

UI 프론트엔드 프로그래밍 과정

# JavaScript

# JavaScript 함수

## 5. 함수

---

### ◆ 함수란?

- 코드의 집합
- 함수의 예 : `alert ( )`, `prompt ( )`

# 5.1 익명 함수

## ◆ 익명 함수

- 함수의 형태
- 괄호 내부에 코드를 넣음

---

```
var 함수 = function () { };
```

---

## 5.1 익명 함수

### ◆ 함수의 생성과 출력

- 함수의 생성과 출력

- 문자열처럼 보일 수 있지만 typeof 연산자를 사용하면 함수 자료형

```
<script>
    // 변수를 생성합니다.
    var 함수 = function () {
        var output = prompt('숫자를 입력해주세요. ', '숫자 ');
        alert(output);
    };

    // 출력합니다.
    alert(함수);
</script>
```



# 5.1 익명 함수

## ◆ 함수의 생성과 출력

- 익명함수 : `function ( ) { }` 형태는 함수이나 이름이 없음
- 선언적 함수 : 이름이 있는 함수
- 함수 호출 : 뒤에 괄호를 열고 닫음으로 코드를 실행

```
var 함수 = function () {  
    var output = prompt('숫자를 입력해주세요.', '숫자');  
    alert(output);  
};
```

함수()

함수()

## 5.2 선언적 함수

### ◆ 선언적 함수

- 선언적 함수 생성

---

```
function 함수() {  
  
}
```

---

---

```
var 함수 = function () { };
```

---

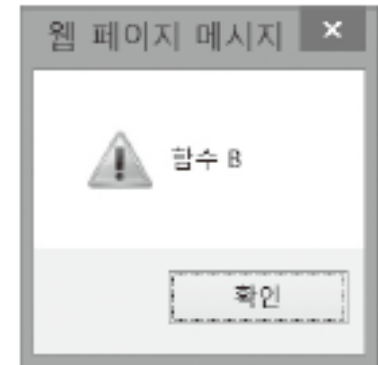
## 5.2 선언적 함수

### ◆ 선언적 함수

#### ■ 선언적 함수의 재정의 (1)

```
<script>
  function 함수() { alert('함수 A'); }
  function 함수() { alert('함수 B'); }
  함수();
</script>
```

```
<script>
  var 함수 = function () { alert('함수 A'); };
  var 함수 = function () { alert('함수 B'); };
  함수();
</script>
```





## 5.3 매개변수와 리턴값

### ◆ 매개변수와 리턴값

- 매개변수
  - 함수를 호출 할 때 괄호 안에 적는 것
- 리턴값
  - prompt( ) 함수를 사용하면 사용자가 입력한 문자열로 변환

```
prompt()
```

```
String prompt([String message], [String defaultValue])
```

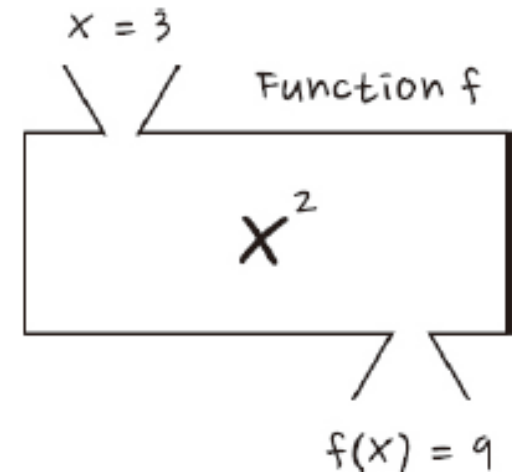
## 5.3 매개변수와 리턴값

### ◆ 매개변수와 리턴값으로 함수 만들기

- 모든 함수에 매개변수와 리턴값을 사용하지 않음
- **선택적** 매개변수와 리턴값 사용

```
function 함수이름(매개변수, 매개변수, 매개변수) {  
    // 함수 코드  
    return 리턴값;  
}
```

- input : 매개변수 / output 리턴값



## 5.4 매개변수

### ◆ Array( ) 함수

- Array( ) 함수의 형태

Array()

▲ 1 of 3 ▼ Array Array()

Array()

▲ 3 of 3 ▼ Array Array(arg1, arg2, [...])

함수 형태

Array( )

Array(number)

Array(any, ... , any)

설명

빈 배열을 만듭니다.

매개변수만큼의 크기를 가지는 배열을 만듭니다.

매개변수를 배열로 만듭니다.

## 5.5 가변 인자 함수

---

### ◆ 가변 인자 함수란?

- 매개변수의 개수가 변할 수 있는 함수
- 매개변수가 선언된 형태와 다르게 사용했을 때 매개변수를 모두 활용하는 함수를 의미
- 가변인자 함수의 예 : Array( ) 함수

## 5.5 가변 인자 함수

### ◆ sumAll( ) 함수

- **arguments**

- 자바스크립트 내부 변수의 기본으로 제공
- 매개변수의 배열

---

```
<script>
  function sumAll() {

    }
</script>
```

---

## 5.5 가변 인자 함수

### ◆ sumAll( ) 함수

- **arguments** 객체를 사용한 가변 인자 함수

---

```
<script>
    // 함수를 생성합니다.
    function sumAll() {
        var output = 0;
        for (var i = 0; i < arguments.length; i++) {
            output += arguments[i];
        }
        return output;
    }

    // 함수를 호출 및 출력합니다.
    alert(sumAll(1, 2, 3, 4, 5, 6, 7, 8, 9));
</script>
```

---

## 5.6 리턴값

### ◆ 리턴값

- 리턴 키워드

- 함수 실행 중 함수를 호출한 곳으로 돌아가라는 의미

---

```
<script>
  // 함수를 생성합니다.
  function returnFunction() {
    alert('문장 A');
    return;
    alert('문장 B');
  }

  // 함수를 호출합니다.
  returnFunction();
</script>
```

---

## 5.6 리턴값

### ◆ 리턴값

- 리턴 키워드 뒤에 값을 입력하지 않을 경우
  - 리턴하지 않으면 아무것도 들어가지 않음

---

```
<script>
  // 함수를 생성합니다.
  function returnFunction() {
    alert('문장 A');
    return;
    alert('문장 B');
  }

  // 함수를 호출합니다.
  var output = returnFunction();
  alert(typeof (output) + ' : ' + output);
</script>
```

---



## 5.8 콜백 함수

### ◆ 콜백 함수

- 매개변수로 전달하는 함수
- callTenTimes ( ) 함수 : 함수를 매개변수로 받아 해당 함수를 10번 호출

---

```
<script>
    // 함수를 선언합니다.
    function callTenTimes(callback) {
        // 10회 반복합니다.
        for (var i = 0; i < 10; i++) {
            // 매개변수로 전달된 함수를 호출합니다.
            callback();
        }
    }

    // 변수를 선언합니다.
    var callback = function () {
        alert('함수 호출');
    };

    // 함수를 호출합니다.
    callTenTimes(callback);
</script>
```

---

## 5.9 자바스크립트 내장 함수

---

- ◆ 내장 함수
  - 기본적으로 내장된 함수
  - alert ( ) 함수와 prompt ( ) 함수

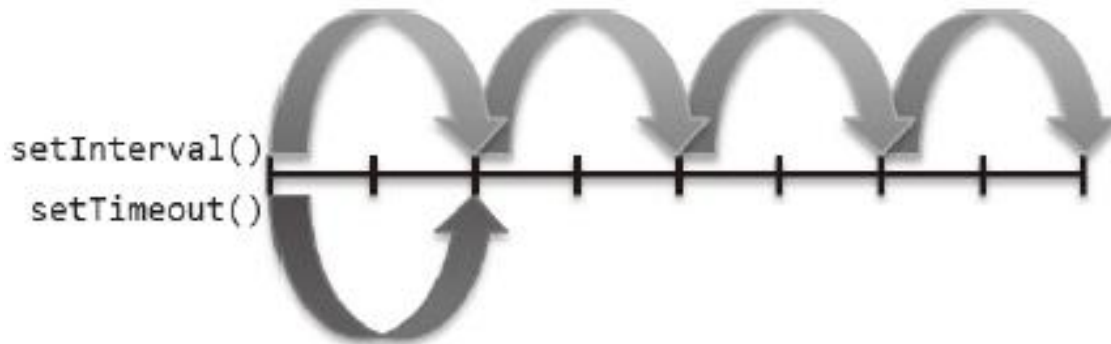
## 5.9 자바스크립트 내장 함수

### ◆ 타이머 함수

- 특정한 시간에 특정한 함수를 실행 가능하게 함

메서드 이름	설명
<code>setTimeout(function, millisecond)</code>	일정 시간 후 함수를 한 번 실행합니다.
<code>setInterval(function, millisecond)</code>	일정 시간마다 함수를 반복해서 실행합니다.
<code>clearTimeout(id)</code>	일정 시간 후 함수를 한 번 실행하는 것을 중지합니다.
<code>clearInterval(id)</code>	일정 시간마다 함수를 반복하는 것을 중단합니다.

- `setTimeout ( )` 메서드 : 특정한 시간 후에 함수를 한 번 실행
- `setInterval ( )` 메서드 : 특정한 시간마다 함수를 실행



## 5.9 자바스크립트 내장 함수

### ◆ 타이머 함수

- setTimeout ( ) 함수의 주의사항 : 특별히 없음
- setInterval ( ) 함수의 주의사항 : 지속적 자원의 소비
  - 해결 방법 : 타이머를 멈춤
  - clearTimeout ( ) 함수/clearInterval ( ) 함수를 사용

---

```
<script>
    // 1초마다 함수를 실행합니다.
    var intervalID = setInterval(function () {
        alert('<p>' + new Date() + '</p>');
    }, 1000);

    // 10초 후 함수를 실행합니다.
    setTimeout(function () {
        // 타이머를 종료합니다.
        clearInterval(intervalID);
    }, 10000);
</script>
```

---

## 5.9 자바스크립트 내장 함수

### ◆ 코드 실행 함수

- 자바스크립트는 문자열을 코드로 실행할 수 있는 특별한 함수를 제공
- `eval( )` 함수는 문자열을 자바스크립트 코드로 실행하는 함수

함수 이름	설명
<code>eval(string)</code>	<code>string</code> 을 자바스크립트 코드로 실행합니다.

## 5.9 자바스크립트 내장 함수

- ◆ 코드 실행 함수
  - 코드 실행 함수 예

---

```
<script>
    // 문자열을 생성합니다.
    var willEval = '';
    willEval += 'var number = 10;';
    willEval += 'alert(number);';

    // eval() 함수를 호출합니다.
    eval(willEval);
</script>
```

---

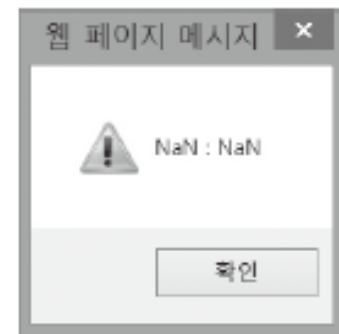
## 5.9 자바스크립트 내장 함수

### ◆ 숫자 변환 함수

함수 이름	설명
parseInt(string)	string을 정수로 바꾸어줍니다.
parseFloat(string)	string을 유리수로 바꾸어줍니다.

- 숫자 변환 함수의 예 : Number( ) 함수
- Number( ) 함수의 단점
- Number( ) 함수는 숫자로 바꿀 수 없으면 Nan로 변환함

```
<script>
    var won = '1000원';
    var dollar = '1.5$';
    alert(Number(won) + ' : ' + Number(dollar));
</script>
```



## 5.9 자바스크립트 내장 함수

### ◆ 숫자 변환 함수

- parseInt( ) 함수/parseFloat( ) 함수  
→ 변환할 수 있는 부분까지 모두 숫자로 변환

---

```
<script>
  var won = '1000원';
  var dollar = '1.5$';
  alert(parseInt(won) + ' : ' + parseInt(dollar));
  alert(parseFloat(won) + ' : ' + parseFloat(dollar));
</script>
```

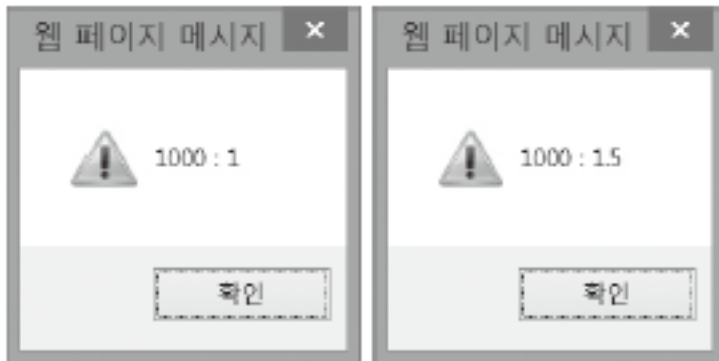
---



## 5.9 자바스크립트 내장 함수

### ◆ 숫자 변환 함수

- parseInt( ) 함수와 parseFloat( ) 함수의 차이



- parseInt( ) 함수와 parseFloat( ) 함수의 주의점
- parseInt( ) 함수의 두 번째 매개변수에 진법 입력  
→ 해당 진법의 수로 인식
- 자바스크립트에서 0,0X로 시작하면 10진수가 아닌 8진수,16진수로 변환

- parseInt('0x273') ⇒ 627
- parseInt('273') ⇒ 273
- parseInt('0273') ⇒ 187

- parseInt('FF', 16) ⇒ 255
- parseInt('52', 10) ⇒ 52
- parseInt('11', 8) ⇒ 9
- parseInt('10', 2) ⇒ 2

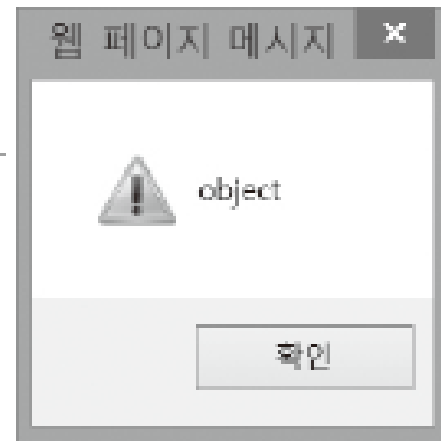
# JavaScript 객체

## 6. 객체

### ◆ 배열의 자료형

- 자바스크립트의 기본 자료형
  - 숫자, 문자열, 불, 함수, undefined
- 코드 실행 결과 : object가 '객체'

```
<script>  
    alert(typeof ([]));  
</script>
```



# 6.1 객체의 개요

## ◆ 배열의 선언

```
<script>  
  // 변수를 선언합니다.  
  var array = ['사과', '바나나', '망고', '딸기'];  
</script>
```

- 배열의 구성 : 인덱스와 요소
  - 배열 요소를 사용하려면 배열 이름 뒤에 인덱스로 접근

- array[0]    ⇨   사과
- array[2]    ⇨   망고

- 배열의 인덱스와 요소

인덱스	요소
0	사과
1	바나나
2	망고
3	딸기

## 6.1 객체의 개요

### ◆ 객체와 배열

- 배열은 객체를 기반으로 함
- 배열은 요소에 인덱스로 접근/객체는 요소에 키로 접근

```
<script>
  // 변수를 선언합니다.
  var product = {
    제품명: '7D 건조 망고',
    유형: '당절임',
    성분: '망고, 설탕, 메타중아황산나트륨, 치자황색소',
    원산지: '필리핀'
  };
</script>
```

키	속성
제품명	7D 건조 망고
유형	당절임
성분	망고, 설탕, 메타중아황산나트륨, 치자황색소
원산지	필리핀

## 6.1 객체의 개요

### ◆ 객체의 사용

- 객체 뒤에 대괄호를 사용하고 키를 표시 → 요소에 접근

- `product['제품명']`    ⇨    '7D 건조 망고'
- `product['유형']`    ⇨    '당절임'
- `product['성분']`    ⇨    '망고, 설탕, 메타중아황산나트륨, 치자황색소'
- `product['원산지']`    ⇨    '필리핀'

- 대괄호 외 일반적으로 사용하는 방법

- `product.제품명`    ⇨    '7D 건조 망고'
- `product.유형`    ⇨    '당절임'
- `product.성분`    ⇨    '망고, 설탕, 메타중아황산나트륨, 치자황색소'
- `product.원산지`    ⇨    '필리핀'

## 6.1 객체의 개요

### ◆ 객체의 사용

- 객체의 키  
→ 식별자 또는 문자열 모두 사용 가능

---

```
<script>
  // 변수를 선언합니다.
  var object = {
    'with space': 273,
    'with ~!@#$$%^&*()_+': 52
  };
</script>
```

---

- 식별자가 아닌 문자를 키로 사용 하려면 대괄호를 사용해야 함
  - object['widht space']                   ⇒ 273
  - object['with ~!@#\$\$%^&\*()\_+']   ⇒ 52

## 6.2 속성과 메서드

### ◆ 속성

- 요소 : 배열 내부에 있는 값
- 속성 : 객체 내부에 있는 값

---

```
var object = {  
  number: 273,  
  string: 'RintIantta',  
  boolean: true,  
  array: [52, 273, 103, 32],  
  method: function () {  
  
  }  
};
```

---



## 6.2 속성과 메서드

### ◆ 메서드

- 객체의 속성 중 함수 자료형인 속성
- 속성과 메서드의 구분
  - 객체 person : name 속성, eat 속성
  - eat 속성은 함수 자료형이므로 eat( ) 메서드라 부름

---

```
<script>
  // 변수를 선언합니다.
  var person = {
    name: '윤인성',
    eat: function (food) { }
  };

  // 메서드를 호출합니다.
  person.eat();
</script>
```

---

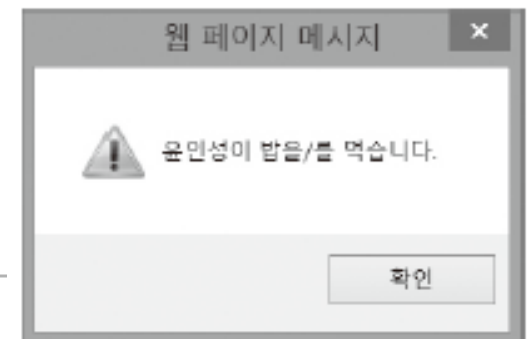
## 6.2 속성과 메서드

### ◆ 메서드

- this 키워드
  - 자기 자신이 가진 속성을 출력하고 싶을 때
  - 자신이 가진 속성임을 표시하는 방법

```
<script>
    // 변수를 선언합니다.
    var person = {
        name: '윤인성',
        eat: function (food) {
            alert(this.name + '이 ' + food + '을/를 먹습니다.');
        }
    };

    // 메서드를 호출합니다.
    person.eat('밥');
</script>
```



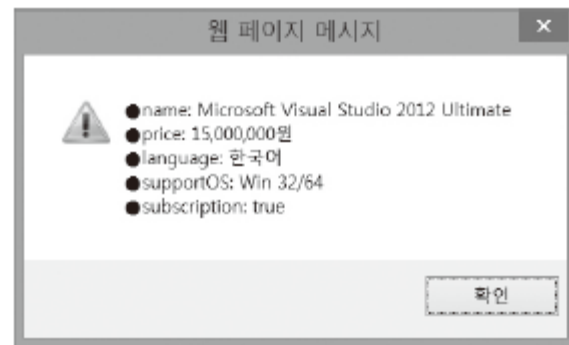
## 6.3 객체와 반복문

### ◆ 객체와 반복문

- 객체는 단순 for 반복문으로 객체의 속성을 살펴보는것이 불가능
- 객체의 속성을 모두 살펴보려면 for in 반복문 사용

```
<script>
    // 변수를 선언합니다.
    var product = {
        name: 'Microsoft Visual Studio 2012 Ultimate',
        price: '15,000,000원',
        language: '한국어',
        supportOS: 'Win 32/64',
        subscription: true
    };

    // 출력합니다.
    var output = '';
    for (var key in product) {
        output += '●' + key + ': ' + product[key] + '\n';
    }
    alert(output);
</script>
```



## 6.5 객체의 속성 추가와 제거

---

### ◆ 객체의 속성 추가와 제거

- 객체 생성 이후 속성을 추가하거나 제거
  - 동적으로 속성을 추가한다
  - 동적으로 속성을 제거한다

## 6.5 객체의 속성 추가

### ◆ 속성 추가

- 동적으로 속성 추가

---

```
<script>
  // 변수를 선언합니다.
  var student = {};

  // 객체에 속성을 추가합니다.
  student.이름 = '윤인성';
  student.취미 = '악기';
  student.특기 = '프로그래밍';
  student.장래희망 = '생명공학자';
</script>
```

---

## 6.5 객체의 속성 추가

### ◆ 속성 추가

```
// toString() 메서드를 만듭니다.
student.toString = function () {
    var output = '';
    for (var key in this) {
        // toString() 메서드는 출력하지 않게 합니다.
        if (key != 'toString') {
            output += key + '\t' + this[key] + '\n';
        }
    }
    return output;
};

// 출력합니다.
alert(student.toString());
</script>
```

---

## 6.6 객체와 배열을 사용한 데이터 관리

### ◆ 객체와 배열을 사용한 데이터 관리

- 추상화  
→ 현실에 존재하는 객체의 필요한 속성을 추출하는 작업
- 학생의 성적 총점과 평균을 계산하는 예제 작성

---

```
<script>
```

```
var student0 = { 이름: '윤인성', 국어: 87, 수학: 98, 영어: 88, 과학: 95 };  
var student1 = { 이름: '연하진', 국어: 92, 수학: 98, 영어: 96, 과학: 98 };  
var student2 = { 이름: '구지연', 국어: 76, 수학: 96, 영어: 94, 과학: 90 };  
var student3 = { 이름: '나선주', 국어: 98, 수학: 92, 영어: 96, 과학: 92 };  
var student4 = { 이름: '윤아린', 국어: 95, 수학: 98, 영어: 98, 과학: 98 };  
var student5 = { 이름: '윤명월', 국어: 64, 수학: 88, 영어: 92, 과학: 92 };  
var student6 = { 이름: '김미화', 국어: 82, 수학: 86, 영어: 98, 과학: 88 };  
var student7 = { 이름: '김연화', 국어: 88, 수학: 74, 영어: 78, 과학: 92 };  
var student8 = { 이름: '박아현', 국어: 97, 수학: 92, 영어: 88, 과학: 95 };  
var student9 = { 이름: '서준서', 국어: 45, 수학: 52, 영어: 72, 과학: 78 };
```

```
</script>
```

---

## 6.6 객체와 배열을 사용한 데이터 관리

### ◆ 객체와 배열을 사용한 데이터 관리

- 배열에 데이터 추가

---

```
<script>
```

```
    // 학생 정보 배열을 만듭니다.
```

```
    var students = [];
```

```
    students.push({ 이름: '윤인성', 국어: 87, 수학: 98, 영어: 88, 과학: 95 });
```

```
    students.push({ 이름: '연하진', 국어: 92, 수학: 98, 영어: 96, 과학: 98 });
```

```
    students.push({ 이름: '구지연', 국어: 76, 수학: 96, 영어: 94, 과학: 90 });
```

```
    students.push({ 이름: '나선주', 국어: 98, 수학: 92, 영어: 96, 과학: 92 });
```

```
    students.push({ 이름: '윤아린', 국어: 95, 수학: 98, 영어: 98, 과학: 98 });
```

```
    students.push({ 이름: '윤명월', 국어: 64, 수학: 88, 영어: 92, 과학: 92 });
```

```
    students.push({ 이름: '김미화', 국어: 82, 수학: 86, 영어: 98, 과학: 88 });
```

```
    students.push({ 이름: '김연화', 국어: 88, 수학: 74, 영어: 78, 과학: 92 });
```

```
    students.push({ 이름: '박아현', 국어: 97, 수학: 92, 영어: 88, 과학: 95 });
```

```
    students.push({ 이름: '서준서', 국어: 45, 수학: 52, 영어: 72, 과학: 78 });
```

```
</script>
```

---



## 6.6 객체와 배열을 사용한 데이터 관리

### ◆ 객체와 배열을 사용한 데이터 관리

- 메서드 추가

---

```
<script>
    // 학생 정보 배열을 만듭니다.
    /* 생략 */

    // 모든 students 배열 내의 객체에 메서드를 추가합니다.
    for (var i in students) {
        // 총점을 구하는 메서드를 추가합니다.
        students[i].getSum = function () {
            return this.국어 + this.수학 + this.영어 + this.과학;
        };

        // 평균을 구하는 메서드를 추가합니다.
        students[i].getAverage = function () {
            return this.getSum() / 4;
        };
    }
</script>
```

---

## 6.6 객체와 배열을 사용한 데이터 관리

### ◆ 객체와 배열을 사용한 데이터 관리

- 학생 성적 출력

---

```
<script>
    // 학생 정보 배열을 만듭니다.
    /* 생략 */

    // 모든 students 배열 내의 객체에 메서드를 추가합니다.
    /* 생략 */

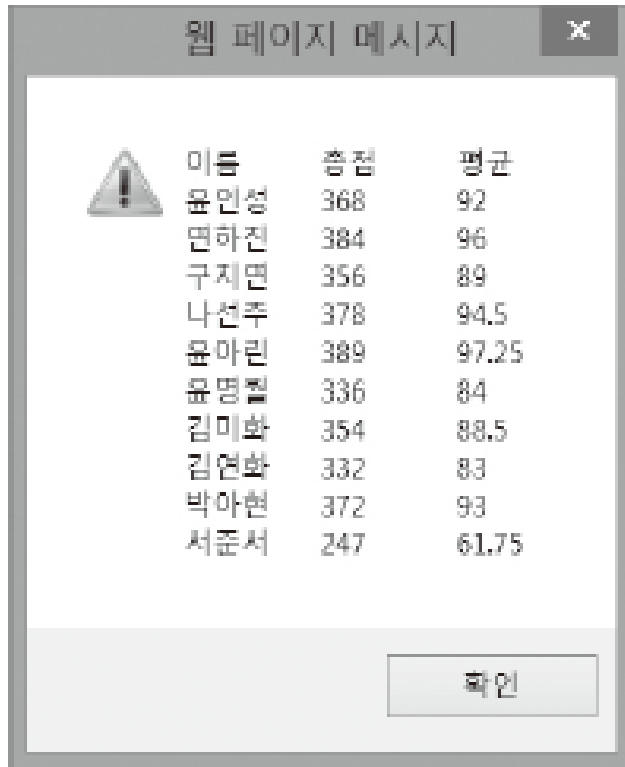
    // 출력합니다.
    var output = '이름\t총점\t평균\n';
    for (var i in students) {
        with (students[i]) {
            output += 이름 + '\t' + getSum() + '\t' + getAverage() + '\n';
        }
    }
    alert(output);
</script>
```

---

## 6.6 객체와 배열을 사용한 데이터 관리

### ◆ 객체와 배열을 사용한 데이터 관리

- 객체를 사용한 학생 성적 관리 결과



A screenshot of a web page message dialog box. The title bar reads '웹 페이지 메시지' (Web Page Message) with a close button. Inside, there is a warning icon (a triangle with an exclamation mark) on the left. To the right of the icon is a table of student scores. At the bottom right of the dialog is a button labeled '확인' (OK).

이름	총점	평균
윤민성	368	92
면하진	384	96
구지면	356	89
나선주	378	94.5
윤아린	389	97.25
윤명철	336	84
김미화	354	88.5
김연화	332	83
박아현	372	93
서준서	247	61.75

# JavaScript 생성자 함수

# 7.1 생성자 함수 개요

---

## ◆ 생성자 함수란?

- new 키워드를 사용해 객체 생성할 수 있는 함수

## ◆ student 생성자

- student 생성자 함수를 만드는 코드

---

```
<script>  
    function Student() {  
  
    }  
</script>
```

---

# 7.1 생성자 함수 개요

## ◆ new 키워드

- new 키워드로 객체 생성

---

```
<script>  
    function Student() {  
  
    }  
  
    var student = new Student();  
</script>
```

---

## 7.1 생성자 함수 개요

### ◆ this 키워드

- 생성자 함수로 생성될 객체의 속성 지정

---

```
<script>
  function Student(name, korean, math, english, science) {
    this.이름 = name;
    this.국어 = korean;
    this.수학 = math;
    this.영어 = english;
    this.과학 = science;
  }
  var student = new Student('윤하린', 96, 98, 92, 98);
</script>
```

---

## 7. 생성자 함수 개요

### ◆ 메서드 생성

---

```
<script>
  function Student(name, korean, math, english, science) {
    // 속성
    this.이름 = name;
    this.국어 = korean;
    this.수학 = math;
    this.영어 = english;
    this.과학 = science;

    // 메서드
    this.getSum = function () {
      return this.국어 + this.수학 + this.영어 + this.과학;
    };
    this.getAverage = function () {
      return this.getSum() / 4;
    };
    this.toString = function () {
      return this.이름 + '\t' + this.getSum() + '\t' + this.getAverage();
    };
  }

  var student = new Student('윤하린', 96, 98, 92, 98);
</script>
```

---



## 7. 생성자 함수 개요

---

### ◆ 생성자 함수를 사용한 객체 배열 생성

---

```
<script>
    function Student(name, korean, math, english, science) {
        /* 생략 */
    }

    // 학생 정보 배열을 만듭니다.
    var students = [];
    students.push(new Student('윤하린', 96, 98, 92, 98));
    /* 생략 */
    students.push(new Student('윤인아', 96, 96, 98, 92));

    // 출력합니다.
    var output = '이름\t총점\t평균\n';
    for (var i in students) {
        output += students[i].toString() + '\n';
    }
    alert(output);
</script>
```

---

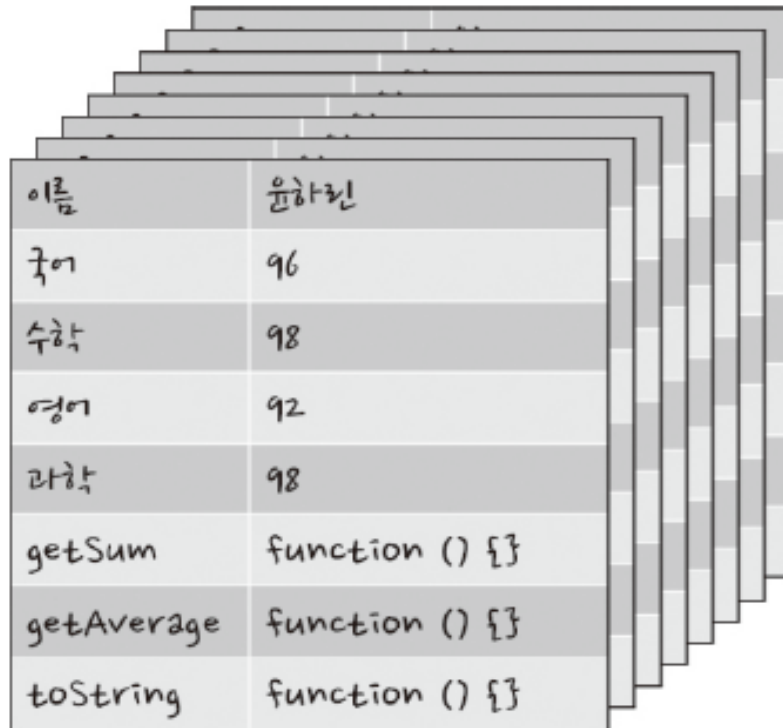
## 7.2 프로토타입

### ◆ 생성자 함수

- 기존의 객체 구조

→ 이름, 국어, 수학, 영어, 과학 속성

→ getSum ( ), getAverage ( ), toString ( ) 메서드



이름	윤하린
국어	96
수학	98
영어	92
과학	98
getSum	function () {}
getAverage	function () {}
toString	function () {}



## 7.2 프로토타입

### ◆ 생성자 함수 구성

- 한 개의 메서드로 모든 객체가 사용
- 생성자 함수로 객체를 만들 때 → 생성자 함수 내부에 속성만 넣음

---

```
<script>
  function Student(name, korean, math, english, science) {
    this.이름 = name;
    this.국어 = korean;
    this.수학 = math;
    this.영어 = english;
    this.과학 = science;
  }
</script>
```

---

## 7.2 프로토타입

### ◆ 프로토타입

- 자바스크립트의 모든 함수는 변수 `prototype`을 갖음
- `prototype`은 객체

```
Student.prototype|
```

⊗ `isPrototypeOf`

● `prototype`

## 7.3 new 키워드

---

### ◆ new 키워드

---

```
<script>
    // 생성자 함수를 선언합니다.
    function Constructor(value) {
        this.value = value;
    }

    // 변수를 선언합니다.
    var constructor = new Constructor('Hello');

    // 출력합니다.
    alert(constructor.value);
</script>
```

---

## 7.3 new 키워드

### ◆ new 키워드를 사용하지 않으면?

- this 키워드 사용 → window 객체를 나타냄
- 일반 함수 호출과 같이 new 키워드를 사용하지 않으면  
→ 함수 실행 중 window 객체에 속성이 추가한 것이 됨
- new 키워드로 함수 호출  
→ 객체를 위한 공간 생성(this 키워드가 해당 공간을 의미)

## 7.4 캡슐화

### ◆ 캡슐화

- 다양한 사람이 많아서 만들어진 기술
- 예제 : Rectangle 객체를 만들어 봄

---

```
<script>
    // 생성자 함수를 선언합니다.
    function Rectangle(width, height) {
        this.width = width;
        this.height = height;
    }
    Rectangle.prototype.getArea = function () {
        return this.width * this.height;
    };

    // 변수를 선언합니다.
    var rectangle = new Rectangle(5, 7);

    // 출력합니다.
    alert('AREA: ' + rectangle.getArea());
</script>
```

---



## 7.4 캡슐화

### ◆ 캡슐화

- 잘못된 속성의 사용

---

```
// 변수를 선언합니다.  
var rectangle = new Rectangle(5, 7);  
rectangle.width = -2;  
  
// 출력합니다.  
alert('AREA: ' + rectangle.getArea());
```

---

## 7.4 캡슐화

### ◆ 캡슐화

- 캡슐화는 잘못 사용될 수 있는 객체의 특정 부분을 사용자가 사용할 수 없게 막는 기술

---

```
// 생성자 함수를 선언합니다.
function Rectangle(w, h) {
    // 변수를 선언합니다.
    var width = w;
    var height = h;

    // 메서드를 선언합니다.
    this.getWidth = function () { return width; };
    this.getHeight = function () { return height; };
    this.setWidth = function (w) {
        width = w;
    };
    this.setHeight = function (h) {
        height = h;
    };
}
```

---

## 7.4 캡슐화

---

### ◆ 캡슐화

- 게터 : `get○○ ()` 형태의 메서드와 같이 값을 가져오는 메서드
- 세터 : `set○○ ()` 형태의 메서드와 같이 값을 입력하는 메서드
- 게터와 세터를 만드는것이 캡슐화는 아님
- 캡슐화는 만일의 상황에 특정 속성이나 메서드를 사용자가 사용할 수 없도록 숨겨 놓는 것임

# JavaScript 기본 내장 객체

## 8. 기본 내장 객체

### ◆ 기본 내장 객체

- 자바스크립트에는 이미 많은 객체가 내장
- w3schools(<http://www.w3schools.com/jsref/default.asp>)
  - 기본 내장 객체의 속성과 메서드에 관한 설명 및 예제 제공

그림 8-1 w3schools(<http://www.w3schools.com/jsref/default.asp>)

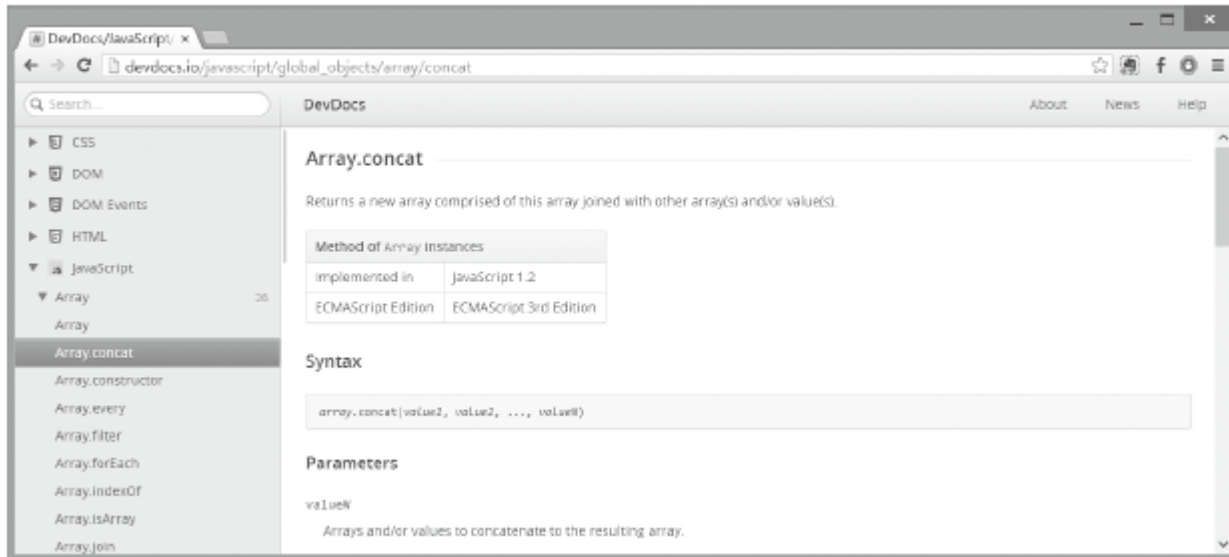


# 8. 기본 내장 객체

## ◆ 기본 내장 객체

- devdocs.io(<http://devdocs.io/javascript/>)
  - 각각의 속성, 메서드와 관련된 자세한 정보

그림 8-2 devdocs.io(<http://devdocs.io/javascript/>)



## 8.1 기본 자료형과 객체의 차이점

### ◆ 기본 자료형

- 숫자, 문자열, 불 세 가지 자료형을 의미
- 기본 자료형과 객체의 자료형과 값을 출력
- 변수 primitiveNumber는 기본 자료형 숫자
- 변수 objectNumber는 생성자 함수를 사용하므로 객체

---

```
<script>
  // 변수를 선언합니다.
  var primitiveNumber = 273;
  var objectNumber = new Number(273);

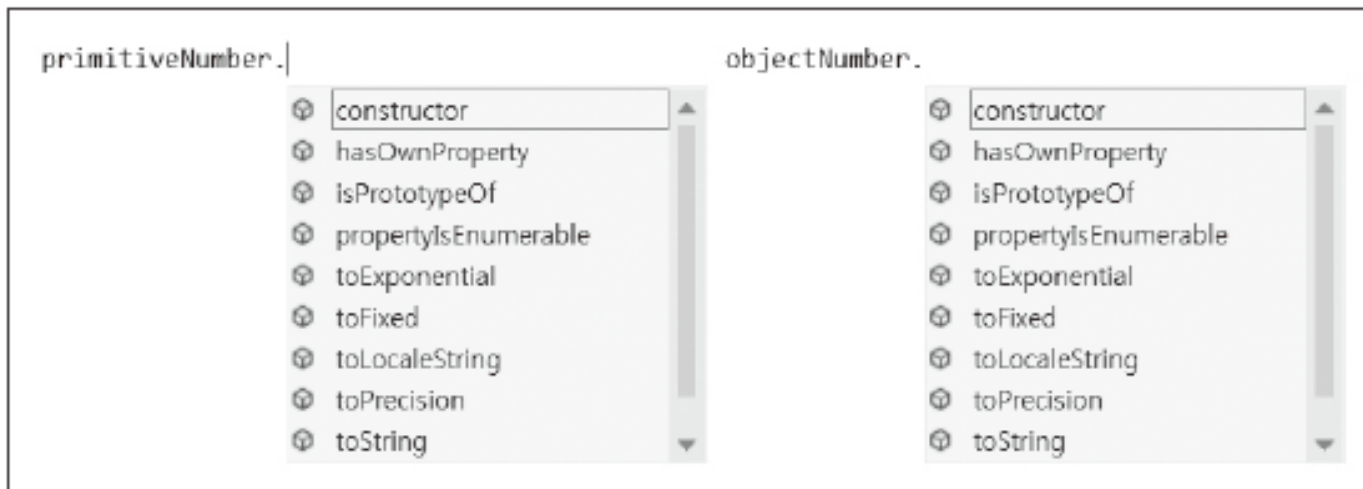
  // 출력합니다.
  var output = '';
  output += typeof (primitiveNumber) + ' : ' + primitiveNumber + '\n';
  output += typeof (objectNumber) + ' : ' + objectNumber;
  alert(output);
</script>
```

---

## 8.1 기본 자료형과 객체의 차이점

### ◆ 기본 자료형과 객체의 메서드

- 기본 자료형의 속성 및 메서드 사용  
→ 기본 자료형이 자동으로 객체 변환





## 8.1 기본 자료형과 객체의 차이점

### ◆ 기본 자료형과 객체의 메서드

- 기본 자료형에 메서드 추가

---

```
<script>
  // 변수를 만듭니다.
  var primitiveNumber = 273;

  // 메서드를 추가합니다.
  primitiveNumber.method = function () {
    return 'Primitive Method'
  };

  // 메서드를 실행합니다.
  var output = primitiveNumber.method() + '\n';
  alert(output);
</script>
```

---

## 8.1 기본 자료형과 객체의 차이점

### ◆ 기본 자료형과 객체의 메서드

- 생성자 함수에 메서드 추가

---

```
<script>
    // 변수를 만듭니다.
    var primitiveNumber = 273;
    var objectNumber = new Number(273);

    // 메서드를 추가합니다.
    Number.prototype.method = function () {
        return 'Method on Prototype';
    };

    // 메서드를 실행합니다.
    var output = '';
    output += primitiveNumber.method() + '\n';
    output += objectNumber.method();

    // 출력합니다.
    alert(output);
</script>
```

---

## 8.2 Object 객체

### ◆ 생성

- Object 객체는 자바스크립트의 최상위 객체
- 두 가지 방법으로 Object 객체 생성

---

```
var object = {};  
var object = new Object();
```

---

- 일곱 가지 메서드

object.|

- ⊗ constructor
- ⊗ hasOwnProperty
- ⊗ isPrototypeOf
- ⊗ propertyIsEnumerable
- ⊗ toLocaleString
- ⊗ toString
- ⊗ valueOf

## 8.2 Object 객체

### ◆ 속성과 메서드

- toString( ) 메서드 재선언

---

```
<script>
    // 변수를 선언합니다.
    var student = {
        name: '윤명월',
        grade: '대학교 1학년',
        toString: function () {
            return this.name + ' : ' + this.grade;
        }
    };

    // 출력합니다.
    alert(student);
</script>
```

---

## 8.2 Object 객체

---

### ◆ 자료형 구분

- Object 객체의 constructor ( ) 메서드는 객체의 생성자 함수를 의미
- constructor ( ) 메서드는 자료형을 검사할 때 유용하게 사용
- 자료형을 검사할 때는 typeof 연산자 사용

## 8.2 Object 객체

### ◆ 자료형 구분

- 자료형을 검사할 때는 typeof 연산자 사용
- 두 변수 모두 숫자지만 생성자 함수로 만든 숫자는 객체

---

```
<script>
    // 변수를 만듭니다.
    var numberFromLiteral = 273;
    var numberFromObject = new Number(273);

    // 출력합니다.
    var output = '';
    output += '1. ' + typeof (numberFromLiteral) + '\n';
    output += '2. ' + typeof (numberFromObject);
    alert(output);
</script>
```

---

## 8.2 Object 객체

### ◆ 자료형 구분

- typeof 연산자를 사용한 자료형 비교

---

```
<script>
    // 변수를 선언합니다.
    var numberFromLiteral = 273;
    var numberFromObject = new Number(273);
    // 자료형을 확인합니다.
    if (typeof (numberFromLiteral) == 'number') {
        alert('numberFromLiteral은 숫자입니다.');
```

```
    }
    if (typeof (numberFromObject) == 'number') {
        alert('numberFromObject는 숫자입니다.');
```

```
    }
</script>
```

---

## 8.3 Number 객체

### ◆ Number 객체

- 가장 단순한 객체로 숫자를 표현할 때 사용
- Number 객체 생성

---

```
<script>
    // 변수를 선언합니다.
    var numberFromLiteral = 273;
    var numberFromConstructor = new Number(273);

    // 출력합니다.
    alert(typeof (numberFromLiteral));
    alert(typeof (numberFromConstructor));
</script>
```

---



## 8.3 Number 객체

### ◆ 메서드

- Object 객체가 갖는 일곱 가지 메서드 + 세 가지 메서드 추가

메서드 이름	설명
toExponential()	숫자를 지수 표시로 나타낸 문자열을 리턴합니다.
toFixed()	숫자를 고정 소수점 표시로 나타낸 문자열을 리턴합니다.
toPrecision()	숫자를 길이에 따라 지수 표시 또는 고정 소수점 표시로 나타낸 문자열을 리턴합니다.

```
<script>
    // 변수를 선언합니다.
    var number = 273.5210332;

    // 출력합니다.
    var output = '';
    output += number.toFixed(1) + '\n';
    output += number.toFixed(4);
    alert(output);
</script>
```

## 8.4 String 객체

### ◆ String 객체

- 자바스크립트에서 가장 많이 사용하는 내장 객체

### ◆ 생성

- 두 가지 방법으로 생성

---

```
<script>
  // 변수를 선언합니다.
  var stringFromLiteral = 'Hello World..!';
  var stringFromConstructor = new String('Hello World..!');

  // 변수의 자료형을 출력합니다.
  var output = '';
  output += typeof (stringFromLiteral) + '\n';
  output += typeof (stringFromConstructor);
  alert(output);
</script>
```

---

## 8.4 String 객체

### ◆ 기본 속성과 메서드

- String 객체는 length 속성을 가짐

속성 이름	설명
length	문자열의 길이를 나타냅니다.

```
<script>
  // 변수를 선언합니다.
  var characters = prompt('사용할 비밀번호를 입력해주세요.', '6글자 이상');

  // 출력합니다.
  if (characters.length < 6) {
    alert('6글자 이상으로 입력하세요.');
```

```
  } else {
    alert('잘했어요!');
  }
</script>
```

## 8.4 String 객체

### ◆ 기본 속성과 메서드

- String 객체의 메서드는 기본 메서드와 HTML 관련 메서드로 구분
- 기본 메서드

메서드 이름	설명
charAt(position)	position에 위치하는 문자를 리턴합니다.
charCodeAt(position)	position에 위치하는 문자의 유니코드 번호를 리턴합니다.
concat(args)	매개변수로 입력한 문자열을 이어서 리턴합니다.
indexOf(searchString, position)	앞에서부터 일치하는 문자열의 위치를 리턴합니다.
lastIndexOf(searchString, position)	뒤에서부터 일치하는 문자열의 위치를 리턴합니다.
match(regExp)	문자열 내에 regExp가 있는지 확인합니다.
replace(regExp, replacement)	regExp를 replacement로 바꾼 뒤 리턴합니다.
search(regExp)	regExp와 일치하는 문자열의 위치를 리턴합니다.
slice(start, end)	특정 위치의 문자열을 추출해 리턴합니다.
split(separator, limit)	separator로 문자열을 잘라서 배열을 리턴합니다.
substr(start, count)	start부터 count만큼 문자열을 잘라서 리턴합니다.
substring(start, end)	start부터 end까지 문자열을 잘라서 리턴합니다.
toLowerCase()	문자열을 소문자로 바꾸어 리턴합니다.
toUpperCase()	문자열을 대문자로 바꾸어 리턴합니다.

## 8.4 String 객체

### ◆ 기본 속성과 메서드

- 올바른 String 객체의 메서드 사용

---

```
<script>
    // 변수를 선언합니다.
    var string = 'abcdefg';

    // 출력합니다.
    string = string.toUpperCase();
    alert(string);
</script>
```

---

## 8.5 Array 객체

### ◆ Array 객체

- 여러가지 자료를 쉽게 관리할 수 있게 도와주는 객체

### ◆ 생성

- Array 생성자 함수

생성자 함수	설명
Array()	빈 배열을 만듭니다.
Array(number)	매개변수만큼의 크기를 가지는 배열을 만듭니다.
Array(mixed, ..., mixed)	매개변수를 배열로 만듭니다.

```
<script>
  // 변수를 선언합니다.
  var array1 = [52, 273, 103, 57, 32];
  var array2 = new Array();
  var array3 = new Array(10);
  var array4 = new Array(52, 273, 103, 57, 32);
</script>
```

## 8.5 Array 객체

### ◆ 속성과 메서드

- 배열은 몇 개의 요소가 있는지 나타내는 length 속성이 있음

속성 이름	설명
length	배열 요소의 개수를 알아냅니다.

```
<script>
  // 변수를 선언합니다.
  var array = ['A', 'B', 'C', 'D'];

  // 출력합니다.
  var output = '';
  for (var i = 0; i < array.length; i++) {
    output += i + ' : ' + array[i] + '\n';
  }
  alert(output);
</script>
```

## 8.5 Array 객체

### ◆ 속성과 메서드

#### ■ Array 객체의 메서드

메서드 이름	설명
concat( )	매개변수로 입력한 배열의 요소를 모두 합쳐 배열을 만들어 리턴합니다.
join( )	배열 안의 모든 요소를 문자열로 만들어 리턴합니다.
pop( )*	배열의 마지막 요소를 제거하고 리턴합니다.
push( )*	배열의 마지막 부분에 새로운 요소를 추가합니다.
reverse( )*	배열의 요소 순서를 뒤집습니다.
slice( )	배열 요소의 지정한 부분을 리턴합니다.
sort( )*	배열의 요소를 정렬합니다.
splice( )*	배열 요소의 지정한 부분을 삭제하고 삭제한 요소를 리턴합니다.



## 8.5 Array 객체

### ◆ 속성과 메서드

- sort( ) 메서드 사용

---

```
<script>
    // 변수를 선언하고 정렬합니다.
    var array = [52, 273, 103, 32];
    array.sort();

    // 출력합니다.
    alert(array);
</script>
```

---

## 8.5 Array 객체

### ◆ 정렬

- `sort()` 메서드의 정렬에 변화를 주고 싶을 때  
→ `sort()` 메서드의 매개변수로 함수를 넣어 줌
- `sort()` 메서드의 매개변수로 들어가는 함수는 기본적으로 매개 변수 두 개를 받을 수 있어야 함

---

```
array.sort(function (left, right) {  
  
});
```

---

오름차순 정렬	내림차순 정렬
<pre>function (left, right) {     return left - right; }</pre>	<pre>function (left, right) {     return right - left; }</pre>

## 8.5 Array 객체

### ◆ 정렬

- sort( ) 메서드의 정렬 방식 지정

---

```
<script>
    // 변수를 선언하고 정렬합니다.
    var array = [52, 273, 103, 32];
    array.sort(function (left, right) {
        return left - right;
    });

    // 출력합니다.
    alert(array);
</script>
```

---

## 8.5 Array 객체

### ◆ 정렬

#### ■ 예제 : 학생 성적 정렬

---

```
<script>
// 생성자 함수를 선언합니다.
function Student(name, korean, math, english, science) {
    // 속성
    this.이름 = name;
    this.국어 = korean;
    this.수학 = math;
    this.영어 = english;
    this.과학 = science;

    // 메서드
    this.getSum = function () {
        return this.국어 + this.수학 + this.영어 + this.과학;
    };
    this.getAverage = function () {
        return this.getSum() / 4;
    };
    this.toString = function () {
        return this.이름 + '\t' + this.getSum() + '\t' + this.getAverage();
    };
}

// 학생 정보 배열을 만듭니다.
var students = [];
students.push(new Student('윤하린', 96, 98, 92, 98));
/* 생략 */
```

## 8.5 Array 객체

---

```
students.push(new Student('윤명월', 92, 94, 88, 98));

// 정렬하고 1등부터 3등까지만 배열에 남겨둡니다.
students.sort(function (left, right) {
    return right.getSum() - left.getSum();
});
students = students.slice(0, 3);

// 출력합니다.
var output = '이름\t총점\t평균\n';
for (var i in students) {
    output += students[i].toString() + '\n';
}
alert(output);
</script>
```

---

## 8.5 Array 객체

### ◆ 정렬

- slice( ) 메서드
- sort( ) 메서드의 매개변수로 총점을 내림차순 정렬
- 0번째 인덱스부터 변수 students에 할당

---

```
// 정렬하고 1등부터 3등까지만 배열에 남겨둡니다.  
students.sort(function (left, right) {  
    return right.getSum() - left.getSum();  
});  
students = students.slice(0, 3);
```

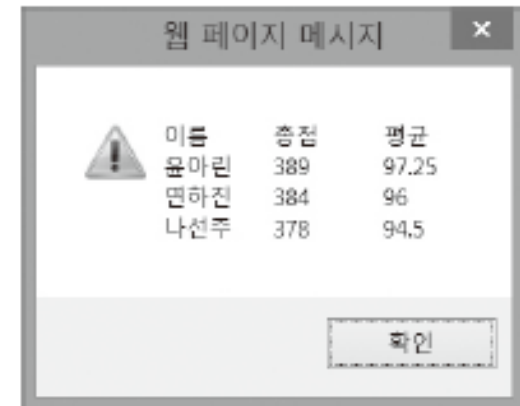
---

## 8.5 Array 객체

### ◆ 정렬

- Array 객체 메서드의 체이닝

```
// 정렬하고 1등부터 3등까지만 배열에 남겨둡니다.  
students = students.sort(function (left, right) {  
    return right.getSum() - left.getSum();  
}).slice(0, 3);
```



## 8.6 Date 객체

### ◆ Date 객체

- 날짜와 시간을 표시하는 객체

### ◆ 생성

- 날짜와 시간을 표시하는 객체
- 생성자 함수에 어떤 매개변수도 입력하지 않으면 현재 시간으로 초기화

---

```
<script>
    // 변수를 선언합니다.
    var date = new Date();

    // 출력합니다.
    alert(date);
</script>
```

---



## 8.6 Date 객체

### ◆ 생성

- 문자열을 사용한 Date 객체 생성

---

```
var date = new Date('December 9');  
var date = new Date('December 9, 1991');  
var date = new Date('December 9, 1991 02:24:23');
```

---

- 숫자를 사용한 Date 객체 생성

---

```
var date = new Date(1991, 11, 9);  
var date = new Date(1991, 11, 9, 2, 24, 23);  
var date = new Date(1991, 11, 9, 2, 24, 23, 1);
```

---

## 8.6 Date 객체

### ◆ 메서드

- get 형태의 메서드 : 게터
- set 형태의 메서드 : 세터

```
var date = new Date();  
date.get
```

- ⊗ getDate
- ⊗ getDay
- ⊗ getFullYear
- ⊗ getHours
- ⊗ getMilliseconds
- ⊗ getMinutes
- ⊗ getMonth
- ⊗ getSeconds
- ⊗ getTime

```
var date = new Date();  
date.set
```

- ⊗ setDate
- ⊗ setFullYear
- ⊗ setHours
- ⊗ setMilliseconds
- ⊗ setMinutes
- ⊗ setMonth
- ⊗ setSeconds
- ⊗ setTime
- ⊗ setUTCDate

## 8.6 Date 객체

### ◆ 메서드

- to○○String ( ) 형태의 메서드



## 8.6 Date 객체

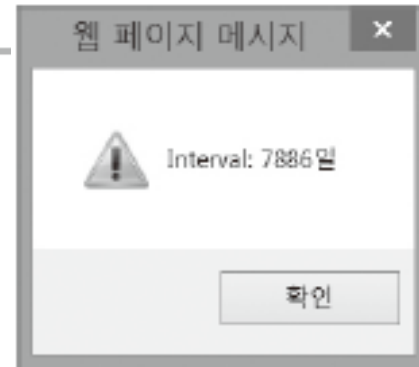
### ◆ 시간 간격 계산

- 날짜 간격 : getTime( ) 함수 사용

```
<script>
    // 변수를 선언합니다.
    var now = new Date();
    var before = new Date('December 9, 1991');

    // 날짜 간격을 구합니다.
    var interval = now.getTime() - before.getTime();
    interval = Math.floor(interval / (1000 * 60 * 60 * 24));

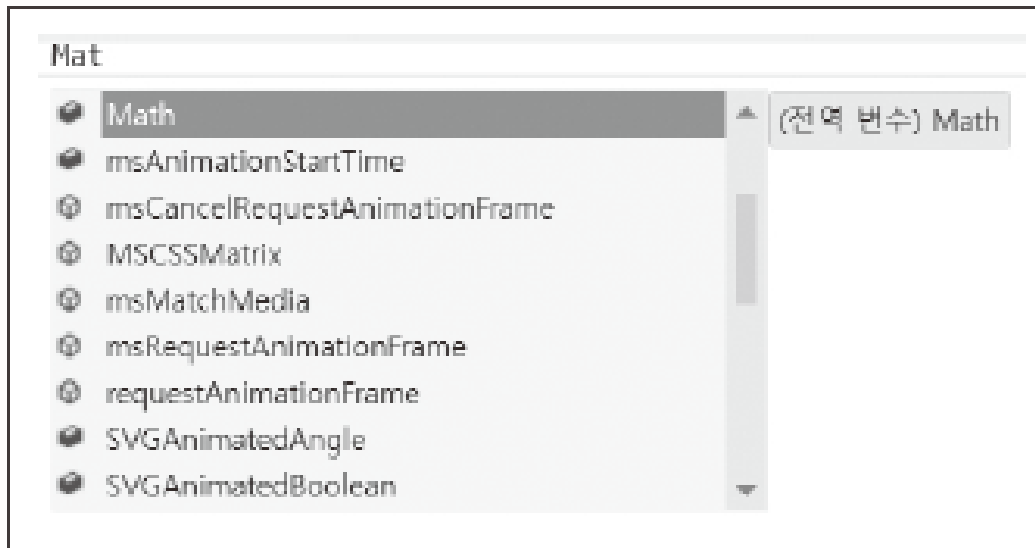
    // 출력합니다.
    alert('Interval: ' + interval + '일');
</script>
```



## 8.7 Math 객체

### ◆ Math 객체

- Math 객체는 변수



## 8.7 Math 객체

### ◆ Math 객체

- Math 객체의 속성

속성 이름	값
E	2.718281828459045
LN2	0.6931471805599453
LN10	2.302585092994046
LOG2E	1.4426950408889633
LOG10E	0.4342944819032518
PI	3.141592653589793
SQRT1_2	0.7071067811865476
SQRT2	1.4142135623730951

## 8.7 Math 객체

### ◆ Math 객체

#### ■ Math 객체의 메서드

메서드 이름	설명
abs(x)	x의 절대 값을 리턴합니다.
acos(x)	x의 아크 코사인 값을 리턴합니다.
asin(x)	x의 아크 사인 값을 리턴합니다.
atan(x)	x의 아크 탄젠트 값을 리턴합니다.
atan2(y, x)	x와 y의 비율로 아크 탄젠트 값을 구해 리턴합니다.
ceil(x)	x보다 크거나 같은 가장 작은 정수를 리턴합니다.
cos(x)	x의 코사인 값을 리턴합니다.
exp(x)	자연 로그의 x 제곱을 리턴합니다.
floor(x)	x보다 작거나 같은 가장 큰 정수를 리턴합니다.
log(x)	x의 로그 값을 리턴합니다.
max(x,y,z,...,n)	매개변수 중 가장 큰 값을 리턴합니다.
min(x,y,z,...,n)	매개변수 중 가장 작은 값을 리턴합니다.
pow(x,y)	x의 y 제곱을 리턴합니다.
random( )	0부터 1까지의 임의의 수를 리턴합니다.
round(x)	x를 반올림하여 리턴합니다.
sin(x)	x의 사인 값을 리턴합니다.
sqrt(x)	x의 제곱근을 리턴합니다.
tan(x)	x의 탄젠트 값을 리턴합니다.

## 8.8 JSON

---

### ◆ JSON

- JSON은 JavaScript Object Notation의 약자.
- JSON은 좀 더 쉽게 데이터를 교환하고 저장하기 위하여 만들어진 텍스트 기반의 데이터 교환 표준이다.
- JSON은 XML의 대안으로서 좀 더 쉽게 데이터를 교환하고 저장하기 위하여 고안되었다.
- JSON은 텍스트 기반이므로 어떠한 프로그래밍 언어에서도 JSON 데이터를 읽고 사용할 수 있다.
  - JSON은 자바스크립트를 확장하여 만들어졌다.
  - JSON은 자바스크립트 객체 표기법을 따른다.
  - JSON은 사람과 기계가 모두 읽기 편하도록 고안되었다.
  - JSON은 프로그래밍 언어와 운영체제에 독립적이다.



## 8.8 JSON

---

### ◆ XML

- XML은 EXtensible Markup Language의 약자.
- HTML과 비슷한 문자 기반의 마크업 언어(text-based markup language)
- 이 언어는 사람과 기계가 동시에 읽기 편한 구조로 되어 있다.
- XML은 데이터를 저장하고 전달할 목적으로만 만들어졌다.
- XML 태그는 HTML 태그처럼 미리 정의되어 있지 않고, 사용자가 직접 정의할 수 있다.

## 8.8 JSON

---

### ◆ JSON과 XML의 공통점

- 데이터를 저장하고 전달하기 위해 고안되었다.
- 기계뿐만 아니라 사람도 쉽게 읽을 수 있다.
- 계층적인 데이터 구조를 가진다.
- 다양한 프로그래밍 언어에 의해 파싱 될 수 있다.

### ◆ JSON과 XML의 차이점

- JSON은 종료 태그를 사용하지 않는다.
- JSON의 구문이 XML의 구문보다 더 짧다.
- JSON 데이터가 XML 데이터보다 더 빨리 읽고 쓸 수 있다.
- XML은 배열을 사용할 수 없지만, JSON은 배열을 사용할 수 있다.

## 8.8 JSON

---

### ◆ JSON 구조

- JSON은 자바스크립트의 객체 표기법으로부터 파생된 부분 집합입니다.
- 따라서 JSON 데이터는 다음과 같은 자바스크립트 객체 표기법에 따른 구조로 구성됩니다.
  - JSON 데이터는 이름과 값의 쌍으로 이루어집니다.
  - JSON 데이터는 쉼표(,)로 나열됩니다.
  - 객체(object)는 중괄호({})로 둘러싸아 표현합니다.
  - 배열(array)은 대괄호([])로 둘러싸아 표현합니다.

## 8.8 JSON

### ◆ JSON 데이터

- JSON 데이터는 이름과 값의 쌍으로 구성된다.
- 이러한 JSON 데이터는 데이터 이름, 콜론(:), 값의 순서로 구성된다.
- 문법 → **"데이터이름": 값**
- 데이터의 이름도 문자열이므로, 항상 큰따옴표("")와 함께 입력해야 한다.
- 데이터의 값으로는 다음과 같은 타입이 올 수 있다.
  - 숫자(number)
  - 문자열(string)
  - 불리언(boolean)
  - 객체(object)
  - 배열(array)
  - NULL

## 8.8 JSON

---

### JSON 객체

```
{  
  "name": "식빵",  
  "family": "웰시코기",  
  "age": 1,  
  "weight": 2.14  
}
```

### JSON 배열

```
"dog": [  
  {"name": "식빵", "family": "웰시코기", "age": 1, "weight": 2.14},  
  {"name": "콩콩", "family": "포메라니안", "age": 3, "weight": 2.5},  
  {"name": "젤리", "family": "푸들", "age": 7, "weight": 3.1}  
]
```

## 8.8 JSON

---

- ◆ JSON은 자바스크립트의 객체 표기법을 제한하여 만든 텍스트 기반의 데이터 교환 표준이다.
- ◆ JSON 데이터는 자바스크립트가 자주 사용되는 웹 환경에서 사용하는 것이 유리하다.
- ◆ 자바스크립트는 JSON 데이터를 처리하기 위한 다음과 같은 메소드를 제공하고 있다.
  - `JSON.stringify()`
  - `JSON.parse()`

## 8.8 JSON

```
<!DOCTYPE html>
<html>
<head>
  <title>자바스크립트 기본 문법</title>
  <script>
```

```
    window.onload = function() {

      var book = {
        name: "JAVA",
        price: 1000,
        publisher: "Orange Media"
      }; // 자바스크립트 객체
```

```
      var data = JSON.stringify(book); // 자바스크립트 객체를 문자열로 변환함.
      document.getElementById("json").innerHTML = data;
    };
  </script>
```

```
  var newdata = '{"name":"JAVA","price":1000,"publisher":"Orange Media"}';
```

```
  var newBook = JSON.parse(newdata); // JSON 형식의 문자열을 자바스크립트 객체로 변환함.
```

```
  alert(newBook.name + ' : ' + newBook.price + ' : ' + newBook.publisher);
```

```
  </script>
</head>
<body>
  <h1 id="json">안녕하세요</h1>
</body>
</html>
```

