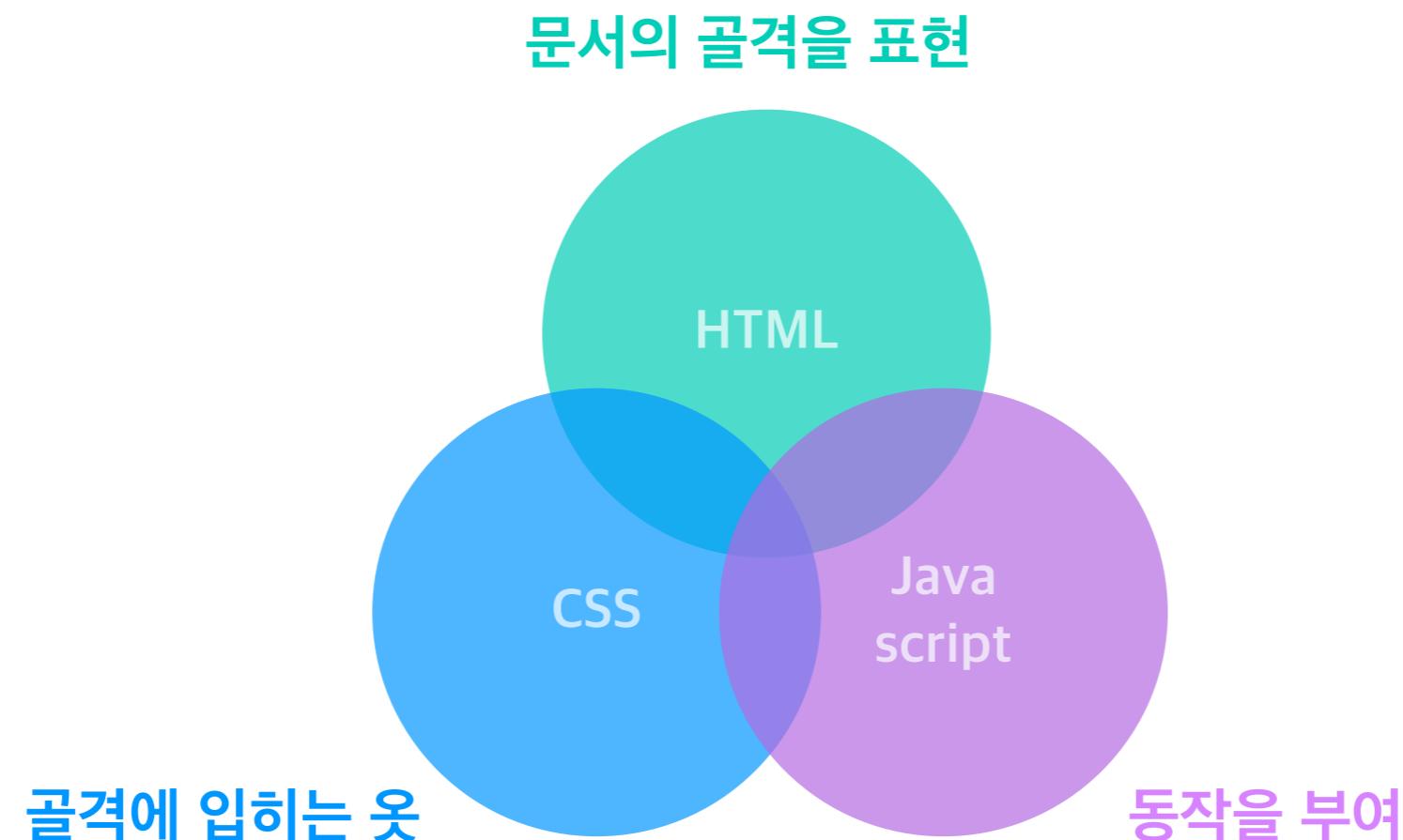
◆ 요청(클라이언트)과 응답(서버)의 과정

- ❖ 클라이언트(사용자) : 서버에 요청하는 쪽
- ❖ 서 버(제공자) : 요청에 응답하는 쪽



❖ 자바스크립트의 역할

- * 웹 문서를 동적으로 제어하기 위해 고안된 프로그래밍 언어



❖ 자바스크립트의 역할

- * 요소의 추가 및 삭제
- * CSS 및 HTML 요소의 스타일 변경
- * 사용자와의 상호작용
- * 폼의 유효성 검증
- * 마우스와 키보드 이벤트에 대한 스크립트 실행
- * 웹 브라우저 제어 및 쿠키 등의 설정과 조회
- * AJAX 기술을 이용한 웹 서버와의 통신

❖ Javascript 특징

- * 인터프리터 언어 - 에러가 발생하면 에러가 발생한 다음 줄 부터는 구문 분석 않는다.
- * 클라이언트 스크립트 언어 - 서버의 부하를 줄일 수 있다.
- * 객체 기반 언어
- * 공개된 언어
- * 다양한 라이브러리 - 예] jQuery, NODE.js

❖ Javascript 사용 이유

- * 서버의 부하를 줄일 수 있다.
- * 동적인 사이트 구현
- * 인터랙티브한 사이트 구현
- * 다양한 라이브러리 언어 활용
- * HTML5 API 기반 언어

❖ Javascript 작성 방법

- 대소문자 구분하여 작성
- 문장은 세미콜론(;)으로 구분

var age=25
document.write("당신의 나이는 " + age + "입니다.")

바른 예

var age=25;
document.write("당신의 나이는 " + age + "입니다.");

var age=25; document.write("당신의 나이는 " + age + "입니다.");

잘못된 예

var age=25 document.write("당신의 나이는 " + age + "입니다.")

- 큰따옴표(" ")와 작은따옴표(' ')를 구분하여 사용

document.write("<div style='color: red;'> 자바스크립트 학습 </div>");

바른 예

document.write(<div style="color: red;"> 자바스크립트 학습 </div>);

잘못된 예

document.write(<div style="color: red;"> 자바스크립트 학습 </div>)

❖ Javascript 포함 방법

- HTML 문서 내부에 코드를 작성하는 방법

- ❖ <head> 태그 또는 <body> 태그 내에 코드 작성

```
<head>
    <meta charset="utf-8"/>
    <title>자바스크립트 예제</title>
    <script>
        // 자바스크립트 코드 작성
    </script>
</head>
<body>
    <script>
        // 자바스크립트 코드 작성
    </script>
</body>
```

- ❖ HTML 태그 안에 속성값으로 정의

```
<button type="button" onclick="alert('자바스크립트')">버튼 클릭</button>
```



❖ Javascript 포함 방법

- ◆ 자바스크립트 코드의 실행 순서 살펴보기

js_test01.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>자바스크립트 예제</title>
    <script>
        var num=0;
        document.write("head 태그 내 실행 순서 :" + num + "<br/>");
    </script>
    <script>
        var num=1;
        document.write("head 태그 내 실행 순서 :" + num + "<br/>");
    </script>
</head>
<body>
    <script>
        var num=2;
        document.write("body 태그 내 실행 순서 :" + num + "<br/>");
    </script>
    <script>
        var num = 3;
        document.write("body 태그 내 실행 순서 :" + num + "<br/>");
    </script>
</body>
</html>
```

❖ Javascript 포함 방법

- 별도로 작성한 후 HTML 문서에서 참조하는 방법

- 외부 자바스크립트 파일을 만든 후 HTML 문서의 <script> 태그에 src 속성을 추가하여 참조

위치	src 속성값
HTML 문서와 같은 디렉터리에 있는 경우	<script src="myscript.js"></script>
HTML 문서와 다른 디렉터리에 있는 경우	<script src="./ejs/myscript.js"></script>
HTML 문서와 다른 서버 디렉터리에 있는 경우	<script src="http://www.hanbit.co.kr/jsfile/myscript.js"></script>

* 자바스크립트 파일을 외부에서 작성했을 때의 장점

- 자바스크립트 파일을 HTML 문서와 분리하여 관리할 수 있음
- 자바스크립트 코드를 관리, 유지보수, 디버깅하기 쉬움
- 자바스크립트 코드의 보안성과 안전성을 높일 수 있음

❖ Javascript 포함 방법

◆ 외부 자바스크립트 문서 작성 후 참조하기

./js/test_js01.js

```
var age=23;  
/* 문자에 스타일 속성 적용 */  
document.write("<div style='color: red; font-size: 24px;'>외부 자바스크립트 파일</div>");  
document.write("당신의 나이는 " + age + "입니다.");
```

js_test02.html

```
<!DOCTYPE html>  
<html>  
<head>  
  <meta charset="utf-8"/>  
  <script src=".js/test_js01.js"></script>  
</head>  
<body>  
  <p>  
    <!-- 버튼을 클릭하면 메시지 창 출력 -->  
    <button type="button" onclick="alert('외부 자바스크립트 파일')">버튼 클릭</button>  
  </p>  
</body>  
</html>
```



❖ Javascript 포함 방법

◆ 내포 관계인 자바스크립트 파일 참조하기

```
document.write("test_js02.js");
document.write("<div style='color: red; font-size: 24px;'>외부 자바스크립트 파일</div>");
document.write("<script src='./js/test_js3.js'> </script>");
```

./js/test_js02.js

```
document.write("test_js03.js는 test_js02.js에 포함");
document.write("<div style='color: blue; font-size: 20px;'>외부 자바스크립트 파일</div>");
document.write("<script src='./ejavascript/test_js04.js'> </script>");
```

./js/test_js03.js

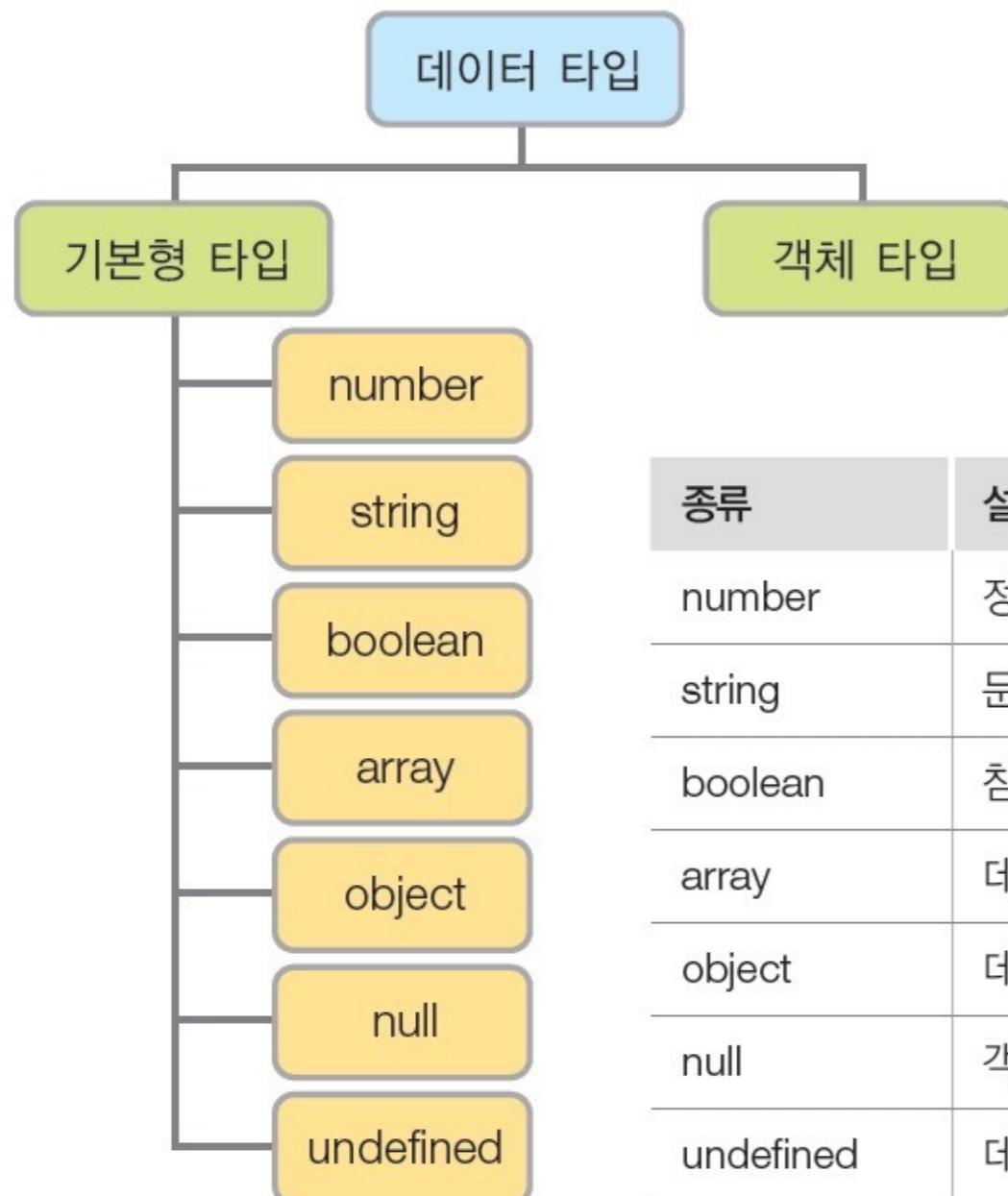
```
document.write("test_js04.js는 test_js03.js에 포함");
document.write("<div style='color: green; font-size: 16px;'>외부 자바스크립트 파일</div>");
alert('Nested Script File');
```

./js/test_js04.js

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
</head>
<body>
  <script src="./ejavascript/njs1.js"> </script>
</body>
</html>
```

js_test03.html

❖ Javascript 데이터 타입



기본형 타입의 종류

종류	설명	사용 예
number	정수 혹은 실수	100, 10.5, 10e+3
string	문자 혹은 문자열	“홍길동”, ‘홍길동’
boolean	참 혹은 거짓	true, false
array	데이터의 집합(배열, 객체로 취급)	[“서울”, “부산”, “인천”]
object	데이터 속성과 값으로 이루어진 집합	{name: ‘홍길동’, age: 25}
null	객체 값이 없음	null
undefined	데이터 값이 정해지지 않음	undefined

❖ Javascript 포함 방법

- ◆ **typeof** 연산자를 사용하여 데이터 타입 확인하기

js_datatype.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>Javascript Data Type</title>
</head>
<body>
    <script>
        var num;      // 변수 값이 없음
        var obj=null; // 객체 변수 값이 없음
        document.write(typeof 100+"<br>");
        document.write(typeof 10.5+"<br>");
        document.write(typeof "홍길동)+"<br>");
        document.write(typeof true+"<br>");
        document.write(typeof [1,2,3)+"<br>");
        document.write(typeof {name:'홍길동', age:25}+"<br>");
        document.write(typeof num+"<br>");
        document.write(typeof obj+"<br>");
    </script>
</body>
</html>
```

❖ Javascript 변수명 규칙

● 변수명 작성 규칙

- 문자, 밑줄(_), 달러 기호(\$)로 시작
- 대소문자 구별('변수 A'와 '변수 a'는 서로 다른 변수)
- 한글은 사용 가능하나 영문자 사용 권장
- 자바스크립트에서 정한 예약어는 변수명으로 사용 불가능

abstract	Arguments	boolean	break	byte
case	catch	char	class	const
continue	debugger	default	delete	do
double	else	enum	eval	export
extends	false	final	finally	float
for	function	goto	if	implements
import	in	instanceof	int	interface
let	long	native	new	null
package	private	protected	public	return
short	static	super	switch	synchronized
this	throw	throws	transient	true
try	typeof	var	void	volatile
while	with	yield		

❖ Javascript 예약어

❖ 변수 사용 방법

● 변수 사용 예

- var x; // 변수 x 선언
- var y=10; // 변수 y 선언 및 초기값 할당
- var x=y; // 변수 y의 값을 변수 x에 저장
- var a, b, c; // 변수 a, b, c 선언
- var a=10, b=11, c=12; // 변수 a, b, c 선언 및 각각 다른 초기값 할당
- var a=b=c=10; // 변수 a, b, c 선언 및 같은 초기값 할당
- var name="홍길동", age=25; // 변수 name, age 선언 및 각각 다른 초기값 할당
- var total=a+b+c; // 변수 a, b, c 값을 더한 결과를 변수 total에 저장

❖ 변수 사용 방법

● 변수 사용 시 문법적으로 오류가 발생한 사례

- var 7num=100; // 숫자로 시작하는 변수명 잘못 사용
- var &num=100; // 특수 문자로 시작하는 변수명 잘못 사용
- var true=1; // 예약어를 변수명으로 잘못 사용
- var 10=x; // 좌변에 상수값 잘못 선언
- var a+b=20; // 좌변에 연산식 잘못 선언
- var “홍길동”=name; // 좌변에 문자열값 잘못 선언
- var get Number=100; // 변수명 사이에 공백(space) 잘못 선언
- var a, b, c=100; // 콤마로 구분한 변수명 잘못 선언

❖ Javascript 포함 방법

◆ 변수의 재선언후 데이터 타입

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
</head>
<body>
    <script>
        stdName="홍길동"; // var 키워드 생략
        comGrade=96;     // var 키워드 생략
        var stdName;     // 변수명 재선언
        var comGrade;    // 변수명 재선언
        document.write("학생 이름 :" + stdName + "<br>");
        document.write("컴퓨터 점수 :" + comGrade + "<br>");
    </script>
</body>
</html>
```

js_data_retype02.html

❖ Javascript 전역변수 & 지역변수

- 전역 변수
 - 코드 내 어느 위치에서든 선언하여 전 영역에서 사용할 수 있는 변수
- 지역 변수
 - 변수가 선언된 해당 블록에서 선언하여 범위 내에서만 유효하게 사용할 수 있는 변수

```
<!DOCTYPE html>                                js_var01.html
<html>
<head>
    <meta charset="UTF-8" />
    <title>variable</title>
    <script type="text/javascript">
        var globValue1;
        globValue2;
        function test01() {
            var locValue;          // 지역변수
            globValue = 999;       // 함수 내부에서 var 생략할 경우 자동 전역변수
            locValue = 10;         // 지역변수
        }
        test01();   // 함수 실행
        alert(globValue);
    </script>
</head>
<body>
</body>
</html>
```

❖ 변수 호이스팅 (hoisting)

- **JavaScript 변수의 특이한 점은 예외를 받지 않고도, 나중에 선언된 변수를 참조할 수 있다는 것**
 - JavaScript 변수가 어떤 의미에서 "끌어올려지거"나 함수나 문의 최상단으로 올려지는 것 (Hoisting)
 - 끌어올려진 변수는 `undefined` 값을 반환
 - * 이 변수를 사용 혹은 참조한 후에 선언 및 초기화하더라도, 여전히 `undefined`를 반환

js_var02.html

```
console.log(x === undefined); // logs "true"
var x = 3;
```

```
// undefined 값을 반환함.
var myvar = "my value";
(function() {
  console.log(myvar); // undefined
  var myvar = "local value";
})();
```



```
var x;
console.log(x === undefined); // logs "true"
x = 3;
```

```
var myvar = "my value";
(function() {
  var myvar;
  console.log(myvar); // undefined
  myvar = "local value";
})();
```

- * 호이스팅 때문에, 함수 내의 모든 `var` 문은 가능한 함수 상단 근처에 두는 것이 좋다.
이 방법은 코드를 더욱 명확하게 만들어줍니다.

❖ 전역 변수와 지역 변수 이해

```
<script>
    function getGrade() { // 함수 정의
        var kor=95;      // 지역 변수
    }
    var kor=100;        // 전역 변수
    getGrade();         // 함수 호출
    document.write("국어 점수 :" + kor + "<br>");
</script>
```

js_var03.html

```
<script>
    function getGrade() { // 함수 정의
        kor=95;          // 자동 전역 변수
    }
    var kor=100;        // 전역 변수
    getGrade();         // 함수 호출
    document.write("국어 점수 :" + kor + "<br>");
</script>
```

js_var04.html

❖ 전역 변수와 지역 변수 이해

```
<script>
    function getGrade() { // 함수 정의
        var kor=95;      // 지역 변수
    }
    getGrade();          // 함수 호출
    document.write("지역 변수 값은 함수 외부에서 사용할 수 없습니다.<br>");
    document.write("국어 점수 :" + kor + "<br>");
</script>
```

js_var05.html

```
<script>
    function getGrade() { // 함수 정의
        var kor=95;      // 지역 변수
        return kor;
    }
    getKor=getGrade(); // 함수 호출
    document.write("국어 점수 :" + getKor + "<br>");
</script>
```

js_var06.html

❖ 전역 변수

- 전역 변수는 **global** 객체의 속성(**property**)

- 웹 페이지에서 global 객체는 window 이므로,
`windows.variable` 구문을 통해 전역 변수를 설정하고 접근할 수 있다.
- `window` 혹은 `frame`의 이름을 지정하여
한 `window` 혹은 `frame`에서 다른 `window` 혹은 `frame`에 선언된 전역 변수에 접근할 수 있다.

예] `memberNo`라는 변수가 문서에 선언된 경우,
`iframe`에서 `parent.memberNo`로 이 변수를 참조할 수 있다.



❖ 전역 변수

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>variable</title>
<link rel="stylesheet" type="text/css" href="./CSS/style.css" />
<script type="text/javascript">
    var memberNo = 777;
</script>
</head>
<body>
    <div class="card">
        <h3>여기는 메인 페이지</h3>
    </div>
    <iframe class="card" src="frame01.html" />
</body>
```

member.html

```
<!DOCTYPE html>
<html lang="UTF-8">
<head>
<meta charset="UTF-8" />
<title>variable</title>
<script type="text/javascript">
    var mNo = parent.memberNo;
    document.write("여기는 iframe_page : 회원번호는 \"" + mNo + "\"");
</script>
<style type="text/css">
    body { border: 0px; text-align: center; }
</style>
</head>
<body>
</body>
```

frame01.html

```
html, body {
    margin: 0px;
    text-align: center;
}
.card {
    box-shadow: 0px 0px 7px gray;
    border: 1px solid gray;
    width: 330px;
}
div {
    background-color: ivory;
    text-align: center;
    height: 50px;
    margin: 20px auto 10px;
}
h3 {
    line-height: 15px;
}
iframe {
    background-color: yellow;
    margin-top: 0px;
    margin-bottom: 0px;
}
```

./CSS/style.css

❖ 연산자

❖ 피연산자에게 연산 명령을 내리기 위해 사용하는 기호

❖ 연산자의 종류

연산자	기호
문자열 연산자	+(문자열 연결)
산술 연산자	++(증가 연산), --(감소 연산), *(곱셈), /(나눗셈), %(나머지), +(덧셈), -(빼셈)
비교 연산자	<(작다), <=(작거나 같다), >(크다), >=(크거나 같다), ==(값이 같다), !=(값이 다르다), ===(값과 타입 모두 같다), !==(값 또는 타입이 다르다)
논리 연산자	&(비트 AND), (비트 OR), ^(비트 XOR), &&(논리 AND), (논리 OR)
조건 연산자	(판단) ? true : false;
대입 연산자	=, +=, -=, *=, /=, %=, <<=, >>=, >>>=, &=, =, ^=

❖ 문자열 연산자

- ‘+’ 기호를 사용하여 문자열을 연결

```
var st = "Hello" + "Javascript"; // 연산결과 "Hello Javascript"가 출력
```

```
var st = "100" + 10; // 10010 출력  
var st = 100 + 10; // 110 출력
```

❖ 산술 연산자

- 사칙 연산을 수행
- 종류 : 더하기(+), 빼기(-), 곱하기(*), 나누기(/), 나머지(%), 증감(++) , 감소(--)
 - 나머지(%) : 나눗셈 결과 나머지 값을 구함
 - 증가(++) : 변수값을 증가시킴
 - 감소(--) : 변수값을 감소시킴

❖ 비교 연산자

- 두 피연산자의 값을 비교하여 참(true) 또는 거짓(false) 값을 반환

비교 연산자	설명	사용 예	결과
<code>==</code>	값이 같은지 비교한다.	<code>x=="5"</code>	true
<code>====</code>	값과 타입이 같은지 비교한다.	<code>x===== "5"</code>	false
<code>!=</code>	값이 다른지 비교한다.	<code>x!="5"</code>	false
<code>!==</code>	값 또는 타입이 다른지 비교한다.	<code>x!=="5"</code>	true

❖ 논리 연산자

* 일반 논리 연산자

논리곱(&&)	두 개의 피연산자 값이 모두 참일 때만 참이고, 하나라도 거짓이면 거짓
논리합()	두 개의 피연산자 값 중 하나라도 참이면 참이고, 모두 거짓이면 거짓
논리 부정(!)	피연산자 값이 참이면 거짓, 거짓이면 참

❖ 대입 연산자

* '=' 기호를 사용하여 값이나 변수를 할당

❖ 조건(삼항) 연산자

- * 조건식을 판별하여 참이냐 거짓이냐에 따라 다음 문장을 선택적으로 실행

```
조건 ? 값1 : 값2
```

❖ 단항 연산자

- * **delete** : 객체, 객체의 속성 또는 배열의 특정한 위치에 있는 객체를 삭제
- * **typeof** : 데이터의 자료형 반환
- * **instanceof** : 명시된 객체가 명시된 객체형인 경우 true를 반환

❖ this

- * 현재 객체를 참조하는 데 **this** 키워드를 사용
: 일반적으로, **this**는 함수에서 호출하는 객체를 참조

참고

◆ 자바스크립트 실행결과를 확인하는 방법

- * **alert();** > 결과를 대화 상자를 이용해서 출력
- * **document.write();** > 결과를 화면에 직접 출력
- * **요소.innerHTML = 출력문장 ;** > 결과를 요소에 출력
- * **console.log();** > 웹브라우저의 콘솔창에 출력

◆ HTML 문서 안의 특정 요소를 선택하는 방법

- * **document.getElementById("아이디이름");** > 아이디로 선택하는 방법
- * **document.getElementsByName("name이름");** > 네임으로 선택하는 방법
- * **document.getElementsByTagName("태그이름") ;** > 태그로 선택하는 방법
- * **document.getElementsByClassName("클래스이름");** > 클래스로 선택하는 방법
- * **document.querySelectorAll("요소.클래스");** > 특정요소의 클래스로 선택

★ 아이디로 선택하는 방법 이외의 방법들은 모두 자동 배열이 된다.

❖ HTML 문서안의 특정요소 선택

```
js_select.html
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>innerHTML</title>
</head>
<style type="text/css">
    html, body {
        margin: 0px;
        text-align: center;
    }
</style>
<body>
    <h3 class="t2" id="test"></h3>
    <div class="test"></div>
    <h1 name="test"></h1>
    <p class="t3"></p>
</body>
<script type="text/javascript">
    var no = prompt("문자를 입력하세요");
    document.getElementById("test").innerHTML = no;
    //document.getElementsByClassName("t2")[0].innerHTML = no;
    //document.getElementsByName("test")[0].innerHTML = no;
    //document.getElementsByTagName("h3")[0].innerHTML = no;
    //document.querySelectorAll("p.t3")[0].innerHTML = no;
</script>
</html>
```

❖ 제어문

- ◆ 프로그램의 실행 과정을 제어하기 위해 사용하는 구문문자, 밑줄(_), 달러 기호(\$)로 시작

유형	설명	구조
조건문	조건에 따라 다음 문장을 선택적으로 실행한다.	<ul style="list-style-type: none">• If문• if~else문• 다중 if~else문• switch~case문
반복문	동일한 명령을 여러 번 처리하거나 특정 연산을 반복적으로 처리한다.	<ul style="list-style-type: none">• for문• while문• do~while문
보조 제어문	조건문을 만나면 건너뛰거나 반복 수행을 종료한다. 반복문 내에서 사용한다.	<ul style="list-style-type: none">• continue문• break문

❖ Javascript 제어문의 종류

❖ 조건문

◆ if 문

- 조건식이 참(true)이면 블록 내의 문장을 처리하고, 거짓(false)이면 블록을 빠져나감

```
if ( 조건 ) {  
    실행문장;  
}
```

```
if ( 조건1 ) {  
    실행문장1;  
    if ( 조건2 ) {  
        실행문장2;  
    }  
}
```

◆ if ~ else 문

- 조건식이 참(true)인 경우와 거짓(false)인 경우 처리할 문장이 각각 따로 있을 때 사용하는 제어문

```
if ( 조건 ) {  
    실행문장1; // 조건이 참인 경우 실행  
}  
else {  
    실행문장2; // 조건이 거짓인 경우 실행  
}
```

❖ 조건문

◆ if ~ else if ~ else 문

- 두개 이상 조건식이 참(true)인 경우와 거짓(false)인 경우 처리할 문장이 각각 따로 있을 때 사용하는 제어문



```
if ( 조건1 ) {
    실행문장1;    // 조건1이 참인 경우 실행
}
else if( 조건2 ) {
    실행문장2;    // 조건2이 참인 경우 실행
}
else {
    실행문장3;    // 두 조건이 모두 거짓인 경우 실행
}
```

❖ if 문

js_if01.html

```
<script>
  var myage = 19;

  // 같은 경우
  if ( myage == 19 ) {
    alert("당신의 나이는 " + myage + " 살입니다.");
  }

  // 다른 경우
  if ( myage != 20 ) {
    alert("당신의 나이는 20살이 아닙니다.");
  }
</script>
```

❖ if ~ else문

js_ifelse01.html

```
<script>
    var gender="M"; // 남자(M), 여자(F)
    var age=21;
    if(gender=="M") {
        if(age>=19) {
            result="남자 성인입니다.";
        }
        else {
            result="남자 미성년자입니다.";
        }
    }
    else {
        if (age>=19) {
            result="여자 성인입니다.";
        }
        else {
            result="여자 미성년자입니다.";
        }
    }
    document.write("당신은 " + result + "<p/>");
</script>
```



❖ if ~ else문

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
</head>
<body>
    <p>아이디, 비밀번호 입력</p>
    <script src=".js/script.js"> </script>
</body>
</html>
```

js_ifelse02.html

```
id=prompt('아이디 입력');
if(id=='admin') {
    password=prompt('비밀번호 입력');
    if(password==='123456') {
        location.href="success.html"
    }
    else {
        location.href="error.html"
    }
}
else {
    location.href="error.html"
}
```

./js/script.js



❖ if ~ else문

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
</head>
<body>
    <h2>회원 인증에 성공했습니다.</h2>
    <p>자바스크립트 공부를 합시다!</p>
    <a href="https://developer.mozilla.org/ko/docs/Web/JavaScript">Mozilla 사이트</a>
</body>
</html>
```

success.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
</head>
<body>
    <h2>회원 인증에 실패했습니다.</h2>
    <p>웹 문서에 접근할 수 없습니다. 관리자에게 문의하시기 바랍니다.</p>
    <p>관리자 e-mail : qwert@asdfg.com</p>
</body>
</html>
```

error.html

❖ if ~ else if ~ else 문

```
<script>
    var point=prompt("과목 점수를 입력하세요!"); // 과목 점수
    var grade=null;
    if(point>100) {
        document.write("<p>0~100점 사이 값을 입력해야 합니다." + "</p>");
    }
    else if(point>=90) {
        grade="A";
        document.write("<p>아주 잘했어요." + "</p>");
    }
    else if(point>=80) {
        grade="B";
        document.write("<p>잘했어요." + "</p>");
    }
    else if(point>=70) {
        grade="C";
        document.write("<p>조금만 노력하면 잘할 수 있어요." + "</p>");
    }
    else if(point>=60) {
        grade="D";
        document.write("<p>좀 더 노력하세요." + "</p>");
    }
    else{
        grade="F";
        document.write("<p>많이 노력하시기 바랍니다." + "</p>");
    }
    document.write("<p>학생의 학점은 <b>" + grade + "</b>입니다.</p>");
</script>
```

js_ifelse03.html

❖ 조건문

◆ switch ~ case 문

- 조건문을 체크하여 다음에 처리할 문장의 위치를 파악한 후 해당 문장으로 가서 바로 처리

```
switch ( 기준값 ) {  
    case 값1:  
        실행문장1;  
        break;  
    case 값2:  
        실행문장2;  
        break;  
    case 값n:  
        실행문장n;  
        break;  
    default:  
        기본실행문장;  
        break;  
}
```

❖ switch ~ case 문

```
<script>
    var day;
    var week=new Date().getDay(); // 0(일요일)~6(토요일)
    switch(week) {
        case 0:
            day="일요일";
            break;
        case 1:
            day="월요일";
            break;
        case 2:
            day="화요일";
            break;
        case 3:
            day="수요일";
            break;
        case 4:
            day="목요일";
            break;
        case 5:
            day="금요일";
            break;
        case 6:
            day="토요일";
            break;
        default:
            day="없는 요일";
            break;
    }
    document.write("<p>오늘은 <b>" + day + "</b>입니다. </p>");

```

js_switchCase03.html

❖ 반복문

- ◆ **for**
- ◆ **while**
- ◆ **do while**

```
// 합계 초기값  
var sum = 0;  
  
// 카운터 변수 i는 1부터 100까지 1씩 증가하면서 sum에 누적
```

```
1   2   5  
for (var i = 1; i < 100 ; i++) {  
    3  
    sum += 1 ; 4  
}
```

* for 명령의 실행 순서

❖ 함수의 반환값

- ◆ 함수에 반환값이 있는 경우 **return 반환값;** 형식으로 값을 반환 해줄 수 있다.

```
function 함수명(매개변수1, 매개변수2, 매개변수3) { // 함수선언
    실행문장;
    return 반환값;
}
result = 함수명(인자1, 인자2, 인자3); // 함수 호출
```

```
<script>
    var result;
    function add(name, n) {
        document.write(name + " 학생이 1부터 " + n + "까지 덧셈 수행<br>");
        var sum=0;
        for(var i=1; i<=n; i++) {
            sum+=i;
        }           =
        return sum;
    }
    result=add('홍길동', 10);
    document.write("결과 : " + result + "<p/>");
    result=add('이영희', 100);
    document.write("결과 : " + result + "<p/>");
</script>
```

❖ 함수를 변수에 담기

- ◆ 함수를 변수에 담을 수도 있다.

```
<script type="text/javascript">
    function abcd() {// 함수 선언
        var a=10;
        var b=5;
        alert(a-b);
    }

    var fun1 = abcd; // 함수 대입
    fun1();           // 변수 실행
</script>
```

❖ 배열

- 여러 데이터 값을 저장하는 공간
- 원소: 배열에 저장된 하나 하나의 데이터
- 인덱스: 원소를 구분하는 번호, 0부터 매김



◆ 배열 리터럴로 생성하기

```
var 배열명=[원소1, 원소2, 원소3, ...];
```

```
<script type="text/javascript">
    var city=[]; // 배열 변수 선언
    city[0]="Seoul";
    city[1]="Busan";
    city[2]="Incheon";
    city[3]="Mokpo";
    city[4]="Sejeong";
    function printArr(){
        var i;
        for(i=0; i<city.length; i++) {
            document.write("배열 데이터 [" + i + "] = " + city[i] + "<br>");
        }
    }
    printArr();
</script>
```

❖ 배열

◆ 배열 객체로 생성하기

```
var 배열명=new Array(원소1, 원소2, 원소3, ... );
```

```
<script type="text/javascript">
    var city=new Array("Seoul","Busan","Incheon");
    function printArr() {
        var i;
        for(i=0; i<city.length; i++) {
            document.write("배열 데이터 [" + i + "] = " + city[i] + "<br>");
        }
    }
    printArr();
</script>
```

❖ 배열

◆ 배열 객체 생성 확인 방법

방법	사용 예	결과
타입 확인 연산자인 typeof 사용	typeof city	object
배열 객체의 메소드인 isArray() 사용	Array.isArray(city)	true
Array 생성자의 연산자인 instanceof 사용	city instanceof Array	true

```
<script type="text/javascript">
var city=new Array("Seoul","Busan","Incheon");
function printArr() {
    if(city instanceof Array) {
        document.write("배열 객체가 생성되었습니다.<p/>");
        var i;
        for(i=0; i<city.length; i++) {
            document.write("배열 데이터 [" + i + "] = " + city[i] + "<br>");
        }
    } else {
        document.write("배열 객체가 아닙니다.<br>");
        document.write("데이터 : " + city + "<br>");
    }
}
printArr();
document.write("<p/> city 변수 타입 : " + typeof city + "<br>");
document.write("배열 객체 확인 결과 : " + Array.isArray(city) + "<br>");
</script>
```

❖ 배열 데이터 접근 및 조작

- ◆ 홀수 번째 저장된 데이터만 0으로 초기화하기

```
<script>
    var arrdata=[];
    function insertArr() {
        var i=0;
        for(i=0; i<=99; i++) {
            arrdata[i]=i+1;
            document.write(arrdata[i] + " ");
        }
    }
    function delArr() {
        var i;
        for(i=0; i<arrdata.length; i++) {
            if(i%2==0) {
                arrdata[i]=0;
            }
            continue;
        }
        selectArr();
    }
    function selectArr() {
        var i;
        for(i=0; i<arrdata.length; i++) {
            document.write(arrdata[i] + " ");
        }
        document.write("<p>홀수 번째 데이터 초기화 완료!" + "</p>");
        document.write("<a href='22_arr.html'>돌아가기</a>");
    }
    insertArr();
</script>
```

❖ 배열 데이터 접근 및 조작

◆ 배열에 저장된 데이터 삭제하기

```
<script>
    var arrdata=[];
    function insertArr() {
        var i=0;
        for(i=0; i<=99 ; i++) {
            arrdata[i]=i+1;          // 1~100 저장
            document.write(arrdata[i] + " "); // 데이터 출력
        }
        document.write("<p>배열 크기 : " + arrdata.length + "</p>");
    }
    function delDataArr() {
        var i;
        for(i=0; i<arrdata.length; i++) {
            arrdata[i]=0;           // 배열 데이터를 0으로 초기화
        }
        selectArr();
    }
    function allDelArr() {
        arrdata.length=0; // 배열 초기화
        selectArr();
    }
    function selectArr() {
        var i;
        for(i=0; i <arrdata.length; i++) {
            document.write(arrdata[i] + " "); // 데이터 조회
        }
        document.write("<p> 배열 크기 : " + arrdata.length + "</p>");
        document.write("<a href='23_arr.html'>돌아가기</a>");
    }
    insertArr(); // 배열 데이터 생성 함수 호출
</script>
<p/>
<button type="button" onclick="delDataArr()">배열 데이터 초기화</button>
<button type="button" onclick="allDelArr()">배열 데이터 삭제</button>
```

❖ 배열

◆ join

- 배열에 저장된 모든 원소를 문자열로 변환한 후 연결하여 출력

```
<script type="text/javascript">
  var city=["서울", "부산", "대전"];
  var joindata1=city.join();
  var joindata2=city.join('-');
  var joindata3=city.join(' 그리고 ');
  document.write("조인 결과1 : " + joindata1 + "<p/>");
  document.write("조인 결과2 : " + joindata2 + "<p/>");
  document.write("조인 결과3 : " + joindata3 + "<p/>");
</script>
```

조인 결과1 : 서울,부산,대전
조인 결과2 : 서울-부산-대전
조인 결과3 : 서울 그리고 부산 그리고 대전

❖ 배열

◆ concat

- 지정된 배열에 두 개 이상의 데이터를 결합하거나 다른 배열 객체를 결합

```
<script type="text/javascript">
    var city01=["서울", "부산", "대전"];
    var city02=["대구", "광주", "인천"];
    var city03=["전주", "부여", "세종"];
    var data1=city01.concat("수원", "오산");
    var data2=city01.concat(city02);
    var data3=city01.concat(city03, city02);
    document.write("결과1 : " + data1 + "<p/>");
    document.write("결과2 : " + data2 + "<p/>");
    document.write("결과3 : " + data3 + "<p/>");
</script>
```

❖ 배열

◆ reverse

- 배열 원소의 순서를 반대로 정렬

```
<script>
    var data=[9, 8, 7, 6, 5, 4, 3, 2, 1];
    document.write("배열 :" + data.join() + "<p/>");
    var rdata=data.reverse(); // 배열 원소를 반대로 정렬
    document.write("결과 :" + rdata + "<p/>");
</script>
```

◆ sort

- 배열 원소를 정렬

```
<script>
    var ndata1=[19, 38, 67, 26, 55, 24, 53, 12, 31];
    var ndata2=[132, 2, 41, 123, 45, 1234, 6, 29, 4567];
    var edata=['Apple', 'Html', 'Game', 'Computer', 'Java'];
    var kdata=['서울', '부산', '구포', '대구', '인천'];
    document.write("수치 정렬1 :" + ndata1.sort() + "<p/>");
    document.write("수치 정렬2 :" + ndata2.sort() + "<p/>");
    document.write("수치 정렬3 :" + ndata2.sort(function(a, b) {return a - b;}) + "<p/>");
    document.write("영문 정렬 :" + edata.sort() + "<p/>");
    document.write("한글 정렬 :" + kdata.sort() + "<p/>");
</script>
```

❖ 배열

◆ slice

- 배열의 특정 범위에 속하는 원소만 선택하여 배열 생성

```
<script>
    var kdata=['서울', '부산', '구포', '대구', '인천', '대전', '세종'];
    var str1=kdata.slice(0, 4);
    var str2=kdata.slice(2, -1);
    var str3=kdata.slice(-4, -2);
    document.write("부분 배열1 : " + str1 + "<p/>");
    document.write("부분 배열2 : " + str2 + "<p/>");
    document.write("부분 배열3 : " + str3 + "<p/>");
</script>
```

◆ splice

- 배열의 원소 추가 또는 제거

```
<script>
    var kdata=['서울', '부산', '구포', '대구', '대전'];
    var str1=kdata.splice(1, 2);
    document.write("삭제 데이터 : " + str1 + "<br>");
    document.write("남은 배열 : " + kdata + "<p/>");
    var str2=kdata.splice(1, 1, '강릉', '세종');
    document.write("삭제 데이터 : " + str2 + "<br>");
    document.write("남은 배열 : " + kdata + "<p/>");
    var str3=kdata.splice(2, Number.MAX_VALUE);
    document.write("삭제 데이터 : " + str3 + "<br>");
    document.write("남은 배열 : " + kdata + "<p/>");
</script>
```

❖ 배열

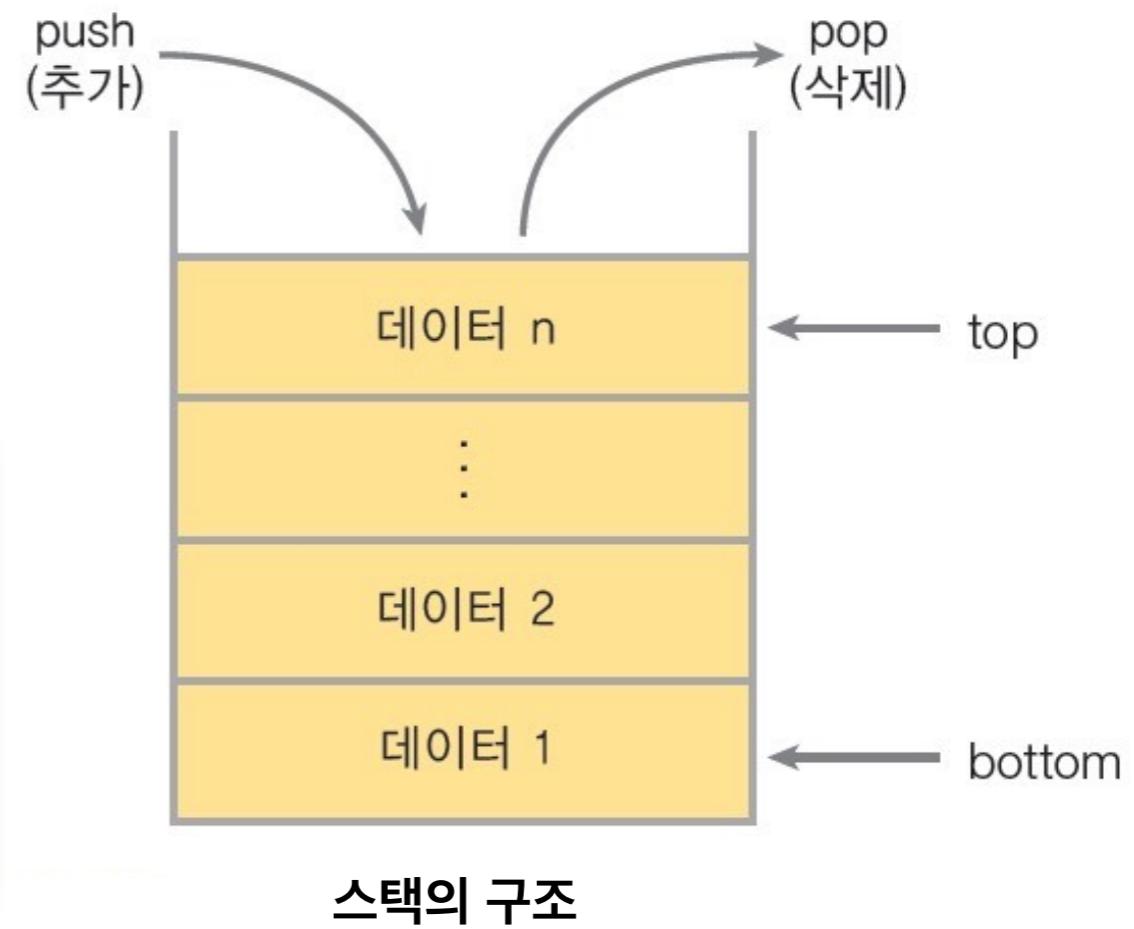
◆ pop & push

- **push** 메소드: 배열의 마지막 위치에 데이터를 추가하고 배열의 길이를 반환
- **pop** 메소드: 배열의 마지막 위치에 있는 데이터를 삭제하고 삭제한 데이터를 반환

* 스택

- 모든 데이터의 삽입과 삭제가 배열의 한쪽 끝에서만 수행되는 자료 구조

```
<script>
  var kdata=['서울', '부산', '구포', '대구', '대전'];
  var p1=kdata.push('청주', '세종');
  document.write("데이터 : " + p1 + "<br>");
  document.write("배열 데이터 : " + kdata + "<p/>");
  var p2=kdata.pop();
  document.write("데이터 : " + p2 + "<br>");
  document.write("배열 데이터 : " + kdata + "<p/>");
</script>
```



❖ 배열

◆ shift & unshift

- **shift** 메소드: 배열의 맨 처음 위치에 데이터를 삭제하고 배열의 삭제된 데이터 반환
- **unshift** 메소드: 배열의 맨 처음 위치에 데이터를 삽입하고 배열의 길이 반환

```
<script>
  var kdata=['서울', '부산'];
  var p1=kdata.unshift('청주', '세종');
  document.write("데이터 : " + p1 + "<br>");
  document.write("배열 데이터 : " + kdata + "<p/>");
  var p2=kdata.shift();
  document.write("데이터 : " + p2 + "<br>");
  document.write("배열 데이터 : " + kdata + "<p/>");
</script>
```

❖ 배열

◆ forEach

- 배열을 반복하며 저장된 데이터를 조회

```
<script>
    var kdata=['서울', '부산', '청주', '대구'];
    function printArr(item, index) {
        document.write("배열 데이터 [" + index + "] :" + item + "<br>");
    }
    kdata.forEach(printArr);
</script>
```

```
<script>
    var data=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
    var sum=0;
    function addArr(value) {
        sum+=value;
    }
    data.forEach(addArr);
    document.write("배열 데이터 합 :" + sum + "<p/>");
</script>
```

❖ 배열

◆ map

- 배열의 데이터를 함수의 인자로 전달하고 함수의 수행 결과를 반환 받아 새로운 배열 생성

```
<script>
    var data=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
    function mapArr(value) {
        return value*value;
    }
    var mapdata=data.map(mapArr);
    document.write("원래 배열 :" + data + "<p/>");
    document.write("map 메소드 적용 배열 :" + mapdata + "<p/>");
</script>
```

◆ filter

- 배열의 데이터 중에 조건이 참인 데이터만 반환하여 새로운 배열 생성

```
<script>
    var data=[21, 42, 33, 14, 25, 12, 37, 28, 16, 11];
    function filterArr(value) {
        return value>=18;
    }
    var fdata=data.filter(filterArr);
    document.write("필터 전 배열 :" + data + "<p/>");
    document.write("필터 후 배열 :" + fdata + "<p/>");
</script>
```

❖ 배열

◆ indexOf & lastIndexOf

- 배열의 데이터를 검색하여 인덱스 위치를 반환

- **indexOf** 메소드: 검색 시작 위치를 지정할 수 있음
- **lastIndexOf** 메소드: 배열의 맨 마지막 원소부터 검색 시작

```
<script>
    var data=[10, 20, 30, 40, 30, 60, 70, 30, 90,100];
    document.write("배열 데이터 : [" + data + "]<p/>");
    document.write("처음부터 검색한 30의 인덱스 : " + data.indexOf(30) + "<p/>");
    document.write("마지막에서 검색한 30의 인덱스 : " + data.lastIndexOf(30) + "<p/>");
    document.write("세 번째부터 검색한 30의 인덱스 : " + data.indexOf(30, 3) + "<p/>");
```

❖ 연관 배열



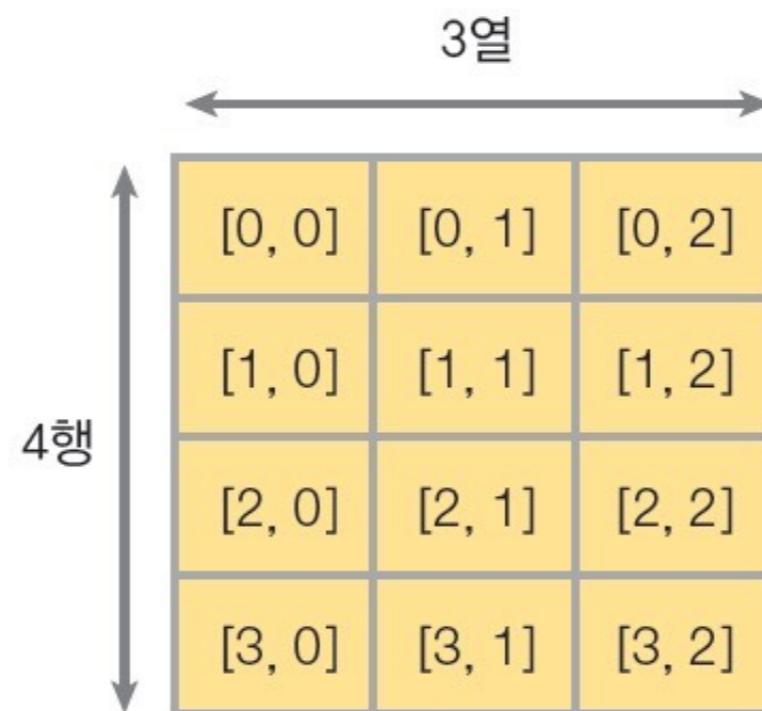
◆ 연관 배열 생성 방법

```
arr={key_1:value1, key_2:value2, ..... , key_n:value_n};
```

```
<script>
    var data={'f0':100, 'f1':200, 'f2':300};
    data['f3']=400; // 배열 데이터 저장
    data.f4=500; // 배열 데이터 저장
    document.write(data.f0 + "<br>"); // 'f0'키 데이터 조회
    document.write(data.f1 + "<br>"); // 'f1'키 데이터 조회
    document.write(data['f2'] + "<br>");
    document.write(data['f3'] + "<br>");
    document.write(data['f4'] + "<br>");
</script>
```

❖ 2차원 배열

◆ 2차원 배열의 구조



```
<script>
    var d2data=[[10, 20, 30, 40, 0], [60, 70, 80, 90, 0]];
    d2data[0][4]=50;
    d2data[1][4]=100;
    document.write("2차원 배열 첫 번째 데이터 : " + d2data[0][0] + "<br>");
    document.write("2차원 배열 마지막 데이터 : " + d2data[1][4] + "<br>");
    document.write("2차원 배열 행 길이 : " + d2data.length + "<br>");
    document.write("2차원 배열 열 길이 : " + d2data[0].length + "<br>");
</script>
```

❖ 2차원 배열

◆ 1차원 배열로 2차원 배열 생성하고 조회하기

```
<script>
    var arr0=[10, 20, 30, 40, 50];
    var arr1=[11, 21, 31, 41, 51];
    var arr2=[12, 22, 32, 42, 52];
    var arr3=[13, 23, 33, 43, 53];
    var allArr=[arr0, arr1, arr2, arr3]; // 2차원 배열 생성
    var partArr=[arr1, arr3];          // 2차원 배열 생성
    function printAll() {
        for(var x=0; x<allArr.length; x++) {
            for(var y=0; y<allArr[x].length; y++) {
                document.write(allArr[x][y] + " ");
            }
            document.write("<p/>");
        }
        document.write("<a href='39_arr.html'>돌아가기</a>");
    }
    function printPart() {
        for(var x=0; x<partArr.length; x++) {
            for(var y=0; y<partArr[x].length; y++) {
                document.write(partArr[x][y] + " ");
            }
            document.write("<p/>");
        }
        document.write("<a href='39_arr.html'>돌아가기</a>");
    }
</script>
<button type="button" onclick="printAll()">전체 배열 데이터 보기</button></p>
<button type="button" onclick="printPart()">홀수 배열 데이터 보기</button>
```

❖ 2차원 배열

◆ for 문 이용 2차원 배열 만들기

```
<script>
    var data=[];
    for(var i=0; i<10; ++i) {
        data[i]=[String(i+"-"+0), String(i+"-"+1), String(i+"-"+2)];
    }
    function printData() {
        for(var x=0; x<data.length; x++) {
            for(var y=0; y<data[x].length; y++) {
                document.write(data[x][y] + " ");
            }
            document.write("<p>");
        }
        document.write("<a href='40_arr.html'>돌아가기</a>");
    }
</script>
<button type="button" onclick="printData()">전체 배열 데이터 보기</button>
```

❖ Javascript Event

- ◆ 특정한 상황이 발생했을 때 호출되도록 사용자가 정의하는 특정한 동작(함수)

특정한 상황

- ◎ 사용자가 웹페이지에게 행한 일련의 동작
(예] 어떤 요소를 클릭한다.
어떤요소에 마우스를 올린다.)
- ◎ 웹 페이지가 실행되는 동안 발생한 어떠한 사건
(예] 페이지 로딩 완료)

특정한 동작

- ◎ 사용자가 정의한 함수로서 그 안의 구문들이 특정한 상황이 발생했을 때, 호출되게 된다.
 - 특정한 상황이 반복적으로 발생하게 되더라도
(함수가 명령어들을 재사용하기 위한 처리이므로)
모든 상황에 대처할 수 있게 된다.

◆ 이벤트의 종류

- ❖ on + HTML 상태
- ❖ HTML 태그의 속성으로 사용

❖ Javascript Event

◆ 이벤트의 종류

이벤트이름	설명
onAbort	이미지를 로딩하는 작업이 사용자의 행동으로 인해 취소되었을 때
onBlur	문서나 윈도우, 프레임세트, 폼 요소에서 현재 입력 포커스가 사라질 때
onChange	텍스트 필드나 텍스트 영역, 파일 업로드 필드, 선택 항목이 변경되어 현재 입력 포커스가 사라질 때
onClick	링크나 클라이언트측 이미지 맵 영역, 폼 요소가 클릭되었을 때
onDbClick	링크나 클라이언트 측 이미지 맵 영역, 문서가 더블 클릭되었을 때
onDragDrop	드래그된 객체가 윈도우나 프레임에 드롭되었을 때
onError	이미지나 윈도우, 프레임을 로딩하는 동안 에러가 발생할 때
onFocus	문서나 윈도우, 프레임 세트, 폼 요소에 입력 포커스가 놓였을 때
onKeyDown	키를 누를 때
onKeyPress	키를 눌렀다 놓았을 때
onKeyUp	키를 놓았을 때
onLoad	이미지나 문서, 프레임이 로드될 때
onMouseDown	마우스 버튼 누를 때
onMouseMove	마우스를 이동할 때
onMouseOver	어떠한 태그 요소를 위로 마우스 커서가 들어갈 때
onMouseOut	어떠한 태그 요소의 위에서 마우스 커서가 빠져나올 때
onMouseUp	마우스 버튼을 놓았을 때
onMove	사용자가 윈도우나 프레임에서 이동할 때
onReset	폼의 리셋 버튼을 클릭하여 폼을 리셋시킬 때

❖ 이벤트 사용 방법

1. 특정 이벤트에 사용할 함수를 **<script>** 태그 블록 안에 만든다.
2. HTML 태그에서 이벤트 이름을 속성으로 사용한다.
3. 속성에 대한 값을 함수 호출 구문으로 적용한다.

```
<script type="text/javascript">
    function sayHello(){
        alert("Hello Javascript");
    }
</script>
```

1

```
<input type="button" value="결과보기" 2 이벤트이름="sayHello()" 3 />
```

2

3

❖ 객체 모델링

◆ 객체

- 세상에 존재하는 모든 것

◆ 자동차 객체의 모델링



객체	속성	메소드
car	car.name="Sonata" car.speed=100 car.color="white" car.door=4	car.start(){} car.accel(){} car.break(){} car.trans(){} ...

❖ 자바스크립트 객체

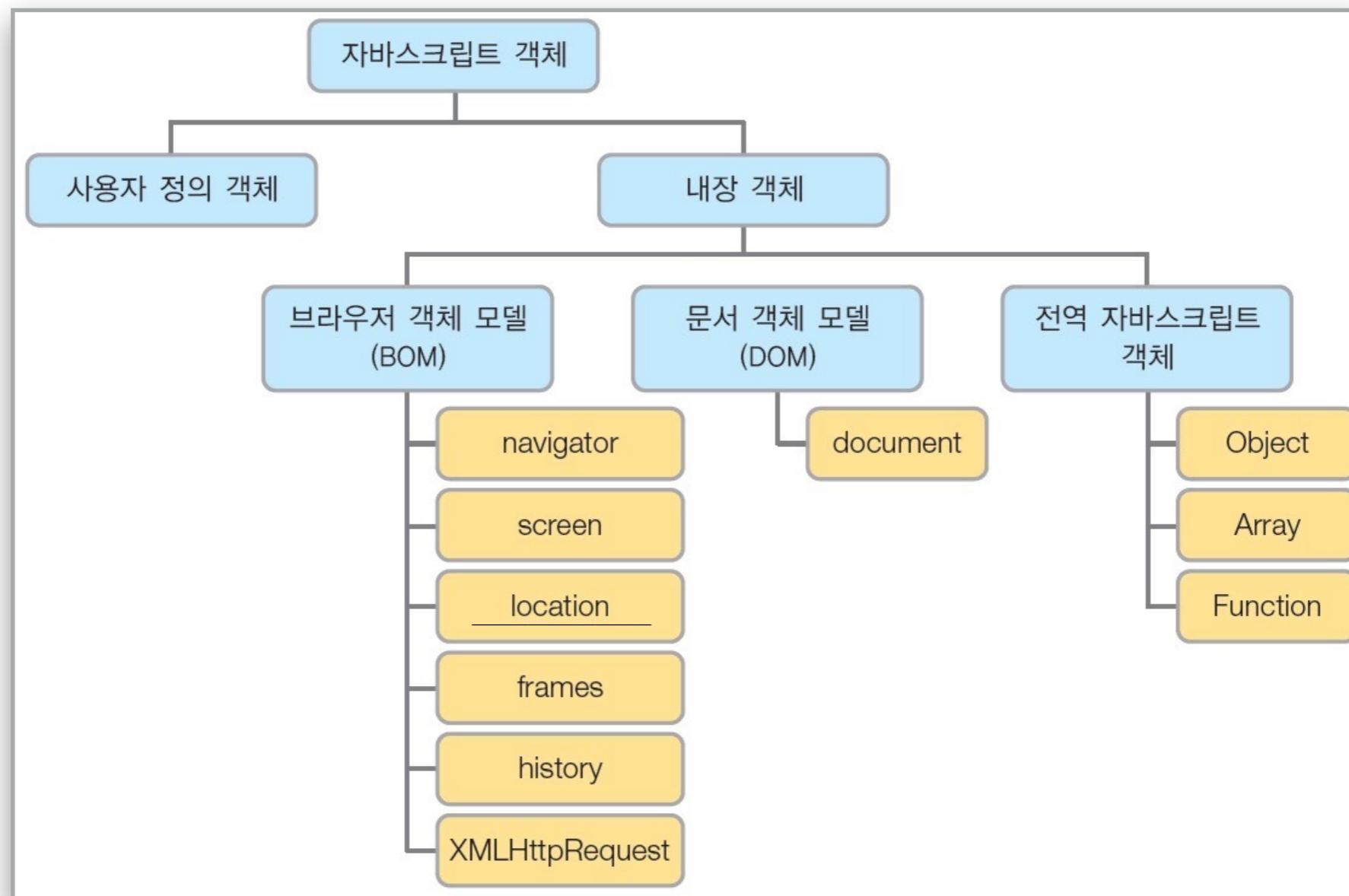
◆ 자바스크립트 객체

- 사용자 정의 객체: 사용자가 직접 객체의 속성과 메소드를 정의하여 사용하는 객체
(예: Car(), House(), Hotel())
- 내장 객체: 자바스크립트 프로그램 자체에서 정의하여 사용자에게 제공하는 객체
(예: Object(), Array(), Date())

◆ 내장 객체의 종류

- 브라우저 객체 모델(BOM, Browser Object Model): 웹 브라우저의 각종 요소를 객체로 표현
- 문서 객체 모델(DOM, Document Object Model): 웹 문서의 각종 요소를 객체로 표현
- 전역 자바스크립트 객체(Global JavaScript Objects): 자바스크립트 프로그램 전체에서 사용하는 내장 객체

❖ 자바스크립트 객체



◆ 자바스크립트 객체의 종류

❖ 객체 변수 이용 방법

◆ 객체 변수를 이용하여 객체 생성

```
객체 변수 속성값  
↓ ↓  
var car = {  
    속성명 → name: "Sonata",  
    speed: 100,  
    color: "white",  
    door: 4,  
  
    메소드명 → start: function() {  
        return this.speed+10;  
    }  
};
```

속성 정의
메소드 정의 (함수 형식)

◆ 객체 속성 접근 방법

방법	사용 예	방법	사용 예
객체명.속성명	car.name car.speed car.color	객체명['속성명']	car['name'] car['speed'] car['color']

❖ 객체 변수 이용 방법

◆ 자바스크립트로 제어할 요소를 찾아 결과를 출력하는 방법

방법	사용 예	의미
innerHTML 속성 이용	document.getElementById("carname").innerHTML;	웹 문서 안에서 아이디가 "carname"인 요소를 찾아 내용을 출력한다.
textContent 속성 이용	var cname=document.getElementById("carname"); cname.textContent;	웹 문서 안에서 아이디가 "carname"인 요소를 찾아서 cname 변수에 반환한 후 cname 변수의 내용을 출력한다.

◆ 속성만 가진 객체 만들기

```
<body>
  <p id="var1"></p>
  <p id="var2"></p>
  <p id="var3"></p>
  <script>
    var car={name: 'Sonata', speed: 100, color: 'white'};
    document.getElementById("var1").innerHTML="자동차 이름 : " + car['name'];
    document.getElementById("var2").innerHTML="자동차 속도 : " + car.speed;
    document.getElementById("var3").innerHTML="자동차 색상 : " + car.color;
  </script>
</body>
```

js_obj01.html

❖ 객체 변수 이용 방법

- ◆ 메소드를 호출하여 연산 결과 출력하기

```
<body>
  <p id="msg1"></p>
  <p id="msg2"></p>
  <p id="msg3"></p>
  <script>
    var obj={
      m1: function() {
        return "Hello Sonata";
      },
      m2: function(a) {
        var result=a;
        return result;
      },
      m3: function(a, b) {
        var result=a+b;
        return result;
      }
    };
    document.getElementById("msg1").innerHTML=obj.m1();
    document.getElementById("msg2").innerHTML=obj.m2(100);
    document.getElementById("msg3").innerHTML=obj.m3(100, 200);
  </script>
</body>
```

js_obj02.html

★ 객체 맴버함수 만들기 2

```
함수이름 : function() {
  return data;
}
```

❖ 객체 변수 이용 방법

◆ 자동차 객체 생성하기

```
<body>
  <p id="carname"></p>
  <p id="carcolor"></p>
  <p id="carspeed"></p>
  <script>
    var car={
      name: 'Sonata',
      speed: 50,
      color: 'white',
      start: function() {
        return this.speed+10;
      }
    };
    var cname=document.getElementById("carname");
    cname.textContent=car.name;
    var colname=document.getElementById("carcolor");
    colname.textContent=car.color;
    var cspeed=document.getElementById("carspeed");
    cspeed.textContent=car.start();
  </script>
</body>
```

js_obj03.html

❖ 객체 변수 이용 방법

◆ 자동차의 속도 조절하기

```
<body>
  <p id="upspeed"></p>
  <p id="downspeed"></p>
  <script>
    var car={
      name: 'Sonata',
      speed: 50,
      color: 'white',
      speedup: function() {
        return this.speed+10;
      },
      speeddown: function() {
        var low=this.speed-10;
        return low;
      }
    };
    var upspeed=document.getElementById("upspeed");
    upspeed.textContent='속도 증가 : ' + car.speedup();
    var downspeed=document.getElementById("downspeed");
    downspeed.textContent='속도 감소 : ' + car.speeddown();
  </script>
</body>
```

js_obj04.html

❖ 객체 변수 이용 방법

◆ 자동차의 속도 제어하기

```
<body>
  <p id="upspeed"></p>
  <p id="downspeed"></p>
  <script>
    var car={
      name: 'Sonata',
      speed: 100,
      color: 'white',
      speedup: function(a) {
        var sp=this.speed+a;
        if(sp>=300) {
          sp=50;
          return sp;
        }
        else {
          return sp;
        }
      },
      speeddown: function(a) {
        var sp=this.speed-a;
        if(sp<0) {
          sp=0;
          return sp;
        }
        else {
          return sp;
        }
      }
    };
    var upspeed=document.getElementById("upspeed");
    upspeed.textContent='속도 증가 : ' + car.speedup(100);
    var downspeed=document.getElementById("downspeed");
    downspeed.textContent='속도 감소 : ' + car.speeddown(30);
  </script>
</body>
```

js_obj05.html

❖ 생성자 함수를 이용하는 방법

- ◆ Object 함수 이용

```
<script>
    var car = new Object();           // 객체 생성
    car.name = 'Lamborghini';        // 속성 정의
    car.speed = 250;
    car.color = 'orange';
    car.speedup = function() {        // 함수 정의
        return this.speed + 50 ;
    };
</script>
```

❖ 생성자 함수를 이용하는 방법

◆ Object 함수 이용 객체 만들기

```
<body>
  <p id="upspeed"></p>
  <p id="downspeed"></p>
  <script>
    var car={
      name: 'Sonata',
      speed: 50,
      color: 'white',
      speedup: function() {
        return this.speed+10;
      },
      speeddown: function() {
        var low=this.speed-10;
        return low;
      }
    };
    var upspeed=document.getElementById("upspeed");
    upspeed.textContent='속도 증가 : ' + car.speedup();
    var downspeed=document.getElementById("downspeed");
    downspeed.textContent='속도 감소 : ' + car.speeddown();
  </script>
</body>
```

js_obj06.html

❖ 생성자 함수를 이용하는 방법

◆ 생성자 함수 정의

```
function Car(name, color, speed) {  
    this.name = name;  
    this.color = color;  
    this.speed = speed;  
    this.speedup = function(){  
        return this.speed + 10;  
    };  
    this.speeddown = function() {  
        return this.speed - 10;  
    }  
}
```

❖ 생성자 함수를 이용하는 방법

- ◆ 생성자 함수 정의 후 객체 만들기

js_obj07.html

```
<body>
<p>[Hong's Car]</p>
<p id="carname"></p>
<p id="carcolor"></p>
<p id="carspeed"></p>
<p>[Kim's Car]</p>
<p id="carname2"></p>
<p id="carcolor2"></p>
<p id="carspeed2"></p>
<script>
    function Car(name, color, speed) {
        this.name=name;
        this.color=color;
        this.speed=speed;
        this.speedup=function() {
            return this.speed+10;
        };
        this.speeddown=function() {
            return this.speed-10;
        };
    }
    var Hongcar=new Car('Sonata', 'blue', 100);
    var Kimcar=new Car('Jeep', 'red', 70);
    var cname=document.getElementById("carname");
    cname.textContent='자동차 이름 : ' + Hongcar.name;
    var colname=document.getElementById("carcolor");
    colname.textContent='자동차 색상 : ' + Hongcar.color;
    var cspeed=document.getElementById("carspeed");
    cspeed.textContent='자동차 속도 : ' + Hongcar.speedup();
    var cname=document.getElementById("carname2");
    cname.textContent='자동차 이름 : ' + Kimcar.name;
    var colname=document.getElementById("carcolor2");
    colname.textContent='자동차 색상 : ' + Kimcar.color;
    var cspeed=document.getElementById("carspeed2");
    cspeed.textContent='자동차 속도 : ' + Kimcar.speedup();
</script>
</body>
```

❖ 생성자 함수를 이용하는 방법

- ◆ 이미 생성된 객체에 속성 추가 및 삭제하기

js_obj08.html

```
<body>
<p>[Hong's Car]</p>
<p id="hong1"></p>
<p id="hong2"></p>
<p id="hong3"></p>
<p>[Kim's Car]</p>
<p id="data1"></p>
<p id="data2"></p>
<p id="data3"></p>
<p id="data4"></p>
<script>
    function Car(name, color, speed) {
        this.name=name;
        this.color=color;
        this.speed=speed;
        this.speedup=function() {
            return this.speed+10;
        };
        this.speeddown=function() {
            return this.speed-10;
        };
    }
    var Hongcar=new Car('Sonata', 'blue', 100);
    var Kimcar=new Car('Jeep', 'red', 70);
    Kimcar.price='3천만 원';
    delete Kimcar.color;
    var cname=document.getElementById("hong1");
    cname.textContent='자동차 이름 : ' + Hongcar.name;
    var colname=document.getElementById("hong2");
    colname.textContent='자동차 색상 : ' + Hongcar.color;
    var cspeed=document.getElementById("hong3");
    cspeed.textContent='자동차 속도 : ' + Hongcar.speedup();
    var cname=document.getElementById("data1");
    cname.textContent='자동차 이름 : ' + Kimcar.name;
    var colname=document.getElementById("data2");
    colname.textContent='자동차 색상 : ' + Kimcar.color;
    var cspeed = document.getElementById("data3");
    cspeed.textContent='자동차 속도 : ' + Kimcar.speedup();
    var cspeed=document.getElementById("data4");
    cspeed.textContent='자동차 가격 : ' + Kimcar.price;
</script>
</body>
```

❖ 객체에 함수 추가하기

```
객체이름.함수이름 = function([매개변수]) {  
    ... 함수 실행 문 ...  
    [return data값;]  
};
```

❖ 함수 안에서 객체의 자원 활용하기

```
객체이름.함수이름 = function([매개변수]) {  
    this.변수이름 = data값;  
    var 변수이름 = this.함수이름(매개변수);  
    [return data값;]  
};
```

❖ 자바스크립트 내장 객체

◆ 자바스크립트 내장객체란?

- 모든 웹 사이트에는 공통적으로 필요한 기능들이 존재한다.
이 기능들을 매번 새로 만들어야 한다면 매우 비효율적이다.
- 이런 불편함 해소를 위해 자바스크립트에서 미리 마련해 둔 내장된 기능을 제공
(예] 내장함수들...)

공통 기능의 필요성

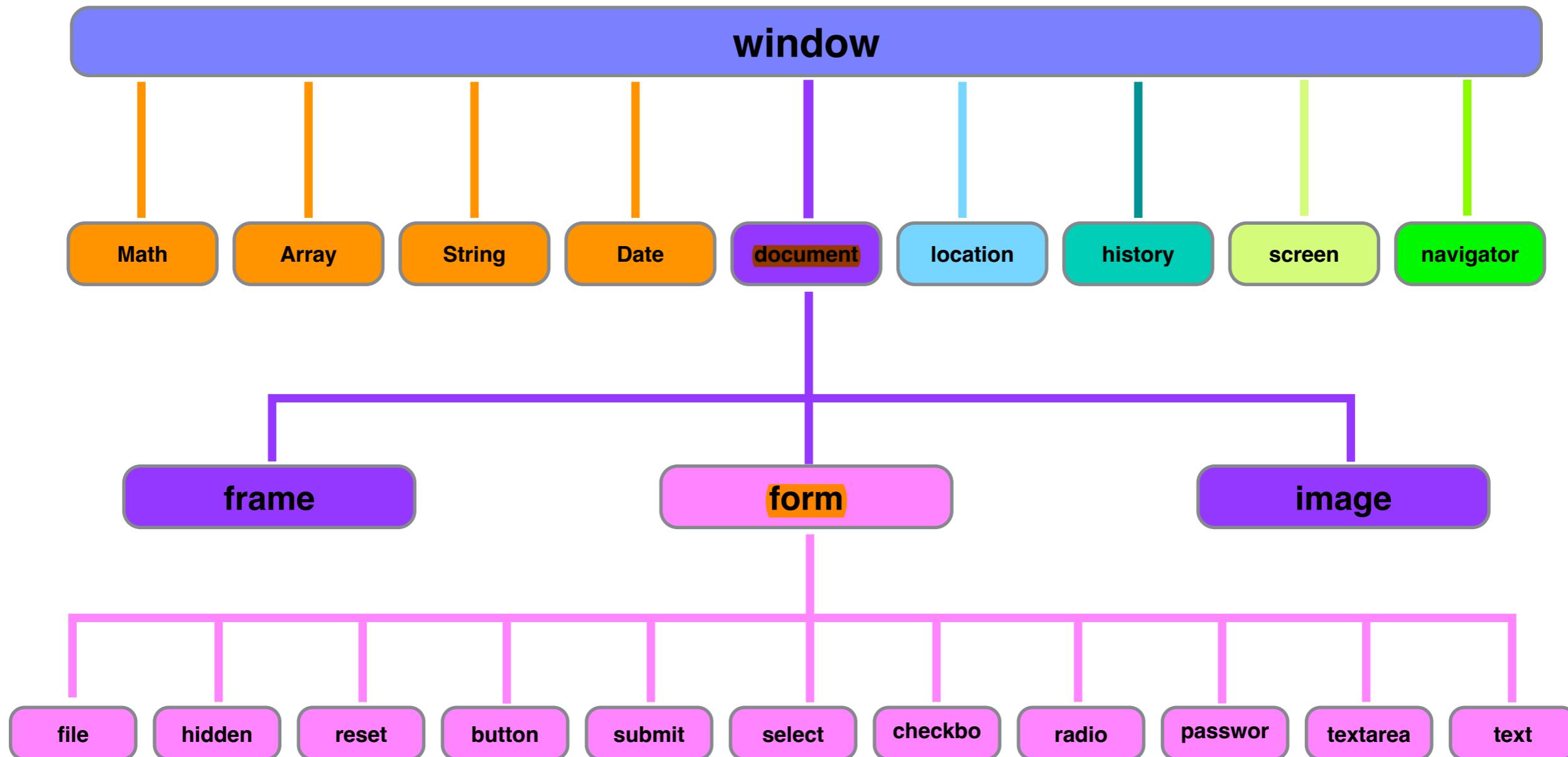
- 웹페이지를 제작하는데 필요한 기능들 중에서
대다수의 기능은
모든 사이트가 공통적으로 요구하는 내용
- 모든 웹 퍼블리셔들이
공통적인 기능을 개별적으로 제작한다면
개발 시간에 대한 낭비를 초래

내장객체의 제공

- 모든 웹 브라우저 개발사들이
이러한 공통 기능의 필요성을 인지하고,
사전에 객체이름과
함수 이름들을 통일하여 구현한뒤,
브라우저 안에 내장
- 웹 퍼블리셔들은
이러한 객체를 활용하는 것 만으로
많은 기능들을 직접 구현하는 수고를 덜 수 있다.

❖ 자바스크립트 내장 객체

- ◆ Javascript에서 제공되는 내장객체는 모두 **window**라는 내장객체의 하위 객체 형태로 존재



* 내장객체의 종류

❖ 자바스크립트 내장 객체

◆ 값의 처리를 위한 내장 객체

이 름	설 명
Date	시스템의 현재 날짜, 시각을 조회하거나 계산하기 위한 기능 제공
Array	같은 종류의 변수를 하나로 묶기 위한 배열에 관련된 기능 제공
String	하나의 문자열을 독립된 객체로 생성 문자열 안에서 특정 글자가 시작하는 위치, 문자열 안에서 원하는 내용만 추출하는 등의 기능 제공
Math	삼각함수, 지수, 로그 등 수학과 관련된 각종 고급 함수를 제공 (일반적으로 잘 사용되지 않는다.)

❖ 자바스크립트 내장 객체

◆ 브라우저의 제어를 위한 내장 객체

이 름	설 명
window	브라우저 창에 대한 모든 상황을 제어하는 최상위 객체 모든 브라우저 제어관련 내장 객체는 Window객체의 하위에 존재
location	URL 정보를 제어하는 객체 페이지 이동, 현재 주소 조회, 새로 고침 등의 기능 제공
history	웹 브라우저에 기록되어 있는 히스토리 정보를 제어
navigator	브라우저의 종류를 판별
screen	브라우저 화면에 대한 정보를 알려준다. 변수값만 포함하고 있으며, 함수는 포함하고 있지 않다.

❖ 자바스크립트 내장 객체

◆ HTML문서를 제어하기 위한 내장 객체

이 름	설 명
document	문서에 대한 정보, 즉 HTML 문서의 각 요소들을 제어하기 위한 기능
image	 태그에 대한 속성을 제어하는 객체
form	입력양식 컴포넌트를 위한 개별 객체들을 포함
frame	웹 페이지 안에 다른 웹 페이지를 포함하는 Frameset과 iframe을 제어하는 기능 제공

❖ 자바스크립트 내장 객체

◆ String 객체

- * 문자 객체(**String Object**)는 문자형 데이터를 객체로 취급하는 것
- * 자바스크립트에서 가장 많이 사용
- * **String** 객체 맴버 변수

이 름	설 명
length	문자열의 길이를 조회

❖ 자바스크립트 내장 객체

◆ String 객체

* String 객체 맴버 함수

반환값타입 함수(변수)

이 름	설 명
<code>String(value)</code>	문자열을 생성하기 위한 생성자
<code>String charAt(int)</code>	지정된 위치의 글자를 리턴
<code>int indexOf(String)</code>	문자열 앞에서부터 파라미터로 주어진 글자를 검색하여 위치를 알려준다. 검색결과가 없을 경우 -1을 리턴
<code>int lastIndexOf(String)</code>	문자열 뒤에서부터 파라미터로 주어진 글자를 검색하여 위치를 알려준다. 검색된 글자의 위치는 앞에서부터 카운트 검색 결과가 없을 경우 -1 리턴
<code>String substring(int, int)</code>	문자열에서 첫 번째 파라미터의 위치부터 두 번째 파라미터의 위치까지 추출 두 번째 파라미터가 없을 경우 끝까지 추출
<code>StringtoUpperCase()</code>	대문자로 변환
<code>StringtoLowerCase()</code>	소문자로 변환
<code>String slice(int, int)</code>	첫 번째 파라미터 만큼 문자를 자르고 두번째 파라미터 이후의 문자를 자른후 남은 문자 반환
<code>String replace(String, String)</code>	첫번째 파라미터의 문자를 찾아 두번째 파라미터로 바꾼 후 반환
<code>String match(String)</code>	파라미터의 문자를 찾아 최초 찾은 문자를 반환 / 없으면 null 반환
<code>int search(String)</code>	왼쪽부터 파라미터의 문자를 찾아 최초로 일치하는 인덱스 번호 반환
<code>String substr(int, int)</code>	첫번째 파라미터 위치에서부터 두번째 파라미터 갯수만큼 문자열을 반환
<code>Array split(String)</code>	파라미터의 문자를 기준으로 문자데이터를 나누어 배열에 저장해서 반환
<code>String concat(String)</code>	문자열에 파라미터로 입력한 문자열을 더해서 반환
<code>String trim()</code>	문자 앞 뒤의 공백 문자를 삭제해서 반환
<code>int charCodeAt(char)</code>	찾을 문자의 아스키코드값을 반환
<code>char fromCharCode(int)</code>	파라미터(아스키코드 값)에 해당하는 문자를 반환

❖ String 내장함수

```
<script type="text/javascript">
var str = "Hello Thank you good luck to you";
document.write("<div class='cont'><div class='c1'>charAt</div><div class='c2'>",
              str.charAt(16), "</div></div>");
document.write("<div class='cont'><div class='c1'>indexOf</div><div class='c2'>"
              + str.indexOf('you') + "</div></div>");
document.write("<div class='cont'><div class='c1'>indexOf(String, int)</div><div class='c2'>"
              + str.indexOf('you',16) + "</div></div>");
document.write("<div class='cont'><div class='c1'>lastIndexOf</div><div class='c2'>"
              + str.lastIndexOf('you') + "</div></div>");
document.write("<div class='cont'><div class='c1'>lastIndexOf(String, int)</div><div class='c2'>"
              + str.lastIndexOf('you',25) + "</div></div>");
document.write("<div class='cont'><div class='c1'>match</div><div class='c2'>"
              + str.match('luck') + "</div></div>");
document.write("<div class='cont'><div class='c1'>search</div><div class='c2'>"
              + str.search('you') + "</div></div>");
document.write("<div class='cont'><div class='c1'>substr</div><div class='c2'>"
              + str.substr(21,4) + "</div></div>");
document.write("<div class='cont'><div class='c1'>substring</div><div class='c2'>"
              + str.substring(6,12) + "</div></div>");
document.write("<div class='cont'><div class='c1'>replace</div><div class='c2'>"
              + str.replace('you','me') + "</div></div>");
document.write("<div class='cont'><div class='c1'>toLowerCase</div><div class='c2'>"
              + str.toLowerCase() + "</div></div>");
document.write("<div class='cont'><div class='c1'>toUpperCase</div><div class='c2'>"
              + str.toUpperCase() + "</div></div>");
document.write("<div class='cont'><div class='c1'>length</div><div class='c2'>"
              + str.length + "</div></div>");

var strArray=str.split(" ");
document.write("<div class='cont'><div class='c1'>split[0]</div><div class='c2'>"
              + strArray[0] + "</div></div>");
document.write("<div class='cont'><div class='c1'>split[4]</div><div class='c2'>"
              + strArray[4] + "</div></div>");
document.write("<div class='cont'><div class='c1'>charCodeAt</div><div class='c2'>"
              + String.charCodeAt('A') + "</div></div>");
document.write("<div class='cont'><div class='c1'>fromCharCode</div><div class='c2'>"
              + String.fromCharCode(65) + "</div></div>");

</script>
```

js_str01.html

❖ String 내장함수

js_str01.html

```
<style type="text/css">
    body {
        text-align: center;
    }
    .cont {
        margin-bottom: 3px;;
    }
    .c1 {
        display: inline-block;
        background-color: orange;
        box-shadow: 3px 3px 20px rgba(100, 100, 100, 0.5);
        border: 1px solid darkgray;
        width: 180px;
        height: 30px;
        padding-top: 10px;
        padding-left: 10px;
        text-align: left;
    }
    .c2 {
        display: inline-block;
        background-color: lightgray;
        box-shadow: 3px 3px 20px rgba(100, 100, 100, 0.5);
        border: 1px solid darkgray;
        width: 300px;
        height: 30px;
        padding-top: 10px;
        padding-left: 10px;
        margin-left: 3px;
        text-align: left;
    }
</style>
```

❖ 자바스크립트 내장 객체

◆ 정규 표현 객체

- * 정규 표현(RegExp) 객체는 입력 요소에 데이터를 규칙에 맞게 작성했는지 판단해서 알려주는 객체
- * 정규 표현 객체 생성 방법

★ 정규 표현 객체 생성 방법 1

```
var 변수이름 = new RegExp(패턴, 검색옵션)
```

★ 정규 표현 객체 생성 방법 2

```
var 변수이름 = /패턴/검색옵션
```

❖ 자바스크립트 내장 객체

◆ 정규 표현 객체

* 정규 표현 검색 옵션

종 류	설 명
*	0회 이상 일치하는 문자 검사
+	1회 이상 일치하는 문자 검사
i	대소문자 구분없이 검사
?	0회 또는 1회, 규칙에 맞는 문자가 있는지 검사
g	옵션에 g가 없으면 왼쪽부터 규칙에 일치하는 문자 한 개만 찾지만, g를 넣으면 규칙에 맞는 모든 문자를 찾는다.
m	데이터의 행이 바뀌어도 규칙에 맞는 문자를 찾는다.
\$	문자열의 끝부분에 규칙에 맞는 문자를 찾는다.
^	문자열 앞에서부터 규칙에 일치하는 문자를 찾는다. 또는 부정을 나타낸다.
\d	숫자 하나를 찾는다.
\D	숫자가 아닌 문자 하나를 찾는다.
\s	공백 문자(스페이스)를 찾는다.
[0-9][A-Z][a-z]	숫자, 대문자, 소문자 를 찾는다.
\w	알파벳, 숫자, 밑줄(_) 기호를 찾는다.
\W	알파벳, 숫자, 밑줄(_) 기호를 제외한 문자를 찾는다.
alb	a 또는 b 인지를 검사할 때 사용

❖ 자바스크립트 내장 객체

◆ 정규 표현 객체

* 정규 표현 함수

종 류	설 명
test()	지정된 규칙에 맞는 단어가 있으면 true 반환
exec()	지정된 규칙에 맞는 단어가 있으면 해당 단어를 반환

❖ 정규 표현 객체

```
<script type="text/javascript">
    var str="Html Css Jquery ";
    var reg1=/css/;
        // var reg1=new RegExp("css")
    var result_1=reg1.test(str);
        /* 표현식에 맞으면 true 반환 */
    document.write(result_1,"<br />");
    var reg2=/css/i;
        // var reg2=new RegExp("css","i")
        /* i : 영문 대소문자를 구분하지 않는다 */
    var result_2=reg2.test(str);
    document.write(result_2,"<br />");
</script>
```

js_regex01.html

❖ 정규 표현 객체

js_regex02.html

```
<script type="text/javascript">
    var userName=prompt("당신의 이름은?");
    reg1=/^[가-힣]{2,5}$/;
    while(true){
        if(reg1.test(userName)) break;

        alert("이름 입력 형식이 잘못되었습니다!");
        userName=prompt("당신의 이름은?");
    }

    var userCell=prompt("당신의 핸드폰 번호는?");
    reg2=/^010|016|011)\d{3,4}\d{4}$/;

    while(true){
        if(reg2.test(userCell)) break;
        alert("핸드폰 입력 형식이 잘못되었습니다!");
        userCell=prompt("당신의 핸드폰 번호는?");
    }

    var userEmail=prompt("당신의 이메일은?");
    reg3=/^\w{5,12}@[a-z]{2,10}[\.][a-z]{2,3}[\.]?[a-z]{0,2}$/;

    while(true){
        if(reg3.test(userEmail)) break;
        alert("이메일 입력 형식이 잘못되었습니다!");
        userEmail=prompt("당신의 이메일은?");
    }

    document.write(userName,"<br />");
    document.write(userCell,"<br />");
    document.write(userEmail,"<br />");
</script>
```

❖ 자바스크립트 내장 객체

◆ Date 객체

- * 날짜와 시간을 다루는 객체
- * Date 객체 생성 방법

```
var 변수이름 = new Date();
var 변수이름 = new Date(1/1000초);
var 변수이름 = new Date(날짜 문자열);
var 변수이름 = new Date(년도, 월, 날짜, 시간, 분, 초, 1/1000초);
```

```
<script>
    // 기본 생성자
    document.getElementById("d1").innerHTML = new Date();
    // 1970.01.01 이후의 밀리초 계산
    document.getElementById("d2").innerHTML = new Date(1491803527400);
    // 문자열 날짜
    document.getElementById("d3").innerHTML=new Date("October 15, 2018 06:18:07");
    // 주의사항 : 월(month)은 0부터 시작
    // 날짜 지정
    document.getElementById("d4").innerHTML=new Date(2018, 11, 25, 18, 30, 29);
</script>
```

js_date01.html

❖ 자바스크립트 내장 객체

◆ Date 객체 내장 함수

구분	메소드	속성 정보
반환 메소드	getDate()	1~31 날짜 반환
	getDay()	0~6 요일 반환(0 : 일요일, 1 : 월요일 ...)
	getFullYear()	연도 반환
	getHours()	0~23 시간 반환
	getMillisecond()	0~999 밀리초 반환
	getMinutes()	0~59 초 반환
	getMonth()	0~11 월 반환
	getSecond()	0~59 초 반환
설정 메소드	setDate()	1~31 날짜 설정
	setDay()	0~6 요일 설정(0 : 일요일, 1 : 월요일 ...)
	setFullYear()	연도 설정
	setHours()	0~23 시간 설정(시간, 분, 초, 밀리초)
	setMillisecond()	0~999 밀리초 설정
	setMinutes()	0~59 초 설정
	setMonth()	0~11 월 설정
	setSecond()	0~59 초 설정

❖ Date 객체 내장 함수

js_datefunc01.html

```
<body>
  <p id="d1"></p>
  <p id="d2"></p>
  <p id="d3"></p>
  <p id="d4"></p>
  <script>
    var today=new Date();
    document.getElementById("d1").innerHTML=today.getFullYear() + "년";
    document.getElementById("d2").innerHTML=today.getMonth()+1 + "월 "
      + today.getDate() + "일";
    document.getElementById("d3").innerHTML=today.getHours() + "시 "
      + today.getMinutes() + "분 " +
      today.getSeconds() + "초";
    document.getElementById("d4").innerHTML="1970년 1월 1일 이후 현재까지 몇 초가 지났나요?<p>" +
      + today.getTime() + "ms가 지났습니다.";
  </script>
</body>
```

❖ Date 객체 내장 함수

```
js_datefunc02.html
<body>
  <p id="d1"></p>
  <p id="d2"></p>
  <p id="d3"></p>
  <p id="d4"></p>
  <p id="d5"></p>
  <p id="d6"></p>
  <p id="d7"></p>
  <p id="d8"></p>
  <p id="d9"></p>
  <script>
    var today=new Date();
    document.getElementById("d1").innerHTML=today.toDateString();
    document.getElementById("d2").innerHTML=today.toISOString();
    document.getElementById("d3").innerHTML=today.toJSON();
    document.getElementById("d4").innerHTML=today.toLocaleDateString();
    document.getElementById("d5").innerHTML=today.toLocaleTimeString();
    document.getElementById("d6").innerHTML=today.toLocaleString();
    document.getElementById("d7").innerHTML=today.toString();
    document.getElementById("d8").innerHTML=today.toTimeString();
    document.getElementById("d9").innerHTML=today.toUTCString();
  </script>
</body>
```

❖ Date 객체 활용

```
js_date02.html
<body>
  <div id="digClock"></div>
  <script>
    function digClock() {
      var today=new Date();
      var day=today.getMonth()+1 + "월 " + today.getDate() + "일 ";
      var time=today.getHours() + "시 " + today.getMinutes() + "분 " + today.getSeconds() + "초";
      document.getElementById("digClock").innerHTML=day+time;
      setTimeout('digClock()', 1000);
    }
    digClock();
  </script>
</body>
```

❖ 브라우저 제어를 위한 내장 객체

◆ window 객체

- * Javascript의 최상위 객체 - 따라서 모든 객체는 window 객체의 하위 객체로 존재한다.

★ 원칙적 내장객체 접근 법

window.내장객체이름.함수이름([파라미터]);

- * 모든 객체가 window 객체안에 내장되어있기 때문에 window 객체 명시를 생략 가능

★ window 객체 생략 내장객체 접근 법

내장객체이름.함수이름([파라미터]);

★ document 객체 사용 예 (파란색은 생략됨)

window.document.getElementById("id value");

❖ window 객체

◆ 브라우저의 창 제어 기능

- * 새로운 창 열기 - 탭 브라우징을 지원하는 웹 브라우저에서 탭을 여는 기능으로 동작

```
window.open("페이지 URL");
```

```
window.open("페이지 URL", "창이름", "옵션");
```

- * 현재 창 닫기 (window: 현재 창 , self : 관계연산자 자기자신)

```
window.close();
```

또는

```
self.close();
```

❖ window 객체

◆ 브라우저의 창 제어 기능

- * 팝업창 열기 - 창 이름을 지정한 경우
 - 반복 클릭하더라도 하나의 창을 계속 사용

```
window.open("페이지 URL", "창이름", "팝업창 옵션");
```

- * 팝업창 열기 - 창 이름을 지정하지 않은 경우
 - 반복 클릭하면 다른 팝업창이 열린다.

```
window.open('페이지 URL', '', '팝업창 옵션');
```

❖ window 객체

◆ 브라우저의 창 제어 기능

- * 팝업창 열기 옵션값의 종류

옵션값	값 지정법	설명
toolbar	yes / no	툴바 아이콘의 표시 여부를 설정
location		주소 표시줄의 표시 여부 설정
status		상태 바의 표시 여부 설정
menubar		메뉴 표시줄의 표시 여부 설정
scrollbars		스크롤바의 표시 여부 설정
resizable		창의 크기를 조절 가능하게 할지 여부를 설정
width	픽셀값	창의 폭을 지정
height		창의 높이를 지정

fullscreen 등의 추가적인 용어가 있으며 scrollbars, resizable 등은 지정을 하지 않을 경우 열려야 하는 창이 팝업사이즈보다 더 클경우 상하 및 좌우스크롤은 자동으로 생성

- * 팝업창 열기 옵션 적용 예

```
window.open('http://www.naver.com', 'mywin', 'width=300, height=500, scrollbars=no,  
toolbar=no, menubar=no, status=no, location=no');
```

❖ window 객체 이벤트처리

js_window02.html

```
<body>
    <h1>window 객체</h1>
    <h3>open 메소드 확인</h3>
    <div>
        <a href="#" onclick="open1(); return false;">새 창 열기</a>
        <br/>
        <a href="#" onclick="open2(); return false;">팝업 창 열기(1)</a>
        <br/>
        <!-- 파라미터를 포함한 함수의 호출 -->
        <a href="#" onclick="open3('http://www.naver.com'); return false;">naver 창 열기(2-1)</a>
        <br/>
        <a href="#" onclick="open3('http://www.daum.net'); return false;">Daum 창 열기(2-2)</a>
        <br/>
    </div>
</body>
```

❖ window 객체 이벤트처리

```
<script type="text/javascript">
    function open1() {
        /* 새 창(혹은 탭) 띄우기 */
        window.open('01-open.html');
    }
    function open2() {
        /* 클릭할 때마다 창이 새로 열리는 팝업창 */
        window.open('01-open.html', "", 'width=300, height=500, scrollbars=no, toolbar=no,
                    menubar=no, status=no, location=no');
    }
    function open3(url) {
        /* 한번 생성된 팝업창을 지속적으로 재 사용하는 팝업창 */
        // 팝업의 주소를 파라미터로 전달받는다.
        window.open(url,'mywin','width=500, height=300, scrollbars=no, toolbar=no, menubar=no,
                    status=no, location=no');
    }
</script>
```

js_window02.html

❖ 브라우저 제어를 위한 내장 객체

◆ location 객체

- * 웹브라우저의 주소표시줄에 표시되는 URL로부터 다양한 정보를 추출할 수 있는 속성값들을 가지고 있다.
- * **location** 객체 기본 속성

속성 이름	설명
href	문서의 URL주소(주소 표시줄에 표시되는 URL 전체를 의미)
host	주소표시줄의 URL에서 호스트이름과 포트를 조회 (예] <u>www.iedu.or.kr:8080</u>)
hostname	호스트 이름을 조회한다. 일반적으로 URL에서 도메인을 조회할 수 있다.
hash	<u>앵커</u> 이름을 조회한다. URL에 “#” 기호와 함께 표시되는 단어를 의미
pathname	디렉터리 이하 경로를 조회
port	포트 번호를 조회
protocol	프로토콜의 종류를 조회 - 웹 주소에서 “http:” 혹은 “https:”를 조회
search	URL에 포함된 파라미터를 조회

❖ location 객체 이벤트처리

js_location01.html

```
<head>
<meta charset="utf-8" />
<script type="text/javascript">
    document.write("<p>문서의 URL주소: " + location.href + "</p>");
    document.write("<p>호스트 이름과 포트: " + location.host + "</p>");
    document.write("<p>호스트 컴퓨터 이름: " + location.hostname + "</p>");
    document.write("<p>앵커이름: " + location.hash + "</p>");
    document.write("<p>디렉토리 이하 경로: " + location.pathname + "</p>");
    document.write("<p>포트번호 부분: " + location.port + "</p>");
    document.write("<p>프로토콜 종류: " + location.protocol + "</p>");
    document.write("<p>URL 조회부분: " + location.search + "</p>");
</script>
</head>
<body>
    <a href="js_location.html?a=1&b=2#top">이 링크로 다시 실행하세요.</a>
</body>
```

❖ location 객체 이벤트처리

js_location02.html

```
<head>
<meta charset="utf-8" />
<script type="text/javascript">
    function goNaver() {
        if (confirm("정말 네이버로 이동하시겠습니까?")) {
            location.href= "http://www.naver.com";
        }
    }
</script>
</head>
<body>
    <input type="button" value="네이버로 이동하기" onclick="goNaver()" />
</body>
```

❖ location 객체 이벤트처리

```
<head>
<meta charset="utf-8" />
<script type="text/javascript">
    /** 두 수 사이의 난수를 리턴하는 함수 */
    function random(n1, n2) {
        return parseInt(Math.random() * (n2 - n1 + 1)) + n1;
    }
    /** 5자리의 인증번호를 id값이 “auth”인 요소에게 출력 */
    function authNo() {
        var value = "";
        for (var i = 0; i < 5; i++) {
            value += random(0, 9);
        }
        document.getElementById("auth").innerHTML = value;
    }
    /** 페이지 새로 고침 */
    function refresh() {
        location.reload();
    }
</script>
</head>
<!-- 페이지 최초 로딩시, authNo() 함수 호출 -->
<body onload="authNo()">
    <p>
        <!-- strong 태그 안이 자바스크립트 출력 부분 -->
        고객님의 인증번호는 <strong id="auth">00000</strong>입니다.
    </p>
    <!-- 페이지 새로고침 이벤트 호출 -->
    <input type="button" value="인증번호 새로 받기" onclick="refresh()" />
</body>
```

js_location03.html

❖ 브라우저 제어를 위한 내장 객체

◆ history 객체

- * 브라우저의 ‘뒤로’, ‘앞으로’ 버튼의 기능을 수행하는 객체로 **back()** 함수와 **forward()** 함수를 내장
- * 이전 페이지로 이동

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

- * 다음 페이지로 이동

```
histroy.forward();
```

❖ history 객체 이벤트처리

js_history01.html

```
<body>
  <h1>History 객체</h1>
  <a href="js_history02.html">페이지 이동</a><br />
  <a href="#" onclick="history.forward(); return false;">앞 페이지로 이동</a>
</body>
```

js_history02.html

```
<body>
  <h1>History 객체</h1>
  <a href="#" onclick="history.back(); return false;">이전 페이지로 이동</a>
</body>
```

❖ 브라우저 제어를 위한 내장 객체

◆ navigator 객체

- * 브라우저의 정보를 조회하기 위한 속성들을 가지고 있는 객체
- * 하나의 웹 서비스로 다양한 브라우저를 지원할 수 있도록 처리하기 위한 기준이 되는 정보를 추출할 수 있다.
- * **navigator** 객체의 주요 속성

속성 이름	설명
appName	브라우저의 이름
appCodeName	브라우저의 코드명
platform	브라우저가 설치된 시스템(클라이언트의)의 환경
userAgent	웹 브라우저의 종류와 버전 (가장 포괄적)
appVersion	웹 브라우저의 버전

❖ navigator 객체

js_navigator01.html

```
<script type="text/javascript">
    var info = "<h1>웹 브라우저 정보 확인</h1>";
    info += "<p>브라우저 이름 : " + navigator.appName + "</p>";
    info += "<p>브라우저 코드명 : " + navigator.appCodeName + "</p>";
    info += "<p>플랫폼 정보 : " + navigator.platform + "</p>";
    info += "<p>사용자 정보 : " + navigator.userAgent + "</p>";
    info += "<p>브라우저 버전 : " + navigator.appVersion + "</p>";
    document.write(info);
</script>
```

❖ navigator 객체

```
<script type="text/javascript">
    /* 모바일 브라우저이면 true, 그렇지 않으면 false를 리턴하는 사용자 정의 함수 */
    function isMobile() {
        var tmpUser = navigator.userAgent;
        var isMobile = false;
        // userAgent값에 iPhone, iPad, iPod, Android 라는 문자열이 하나라도 검색되면, 모바일로 간주한다.
        if (tmpUser.indexOf("iPhone") > 0 || tmpUser.indexOf("iPad") > 0 || tmpUser.indexOf("iPod") > 0 || tmpUser.indexOf("Android ") > 0) {
            isMobile = true;
        }
        return isMobile;
    }
    var isMobileWeb = isMobile();
    if (isMobileWeb) {
        document.write("<h1>모바일 웹 브라우저로 접속하셨습니다.</h1>");
    } else {
        document.write("<h1>PC용 웹 브라우저로 접속하셨습니다.</h1>");
    }
</script>
```

js_navigator02.html

❖ navigator 객체

```
<script type="text/javascript">
    /* 브라우저의 이름을 리턴하는 함수 */
    function getWebBrowserName() {
        // userAgent값을 모두 소문자로 변환하여 변수에 대입
        var agt = navigator.userAgent.toLowerCase();
        // 특정 브라우저의 이름이 검색되는지 여부를 판별하여
        // 특정 이름이 검색될 경우, 브라우저 이름을 의미하는 문자열을 리턴
        if (agt.indexOf("chrome") != -1) {
            return 'Chrome';
        } else if (agt.indexOf("opera") != -1) {
            return 'Opera';
        } else if (agt.indexOf("firefox") != -1) {
            return 'Firefox';
        } else if (agt.indexOf("safari") != -1) {
            return 'Safari';
        } else if (agt.indexOf("skipstone") != -1) {
            return 'SkipStone';
        } else if (agt.indexOf("msie") != -1 || agt.indexOf("trident") != -1) {
            return 'Internet Explorer';
        } else if (agt.indexOf("netscape") != -1) {
            return 'Netscape';
        } else {
            return "Unknown";
        }
    }
    document.write("<h1>" + getWebBrowserName() + "</h1>");
</script>
```

js_navigator03.html

❖ 브라우저 제어를 위한 내장 객체

◆ screen 객체

- * 장치의 디스플레이 정보를 조회할 수 있는 변수들을 내장한 객체
 - * 함수를 내장하고 있지 않다.
-
- * navigator 객체의 주요 속성

속성 이름	설명
availHeight	화면 높이
availWidth	화면 너비
colorDepth	색상 수
height	픽셀당 높이
width	픽셀당 너비
pixelDepth	픽셀당 비트수 (IE9 미만 버전은 지원 안함)

❖ screen 객체

```
<script type="text/javascript">
    /* screen 객체의 정보 조회 */
    document.write("<h1>화면 높이 : " + screen.availHeight + "</h1>");
    document.write("<h1>화면 너비 : " + screen.availWidth + "</h1>");
    document.write("<h1>색상 수 : " + screen.colorDepth + "</h1>");
    document.write("<h1>픽셀당 높이 : " + screen.height + "</h1>");
    document.write("<h1>픽셀당 너비 : " + screen.width + "</h1>");
    document.write("<h1>픽셀당 비트수(IE는 지원안함) : " + screen.pixelDepth + "</h1>");

</script>
```

js_screen01.html

```
<script type="text/javascript">
    /* 화면의 중심 좌표를 가로/세로 형태의 배열로 리턴하는 함수 */
    function getCenterPixel() {
        var center = new Array(
            parseInt(screen.availWidth/2), parseInt(screen.availHeight/2)
        );
        return center;
    }
    var screenCenter = getCenterPixel();
    document.write("<h1>모니터 화면의 중심 좌표: x=" + screenCenter[0] + "px, y=" +
screenCenter[1] + "px</h1>");
```

js_screen02.html

❖ 브라우저 제어를 위한 내장 객체

◆ document 객체

- * Javascript에서 가장 핵심적인 객체
- * HTML 문서의 구조나 내용을 제어하기 위한 기본 기능을 가지고 있다.
- * 하위 객체로 image, form 등을 가지고 있다.

❖ 특정 HTML 요소를 객체 형태로 가져오기

```
var 변수이름 = document.getElementById("아이디이름");
var 변수이름 = document.getElementsByClassName('클래스이름'); - 배열로 반환
... css 선택자를 사용할 수 있다.
```

❖ 특정 HTML 요소 내용 제어

```
변수이름 = document.getElementById('아이디이름').innerHTML; // 내용 읽기
document.getElementById('아이디이름').innerHTML = 데이터; // 내용 쓰기
```

❖ document 객체

◆ HTML 요소의 CSS 제어

```
document.getElementById("아이디이름").style.CSS속성 = "속성값";
```

❖ 배경 관련 속성

- Javascript의 CSS 대응 속성 이름은 CSS 속성에서 “-“ 가 붙는 경우 빼고 뒷 단어 첫글자만 대문자로 해준다.
(카멜(camel)표기법을 따른다.)

Javascript 속성 이름	CSS 속성 이름	설명
backgroundColor	background-color	배경 색상 지정
backgroundImage	background-image	배경 이미지 경로 지정
backgroundPosition	background-position	배경 이미지 위치 지정
backgroundRepeat	background-repeat	배경 이미지 반복 속성 지정

❖ document 객체

◆ image 객체

❖ image 객체 기본 속성

속성 이름	설명
src	객체가 표시하고 있는 이미지 파일의 경로 설정
width	객체가 표시하고 있는 이미지의 가로 너비 지정
height	객체가 표시하고 있는 이미지의 세로 높이 지정

❖ image 객체

js_image01.html

```
<style type="text/css">
    /* 목록 정의 초기화 및 목록 박스 좌측 배치 */
    .thumbnail {
        padding: 0; margin: 0;
        list-style: none;
        width: 120px; float: left;
    }
    /* 목록의 각 항목에 대한 크기 및 여백 설정 */
    .thumbnail li {
        width: 100px; height: 100px; padding: 5px 10px;
    }
    /* 썸네일 이미지의 크기 설정 */
    .thumbnail img {
        width: 100px; height: 100px;
    }
    /* 큰 이미지 영역의 배치와 크기, 여백 설정 */
    .view {
        float: left; width: 500px; height: 320px; padding: 5px 0;
    }
    /* 큰 이미지의 크기 설정 */
    .view img {
        width: 500px; height: 320px;
    }
</style>
</head>
<body>
    
    <ul class="thumbnail">
        <li><a href="#" onclick="setImage(0); return false;"></a></li>
        <li><a href="#" onclick="setImage(1); return false;"></a></li>
        <li><a href="#" onclick="setImage(2); return false;"></a></li>
    </ul>
    <div class="view">
        
    </div>
</body>
```

❖ image 객체

js_image01.html

```
<script type="text/javascript">
    /* 링크에 의해서 호출될 함수 */
    function setImage(index) {
        // 이미지의 경로를 담고 있는 배열
        var image_list = [
            'img/1.jpg',
            'img/2.jpg',
            'img/3.jpg'
        ];
        // 이미지 요소의 객체화
        var image = document.getElementById("target");
        // 객체의 src속성에 배열의 값을 중에서 파라미터로 전달된 위치의 값을 설정한다.
        image.src = image_list[index];
    }
</script>
```

❖ document 객체

◆ form 객체

* 입력 태그들의 입력 제어

❖ form 객체의 접근은 name 값으로 하므로 name속성을 지정해주자.

* id 값에 의한 객체 획득 방법

```
변수 = document.getElementById("id이름");
```

* name값에 의한 객체 획득 방법

```
변수 = document.form-name값;
```

❖ document 객체

◆ form 객체

* 입력 태그들의 입력 내용을 웹 프로그램에 전달하고 입력 내용을 제어

❖ form 객체의 접근은 name 값으로 하므로 name속성을 지정해주자.

* id 값에 의한 객체 획득 방법

```
변수 = document.getElementById("id이름");
```

* name값에 의한 객체 획득 방법

```
변수 = document.form-name값;
```

❖ document 객체

◆ form 객체

- * 입력값 처리 - value 속성으로 읽고 쓴다.

```
변수 = document.form_name.태그name.value; // 읽기
```

```
document.form_name.태그name.value = 데이터; // 쓰기
```

❖ value 속성은 input 태그의 입력 여부 체크에도 사용할 수 있다.

```
var myform = document.form1;
var username = myform.user_name;

if (!username.value){
    alert("내용을 입력하세요");
}
```

❖ document 객체

◆ form 객체

- * 선택 항목 처리
 - select 태그에서 사용
 - selectedIndex 속성 사용 (index는 0 부터 시작)
- * 선택 항목 Index 얻기

```
var no = document.form_name.태그name.selectedIndex;
```

- * 선택 항목 데이터 얻기

```
var no = document.form_name.태그name.selectedIndex;  
var str = document.form_name.태그name[no].value;
```

- * 선택 여부 검사

```
<form name="form1">  
  <select name="foo">  
    <option>##선택하세요!##</option>  
    <option>항목 1</option>  
    <option>항목 2</option>  
  </select>  
</form>
```

```
var myform = document.form1;  
if(myform.foo.selectedIndex < 1){  
  ## 처리내용 ##  
}
```

❖ document 객체

◆ form 객체

- * 라디오 버튼과 체크박스의 체크 상태
 - checked 속성을 사용 - 체크되면 true
- * focus 지정 - focus() 함수로 강제 지정

```
document.form1.foo.focus();
```

- * 작성된 내용 리셋 - reset() 함수 사용

```
document.form1.reset();
```

- * 작성된 내용 전송 - submit() 함수 사용

```
document.form1.submit();
```

- * 작성된 내용 전송 하기전 입력 여부 검사 - onsubmit() 이벤트 함수 사용

```
<form name="form1" onsubmit="함수이름(); return false;">
```

❖ JSON 표기법

- * 경량의 데이터 교환 형식
- * 구문형식은 Javascript를 따르지만 언어로서 독립적
- * 프로그래밍언어간 데이터 교환을 위해 가장 널리 쓰이는 표기법
- * Javascript 문법으로 사용
- * 자체가 하나의 데이터로 사용

◆ 기본 형식

```
var 객체이름 = { 이름: 값, 이름: 값, . . . };
```

```
var 객체이름 = { };  
객체이름.이름 = “새로운값”;
```

❖ Array 객체

◆ Method 종류

종 류	설 명
join("연결문자")	배열 객체에 데이터를 연결 문자 기준으로 1개의 문자형 데이터로 반환
reverse()	배열 객체의 데이터의 순서를 거꾸로 바꾼 후 반환
sort()	배열 객체의 데이터를 오름차순으로 정렬
slice(index1, index2)	배열 객체에 데이터중 원하는 인덱스 구간만큼 잘라서 배열 객체로 반환
splice()	배열 객체에 지정 데이터를 삭제하고 그 구간에 새 데이터를 삽입
concat()	2개의 배열 객체를 하나로 결합
pop()	저장된 데이터중 마지막 인덱스에 저장된 데이터 삭제
push(new data)	마지막 인덱스에 새 데이터를 삽입
shift()	첫 번째 인덱스에 저장된 데이터를 삭제
unshift(new data)	가장 앞의 인덱스에 새 데이터 삽입

◆ Property 종류

종 류	설 명
length	배열에 지정된 총 데이터의 개수를 반환

❖ Array 객체

◆ splice()

- * 배열의 중간 위치에 데이터를 다른 데이터로 변경하는 기능을 가진 함수

```
splice(시작위치, 개수, 데이터1, 데이터2, …);
```

- * 시작 위치부터 지정한 개수 만큼의 데이터를 지정한 데이터로 변경한다.
- * 개수와 뒤에있는 데이터의 개수는 아무 관계가 없다.
- * 데이터는 없어도 된다. - 중간의 데이터를 삭제하는 기능의 효과가 된다.

❖ window 객체

◆ Method 종류

종 류	설 명
open()	새 창을 열 때 사용
alert()	경고 창을 띄울 때 사용 - 버튼 한개
prompt()	입력 창을 띄울 때 사용 - 버튼 두개 (확인/취소) / 변수에 담아서 사용
confirm()	확인/취소 창을 띄울 때 사용 - 버튼 두개(확인/취소) / 반환값 : true/false
moveTo()	창의 위치를 이동 시킬 때 사용
resizeTo()	창의 크기를 변경 시킬 때 사용
setInterval()	일정 간격으로 지속적으로 실행문을 실행 시킬 때 사용
setTimeout()	일정 간격으로 한번만 실행문을 실행시킬 때 사용

❖ window 객체 method

- * **open()** - 새 창을 띄울 때 사용

```
[window.]open("url 경로", "창 이름", "옵션");
```

- 팝업창 열기 옵션값의 종류

옵션값	값 지정법	설명
toolbar		툴바 아이콘의 표시 여부를 설정
location		주소 표시줄의 표시 여부 설정
status		상태 바의 표시 여부 설정
menubar	yes / no	메뉴 표시줄의 표시 여부 설정
scrollbars		스크롤바의 표시 여부 설정
resizable		창의 크기를 조절 가능하게 할지 여부를 설정
width	픽셀값	창의 폭을 지정
height		창의 높이를 지정

❖ window 객체 method

- * **alert()** - 경고 창을 띄울 때 사용

```
[window.]alert("메세지");
```

- * **prompt()** - 질의응답 창을 띄울 때 사용

```
[window.]prompt("질의내용", "기본답변");
```

- * **confirm()** - 확인/취소 창을 띄울 때 사용

```
[window.]confirm("질의내용");
```



버튼클릭 반환값은 true / false

❖ window 객체 method

- * **moveTo()** - 창의 위치를 이동시킬 때

```
var 변수이름 = open("url", "창이름", "옵션");  
변수이름.moveTo(x 위치값, y 위치값);
```

❖ **픽셀 단위 입력**

- * **resizeTo()** - 창의 크기 변경

```
var 변수이름 = open("url", "창이름", "옵션");  
[window.]resizeTo(창 너비, 창 높이);
```

❖ **픽셀 단위 입력**

❖ window 객체 method

- * **setInterval()** - 일정 시간 간격으로 실행문 실행

```
var 변수이름 = setInterval("실행문(함수이름)", 시간간격(1/1000초));
```

- * **clearInterval()** - 일정 시간 간격으로 실행문 실행을 취소

```
clearInterval(변수이름);
```

- * **setTimeout()** - 일정 간격으로 실행문을 한 번만 실행

```
var 변수이름 = setTimeout("실행문(함수이름)", 시간간격(1/1000초));
```

- * **clearTimeout()** - 일정 시간 간격으로 한 번 실행 실행문 실행을 취소

```
clearTimeout(변수이름);
```

❖ **history 객체 method**

◆ Method 종류

종 류	설 명
history.back()	이전 방문 페이지로 이동
history.forward()	다음 방문 페이지로 이동
history.go(이동숫자)	이동 숫자만큼 이동 - 이동 숫자가 -2이면 2단계 이전 페이지로 이동

◆ Property 종류

종 류	설 명
length	배열에 지정된 총 데이터의 개수를 반환

❖ DOM(Document Object Model) 객체

◆ 선택자 종류

구분	종 류	설 명
직접 선택자	<code>document.getElementById("아이디이름")</code>	아이디로 요소 선택
	<code>document.getElementsByTagName("태그")</code>	태그를 이용해 선택 - 배열 객체로 반환
	<code>document.formName.inutName</code>	폼 요소에 name 속성을 부여하고 그것을 이용해 선택
인접 관계 선택자	<code>parentNode</code>	선택 요소의 부모(상위) 요소 선택
	<code>childNodes</code>	선택 요소의 모든 자식(하위) 요소 선택 - 배열 객체로 반환
	<code>children</code>	선택 요소의 자식(하위) 요소 태그만 선택 - 배열 객체로 반환
	<code>firstChild</code>	선택 요소의 첫번째 자식(하위) 요소만 선택
	<code>previousSibling</code>	선택 요소의 이전에 오는 형제 요소 선택
	<code>nextSibling</code>	선택 요소 다음에 오는 형제 요소 선택

❖ CSS 선택자와 약간의 차이가 있으니 확인하세요!

❖ DOM(Document Object Model) 객체

◆ Event Handler

- * 기본형

```
요소.이벤트 = function() {      실행문; }
```

❖ callback 함수

- * 콜백 함수는 파라메터를 통해 다음 실행지점을 지시하는 함수를 전달한다.
- * 콜백 함수를 전달받은 함수는 실행 지점 마지막에 호출자 측으로 반환되는 대신 이 콜백 함수로 다시 한 번 진입 한다. 즉 재귀함수와 비슷하게 동작한다.
- * 호출자가 결과값을 반환받는다.
- * 해당 함수가 실행하는 그 순간에 호출자로부터의 동기화가 끊긴다.

❖ DOM(Document Object Model) 객체

◆ EventListener

* 자바 스크립트에서 이벤트를 설치하는 방법 - 주로 동적(변동이 있는)인 이벤트 설치에 사용한다.

요소.addEventListener("이벤트이름", 실행함수이름);

예]

```
<script>
    var      temp = document.getElementById("???");
    temp.addEventListener("click", abc);
</script>
```

요소.addEventListener("이벤트이름", function() {
 실행내용;
});

❖ DOM(Document Object Model) 객체

◆ EventListener

- * 하나의 요소에 여러 이벤트를 동시에 처리할 수 있다.
- * 같은 이벤트에 여러 함수를 동시에 처리할 수 있다.- 이때는 등록된 함수가 순차적으로 실행된다.
- * **this**를 사용할 수 있다.- 이때 **this**는 이벤트가 일어난 요소 자체를 의미하는 예약된 변수이다.
- * 자바스크립트를 이용한 이벤트 등록에 있어서 함수를 내부적으로 만들어서 사용할 수 있다.
- * 이벤트 처리 순서 변경 - HTML의 특성상 여러 요소가 겹쳐서 사용할 필요가 있다.

예] <div>???<p>???</p>???</div>

- * 만약 중첩된 요소에 이벤트가 동시에 설치되면 이벤트 처리 순서는
안쪽 요소가 먼저 처리되고 바깥 요소가 나중에 실행된다.
- * 필요하다면 이벤트 처리 순서를 변경할 수 있다.

요소.addEventListener("이벤트이름", 함수이름, true/false);

- ★ **true** : 바깥 요소 먼저 실행
- ★ **false** : 안쪽 요소 먼저 실행 (기본값)

❖ DOM(Document Object Model) 객체

◆ removeEventListener

- * 불필요한 이벤트는 제거할 수 있다.

요소.removeEventListener("이벤트이름", 사용함수);

◆ attachEvent()

- * 이벤트 등록을 하는 함수로 addEventListener()에 해당

◆ detachEvent()

- * 이벤트 제거를 하는 함수로 removeEventListener()에 해당

★ IE 8버전 이전에는 위의 함수를 사용해서 이벤트 등록을 해야만 했었다.

★ 하지만 IE9 이후 버전 부터는 add, remove를 이용해서 이벤트 등록을 하도록 권장하고 있다.

❖ DOM(Document Object Model) 객체

◆ 사용자 브라우저 환경에 따른 이벤트 처리

* 클라이언트가 어떤 버전의 브라우저를 사용할지 모르므로....

```
var target = document.getElementById("???");
if(target.addEventListener) {
    // 현재 사용자가 웹 브라우저가 addEventListener을 제공하면...
    target.addEventListener("이벤트이름", 함수이름);
}
else if(target.attachEvent) {
    // 현재 사용자가 웹 브라우저가 attachEvent을 제공하면...
    target.attachEvent("이벤트이름", 함수이름);
}
```

❖ DOM(Document Object Model) 객체

◆ DOM Node

- * HTML 문서의 각각의 요소를 트리 구조식으로 만든것

◆ DOM Node에서 원하는 요소를 찾고 동적으로 DOM Node를 만들거나 삭제하는 방법

- * 태그 안에 기록된 실제 데이터도 하나의 Node로 간주

- * 실제 데이터를 알아내거나 수정하는 방법

종 류	설 명
innerHTML	강제로 데이터를 입력하거나 수정
firstChild.nodeValue	firstChild는 현재 노드에서 하위 노드를 지칭하는 예약어
childNode[?].nodeValue	childNode[?] 현재 노드에서 지정한 위치(?)의 자식 노드를 지칭하는 예약어

❖ **nodeValue - 실질적인 데이터를 의미하는 예약어**

❖ DOM(Document Object Model) 객체

◆ DOM Node를 이용한 동적 HTML 만들기

- * 처음에 서버가 클라이언트에게 응답하는 문서에는 존재하지 않는 내용을 필요한 순간에 새롭게 만들어서 사용하는 방법
- * 주로 동적 폼을 만들 때 사용

함 수	설 명
createElement	새로운 태그를 생성
createTextNode	새로운 데이터 node를 생성
appendChild	새로만든 태그를 필요한 위치에 추가
insertBefore	특정 요소 앞에 새로운 요소를 추가
removeChild	특정 요소를 삭제
replaceChild	특정 요소를 다른 요소로 변경 - 주로 innerHTML로 처리

❖ DOM(Document Object Model) 객체

◆ DOM Node를 이용한 동적 HTML 만들기

```
<script type="text/javascript">
    var my_div = null;
    var newDiv = null;
    function addElement() {
        // 새로운 div 엘리먼트 생성
        // 내용을 작성
        newDiv = document.createElement("div");
        newDiv.innerHTML = "<h1>안녕! 반가워!</h1>";
        // 생성된 엘리먼트를 추가
        my_div = document.getElementById("org_div1");
        document.body.insertBefore(newDiv, my_div);
    }
</script>
<body onload="addElement()">
<div id='org_div1'> 위의 문장은 동적으로 만들어 진 것입니다.</div>
</body>
```

js_domnode01.html