

# JavaScript

출처 : Javascript jQuery 입문(윤인성, 한빛미디어)

# 1. 자바스크립트로 할 수 있는 일

---

## ❖ 웹 클라이언트 애플리케이션 개발

- 초기에 웹 문서를 동적으로 제어하기 위해 사용되었으나, 현재는 웹 브라우저에서 실행되는 웹 클라이언트 애플리케이션 개발이 목적
- 웹 브라우저에서 실행할 수 있는 유일한 프로그래밍 언어

## ❖ 웹 서버 개발

- 기존에 웹 개발은 두 가지 이상의 프로그래밍 언어가 필요했음
  - 웹 클라이언트를 자바스크립트로 개발하고, 웹 서버를 다른 언어(Java, C#, Ruby, Python 등)로 개발
- Node.js가 등장하면서 웹 서버도 자바스크립트로 개발 가능

# 1. 자바스크립트로 할 수 있는 일

## ❖ 모바일 애플리케이션 개발

### ■ 네이티브 애플리케이션

- 스마트폰에서 인식할 수 있는 프로그래밍 언어(자바, 스위프트 등)로 만든 애플리케이션
- 기업에서 애플리케이션을 만들 경우 2가지 언어로 만들기에 비용이 2배가 됨
- 자바스크립트를 사용하면 1개의 애플리케이션만 개발해도 스마트폰 동작 가능

### ■ 페이스북의 React Native

- 자바스크립트만으로 모든 운영체제에서 빠르게 동작하는 네이티브 애플리케이션을 개발(안드로이드와 아이폰에 있는 페이스북, 페이스북 그룹, 사운드 클라우드 애플리케이션이 모두 자바스크립트로 만든 네이티브 애플리케이션임)



그림 1-6 React Native

# 1. 자바스크립트로 할 수 있는 일

## ❖ 데스크톱 애플리케이션 개발

- 일렉트론Electron 모듈 : 자바스크립트로 개발 전용 텍스트 에디터를 만들어 배포, 본격적으로 데스크톱 애플리케이션 개발

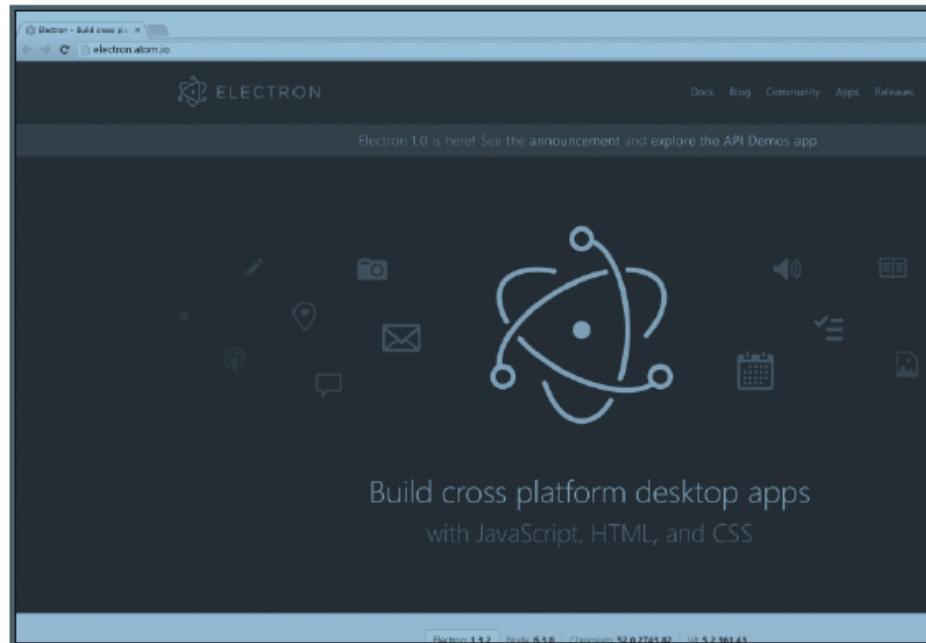


그림 1-7 일렉트론(<http://electron.atom.io/>)

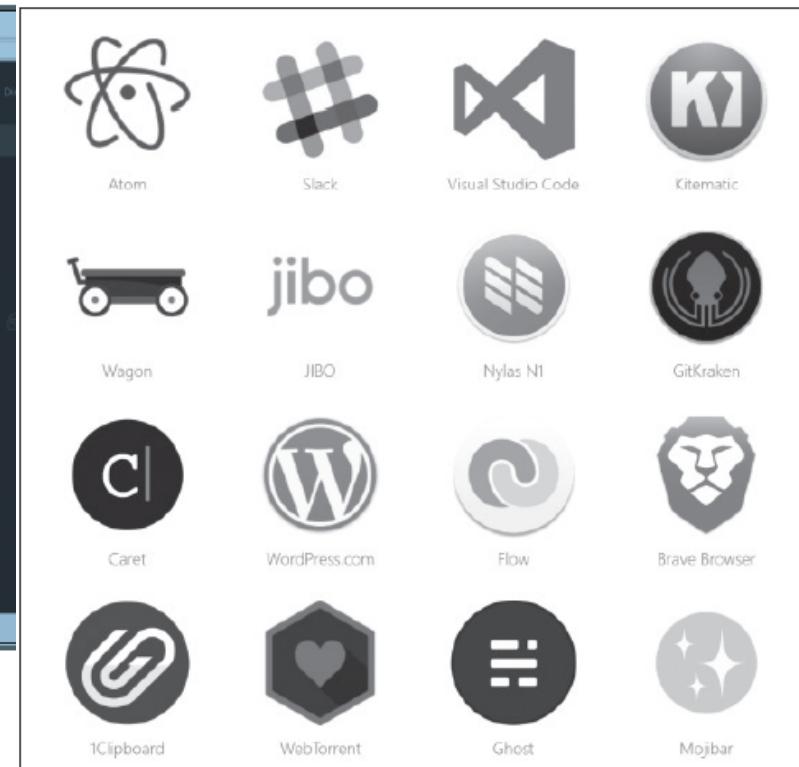


그림 1-8 자바스크립트로 만든 대표적인 데스크톱 애플리케이션

# 1. 자바스크립트로 할 수 있는 일

## ❖ 게임 개발

- 원래 게임은 서버와 클라이언트 모두 C++로 제작(속도가 빠름)
- 스마트폰이 활성화 되면서 '한 번에 여러 스마트폰 운영체제에서 실행할 수 있는 애플리케이션을 개발하는 것'이 경제적으로 이득이 됨
- 유니티 Unity 게임 엔진 등장 : 자바스크립트 기반

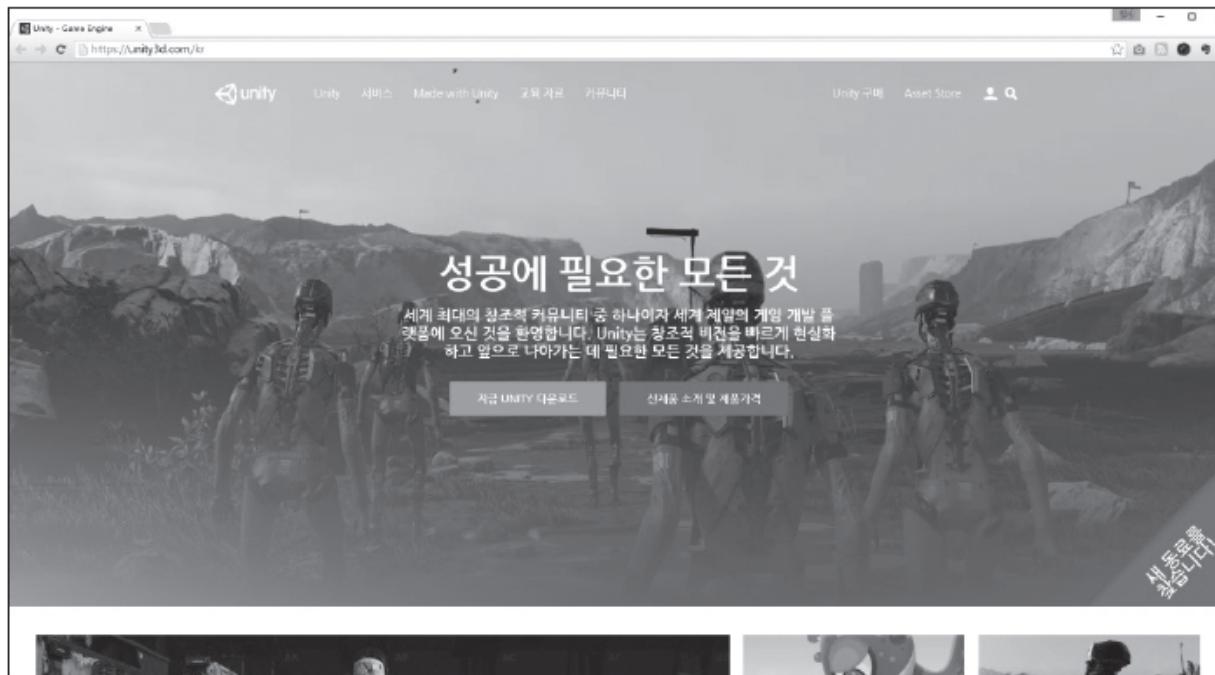


그림 1-9 유니티 게임 엔진

# 1. 자바스크립트로 할 수 있는 일

## ❖ 데이터베이스 관리

### ■ 데이터베이스

- NoSQL : 기존의 SQL은 복잡하고 무거워 사용하기 쉬운 데이터베이스 등장
- 2016년 9월 기준으로 데이터베이스 엔진 순위
  - Oracle, MySQL, Microsoft SQL Server, MongoDB, PostgreSQL
  - MongoDB : 데이터베이스를 관리할 때 자바스크립트를 활용하는 NoSQL 데이터 베이스

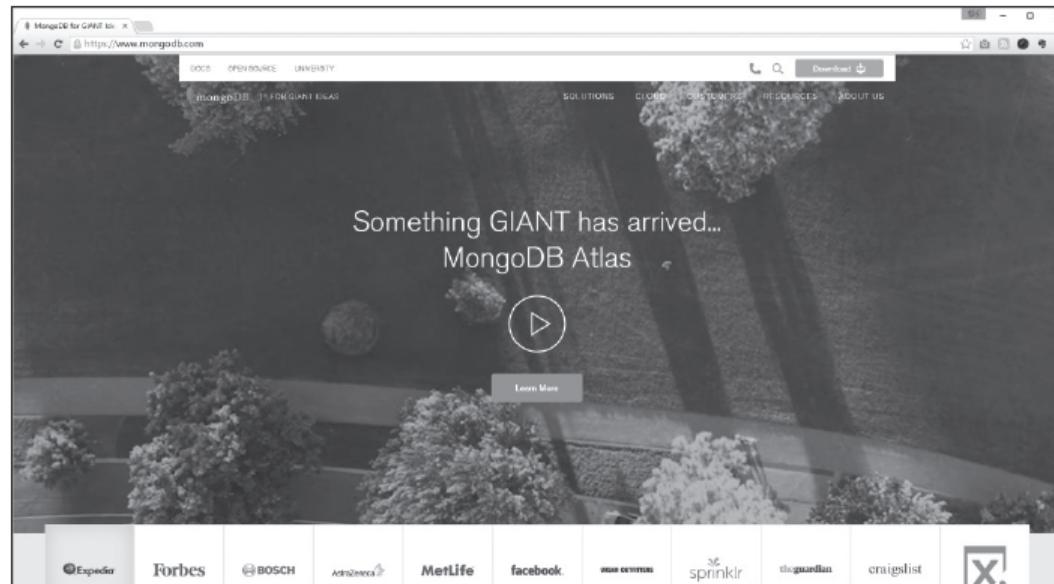


그림 1-11 MongoDB(<https://www.mongodb.com/>)

## 2. 자바스크립트 종류

### ❖ 자바스크립트 버전

- 자바스크립트가 많은 곳에서 사용되자 유럽컴퓨터제조협회(ECMA)에서 ECMAScript라는 이름으로 표준화

| 버전             | 표준발표시기   |
|----------------|----------|
| ECMAScript 1   | 1997. 06 |
| ECMAScript 2   | 1998. 06 |
| ECMAScript 3   | 1999. 12 |
| ECMAScript 4   | 2008. 10 |
| ECMAScript 5   | 2009. 12 |
| ECMAScript 5.1 | 2011. 06 |
| ECMAScript 6   | 2015. 06 |
| ECMAScript 7   | 2016. 06 |

# 문법

# 기본 용어

## ❖ 표현식과 문장

- 표현식 : 값을 나타내는 간단한 코드

```
273
```

```
10 + 20 + 30 * 2  
"JavaScript Programming"
```

- 문장 : 표현식이 하나 이상 모인 경우, 마지막에 종결 의미로 세미콜론(;)
- 프로그램 : 문장이 모이면 프로그램이 됨

```
273;  
10 + 20 + 30 + 2;  
let name = "윤" + "인" + "성"  
console.log("Hello World...!")
```

# 기본 용어

## ❖ 키워드

- 자바스크립트에서 특별한 의미가 부여된 단어

표 2-1 키워드

|          |          |            |        |
|----------|----------|------------|--------|
| break    | else     | instanceof | true   |
| case     | false    | new        | try    |
| catch    | finally  | null       | typeof |
| continue | for      | return     | var    |
| default  | function | switch     | void   |
| delete   | if       | this       | while  |
| do       | in       | throw      | with   |
| let      | const    |            |        |

# 기본 용어

## ❖ 식별자

- 이름을 붙일 때 사용하는 단어, 변수와 함수 이름 등으로 사용

- 키워드를 사용 안됨
- 특수 문자는 \_와 \$만 허용
- 숫자로 시작하면 안됨
- 공백은 입력하면 안됨

|         |           |
|---------|-----------|
| alpha   | break     |
| alpha10 | 273alpha  |
| _alpha  | has space |
| \$alpha |           |
| AlPha   | X         |
| ALPHA   | O         |

- 식별자 사용 규칙(Camel Notation)

- 생성자 함수의 이름은 항상 대문자로 시작
- 변수, 함수, 속성, 메소드의 이름은 항상 소문자로 시작
- 여러 단어로 된 식별자는 각 단어의 첫 글자를 대문자로 함

```
function Product(name, price) {  
    this.name = name;  
    this.price = price;  
}
```

```
let result = Number('100');  
let productList = '';
```

```
function sum(min, max) {  
    let result = 0;  
    for (let i = min; i <= max; i++) {  
        result += i;  
    }  
    return result;  
}
```

# 기본 용어

표 2-2 자바스크립트 식별자의 종류

| 구분           | 단독으로 사용  | 다른 식별자와 사용 |
|--------------|----------|------------|
| 식별자 뒤에 괄호 없음 | 변수 또는 상수 | 속성         |
| 식별자 뒤에 괄호 있음 | 함수       | <u>메소드</u> |

|  |            |
|--|------------|
| <code>alert('Hello World')</code>        | ⇒ 함수       |
| <code>Array.length</code>                | ⇒ 속성       |
| <code>input</code>                       | ⇒ 변수 또는 상수 |
| <code>prompt('Message', 'Defstr')</code> | ⇒ 함수       |
| <code>Math.PI</code>                     | ⇒ 속성       |
| <code>Math.abs(-273)</code>              | ⇒ 메소드      |

# 기본 용어

## ❖ 주석

- 프로그램의 진행에 영향을 주지 않는 코드

표 2-3 주석 처리 방법

| 방법         | 표현                   |
|------------|----------------------|
| 한 줄 주석 처리  | // 주석                |
| 여러 줄 주석 처리 | /*<br>주석<br>주석<br>*/ |

코드 2-1 주석

```
// 주석은 코드의 실행에 영향을 주지 않습니다.  
/*  
console.log("JavaScript Programming")  
console.log("JavaScript Programming")  
console.log("JavaScript Programming")  
*/
```

# 출력

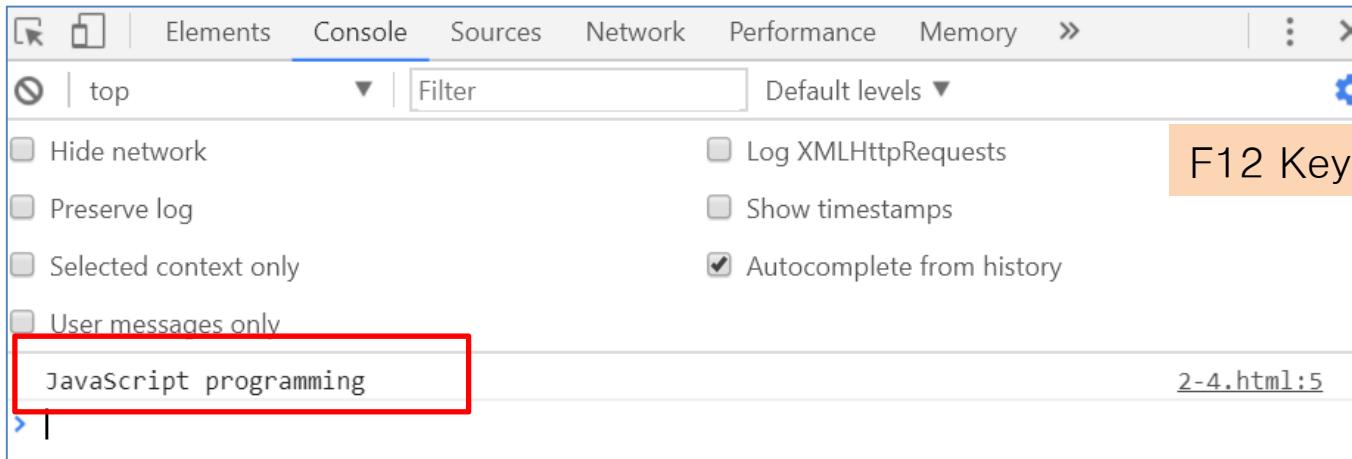
## ❖ 출력 메소드

- console.log( ) 메소드

console.log("문자열")

그림 2-1 console.log() 메소드의 형태

```
<head>
  <script>
    console.log("JavaScript programming");
  </script>
</head>
```

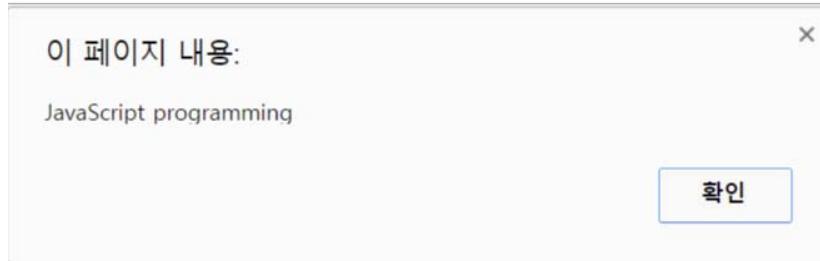


# 출력

## ❖ alert() 메소드

- 웹브라우저에 경고창 출력

```
<script>
    alert("JavaScript programming");
</script>
```



```
<script>
    alert('동해물과 백두산이\n마르고 닳도록');
</script>
```



# 변수 선언

## ❖ 변수 선언

- var 키워드 : 함수 단위 scope

```
function () {
    for (var i = 0; i < 10; i++) {

    }

    console.log(i); //--> 10
}

function () {
    var i;

    for (i = 0; i < 10; i++) {

    }

    console.log(i); //--> 10
}
```

```
// var 변수
var foo = 'foo1';
console.log(foo);

if (true) {
    var foo = 'foo2';
    console.log(foo);
}

console.log(foo); ?
```

- Let : 블록 단위 scope

```
function () {
    for (let i = 0; i < 10; i++) {

    }

    console.log(i); //--> undefined
}
```

```
let j = 0;
for (let j = 0; j < 10; j++) {
    console.log('let j = ' + j);
}
console.log('let j == ' + j); ?
```

# 변수 선언

## ❖ 변수 타입

```
<script>
    var number = 5;                      // 숫자 타입
    var string = '자바스크립트';           // 문자 타입
    var bool = true;                      // boolean(true/false)
    var array = [];                        // 배열
    var object = {} ;                     // 객체
    var who_am_i;                         // 변수에 값을 지정하지 않음 (?)

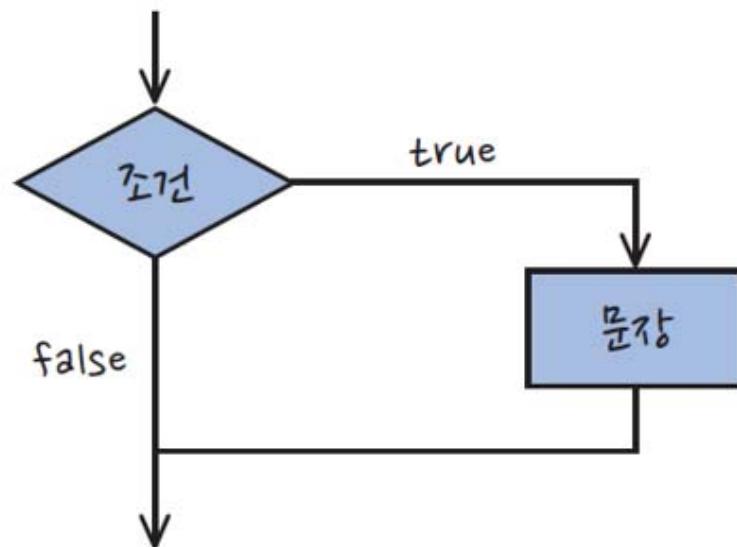
    // 변수의 타입을 출력
    document.write(typeof number + '<br>');      // number
    document.write(typeof string + '<br>');        // string
    document.write(typeof bool + '<br>');          // boolean
    document.write(typeof array + '<br>');         // object
    document.write(typeof object + '<br>');        // object
    document.write(typeof who_am_i + '<br>');       // undefined
</script>
```

# 제어문

## ❖ If 문

```
if (<불 표현식>){  
}
```

- 불 표현식이 true이면 문장을 실행, false이면 문장을 무시함



# 제어문

## ❖ If 문

```
let input = 32;

if (input % 2 == 0) {
    console.log("짝수입니다!");
}

if (input % 2 == 1) {
    console.log("홀수입니다!");
}
```

```
let date = new Date();

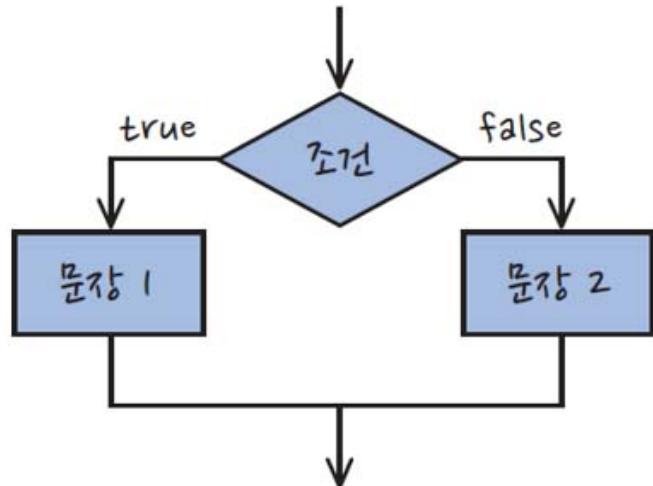
if (date.getHours() < 12) {
    console.log("오전입니다.");
}

if (12 <= date.getHours()) {
    console.log("오후입니다.");
}
```

# 제어문

## ❖ If ~ else 문

```
if (<불 표현식>) {  
    // 불 표현식이 참일 때 실행할 문장  
} else {  
    // 불 표현식이 거짓일 때 실행할 문장  
}
```



# 제어문

## ❖ If ~ else 문

```
let input = 32;  
  
if (input % 2 == 0) {  
    console.log("짝수입니다!");  
} else {  
    console.log("홀수입니다!");  
}
```

```
let date = new Date();  
  
if (date.getHours() < 12) {  
    console.log("오전입니다.");  
} else {  
    console.log("오후입니다.");  
}
```

# 제어문

## ❖ 중첩 if 문

```
if (불 표현식) {  
    if (불 표현식) {  
        문장;  
    } else {  
        문장;  
    }  
} else {  
    if (불 표현식) {  
        문장;  
    } else {  
        문장;  
    }  
}
```

```
let date = new Date();  
let hours = date.getHours();  
  
if (hours < 11) {  
    console.log("아침 먹을 시간입니다.");  
} else {  
    if (hours < 15) {  
        console.log("점심 먹을 시간입니다.");  
    } else {  
        console.log("저녁 먹을 시간입니다.");  
    }  
}
```

# 제어문

## ❖ If ~ else if 문

```
if (<불 표현식>) {  
  
} else if (<불 표현식>) {  
  
} else if (<불 표현식>) {  
  
} else {  
  
}
```

```
let date = new Date();  
let hours = date.getHours();  
  
if (hours < 11) {  
    console.log("아침 먹을 시간입니다.");  
} else if (hours < 15) {  
    console.log("점심 먹을 시간입니다.");  
} else {  
    console.log("저녁 먹을 시간입니다.");  
}
```

# 제어문

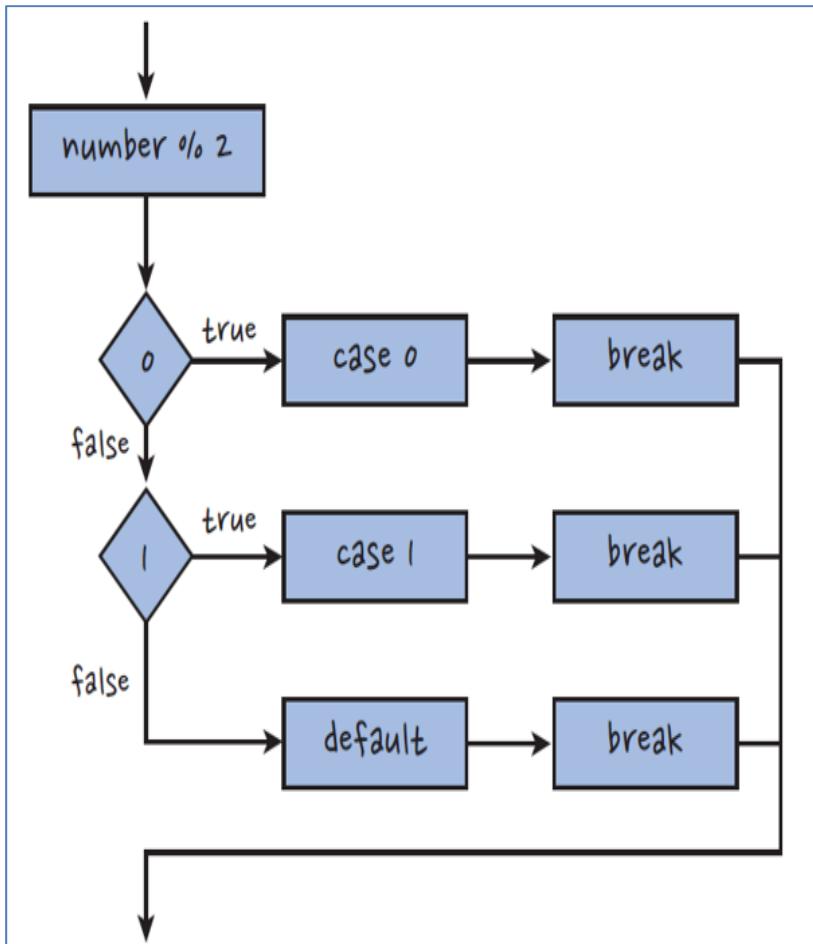
## ❖ switch case 문

```
switch (<비교할 값>) {  
    case <값>:  
        <문장>  
        break;  
    case <값>:  
        <문장>  
        break;  
    default:  
        <문장>  
        break;  
}
```

```
let input = 32;  
  
switch (input % 2) {  
    case 0:  
        console.log("짝수입니다.");  
        break;  
    case 1:  
        console.log("홀수입니다.");  
        break;  
}
```

# 제어문

## ❖ switch case 문



```
<script>
  var answer = 3; // 답을 입력

  switch(answer) {
    case 1:
      msg='틀렸습니다';
      break;
    case 2:
      msg='틀렸습니다';
      break;
    case 3:
      msg='정답입니다';
      break;
    case 4:
      msg='틀렸습니다';
      break;
    default:
      // 기타
      break;
  }
  document.write(msg); // 정답입니다
</script>
```

# 제어문

## ❖ switch case 문

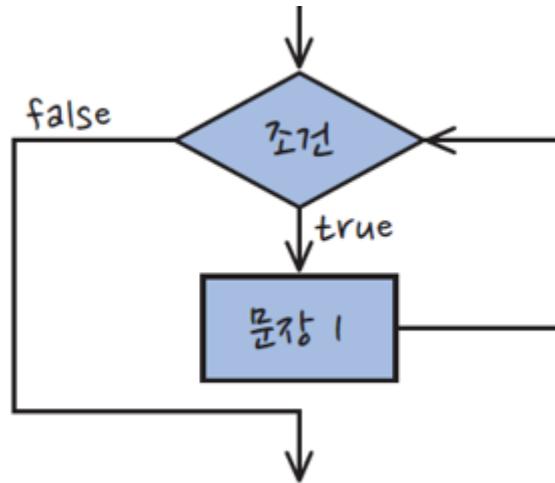
```
let date = new Date();

switch (date.getMonth() + 1) {
    case 12:
    case 1:
    case 2:
        console.log("겨울입니다.");
        break;
    case 3:
    case 4:
    case 5:
        console.log("봄입니다.");
        break;
    case 6:
    case 7:
    case 8:
        console.log("여름입니다.");
        break;
    case 9:
    case 10:
    case 11:
        console.log("가을입니다.");
        break;
    default:
        console.log("대체 어떤 행성에 살고 계신가요?");
        break;
}
```

# 반복문

## ❖ While 문

```
while (<불 표현식>) {  
    // 불 표현식이 참인 동안 실행할 문장  
}
```

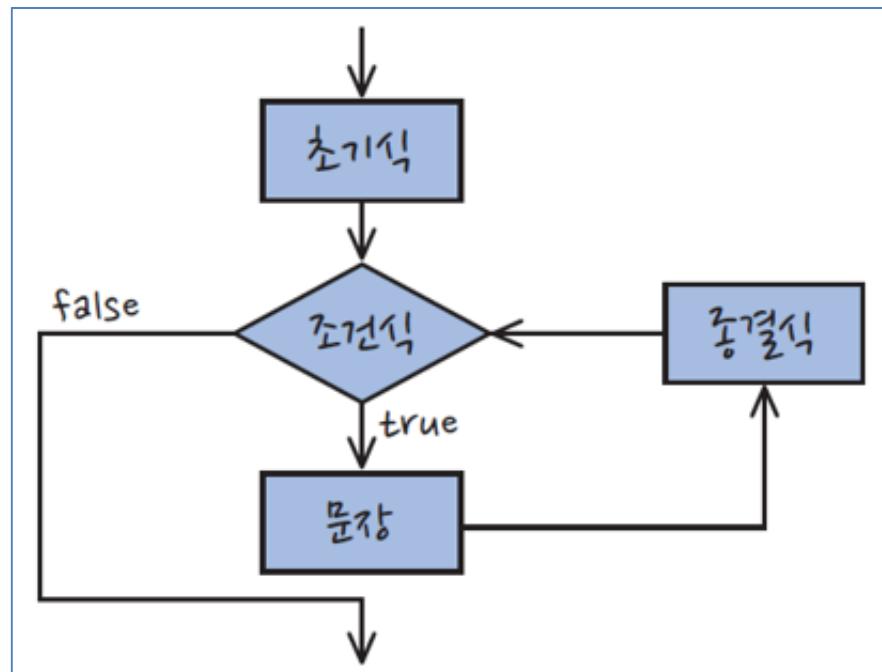


```
// 변수를 선언합니다.  
let i = 0;  
let array = [52, 273, 32, 65, 103];  
  
// 반복을 수행합니다.  
while (i < array.length) {  
    // 출력합니다.  
    console.log(i + "번째 출력:" + array[i]);  
  
    // 탈출하려고 변수를 더합니다.  
    i++;  
}
```

# 반복문

## ❖ for 문

```
for (let i = 0; i < <반복 횟수>; i++) {  
}
```



```
// 변수를 선언합니다.  
let output = 0;  
  
// 반복을 수행합니다.  
for (let i = 0; i <= 100; i++) {  
    output += i;  
}  
  
// 출력합니다.  
console.log(output);
```

# 반복문

## ❖ for in 문

```
for (let 인덱스 in 배열) {  
}  
}
```

## ❖ for of 문

```
for (let 요소 of 배열) {  
}  
}
```

```
// 변수를 선언합니다.  
let array = ["사과", "배", "포도", "딸기", "바나나"];  
  
// 반복을 수행합니다.  
for (let i in array) {  
    // 출력합니다.  
    console.log(`${i}번째 요소: ${array[i]}`);  
}  
  
console.log("----- 구분선 -----");  
  
// 반복을 수행합니다.  
for (let item of array) {  
    // 출력합니다.  
    console.log(item);  
}
```

```
0번째 요소: 사과  
1번째 요소: 배  
2번째 요소: 포도  
3번째 요소: 딸기  
4번째 요소: 바나나  
----- 구분선 -----  
사과  
배  
포도  
딸기  
바나나
```

# 반복문

## ❖ break 문

```
let i = 0;
let array = [1, 31, 273, 57, 8, 11, 32];
let output;

while (true) {
    if (array[i] % 2 == 0) {
        output = array[i];
        break;      반복문을 빠져나감
    }

    i = i + 1;
}

console.log(`처음 발견한 짝수는 ${output}입니다.`)
```

# 반복문

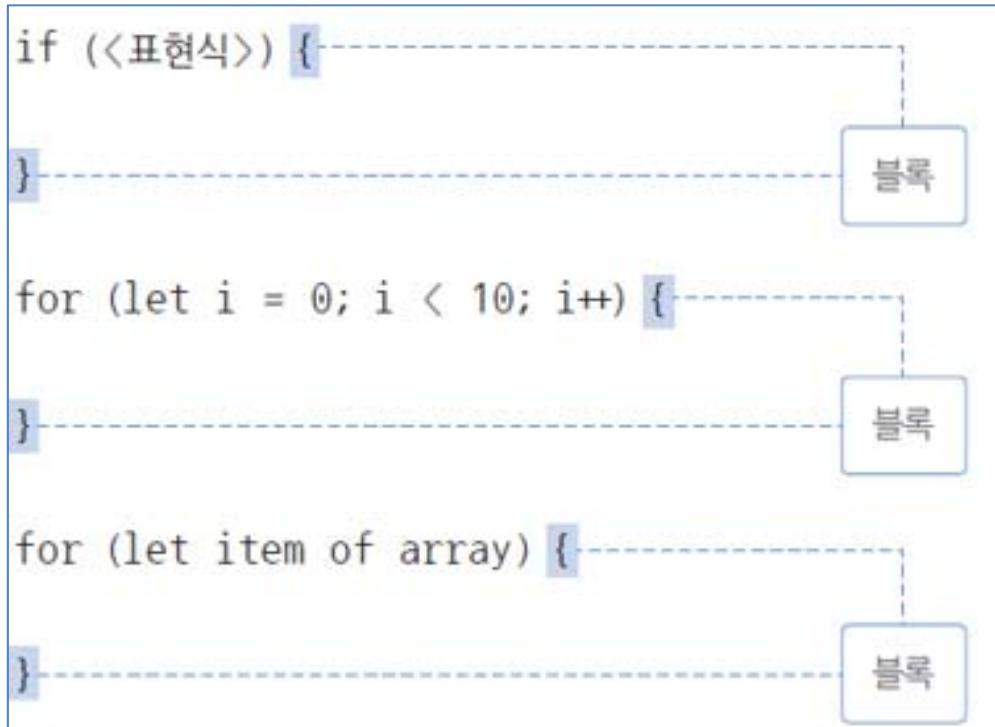
## ❖ continue 문

```
for (let i = 1; i < 10; i++) {  
    if (i % 2 == 0) {  
        continue;  
    }  
    console.log(i)  
}
```

짝수라면 다음 반복으로 바로 넘어갑니다.  
따라서 이 다음 코드는 실행되지 않습니다.

# scope

- ❖ 변수를 사용할 수 있는 범위



# scope

- ❖ 블록 내부에 선언된 변수는 해당 변수 내부에서만 사용 가능

```
{  
    let a = 10;  
}  
  
console.log(a); X
```

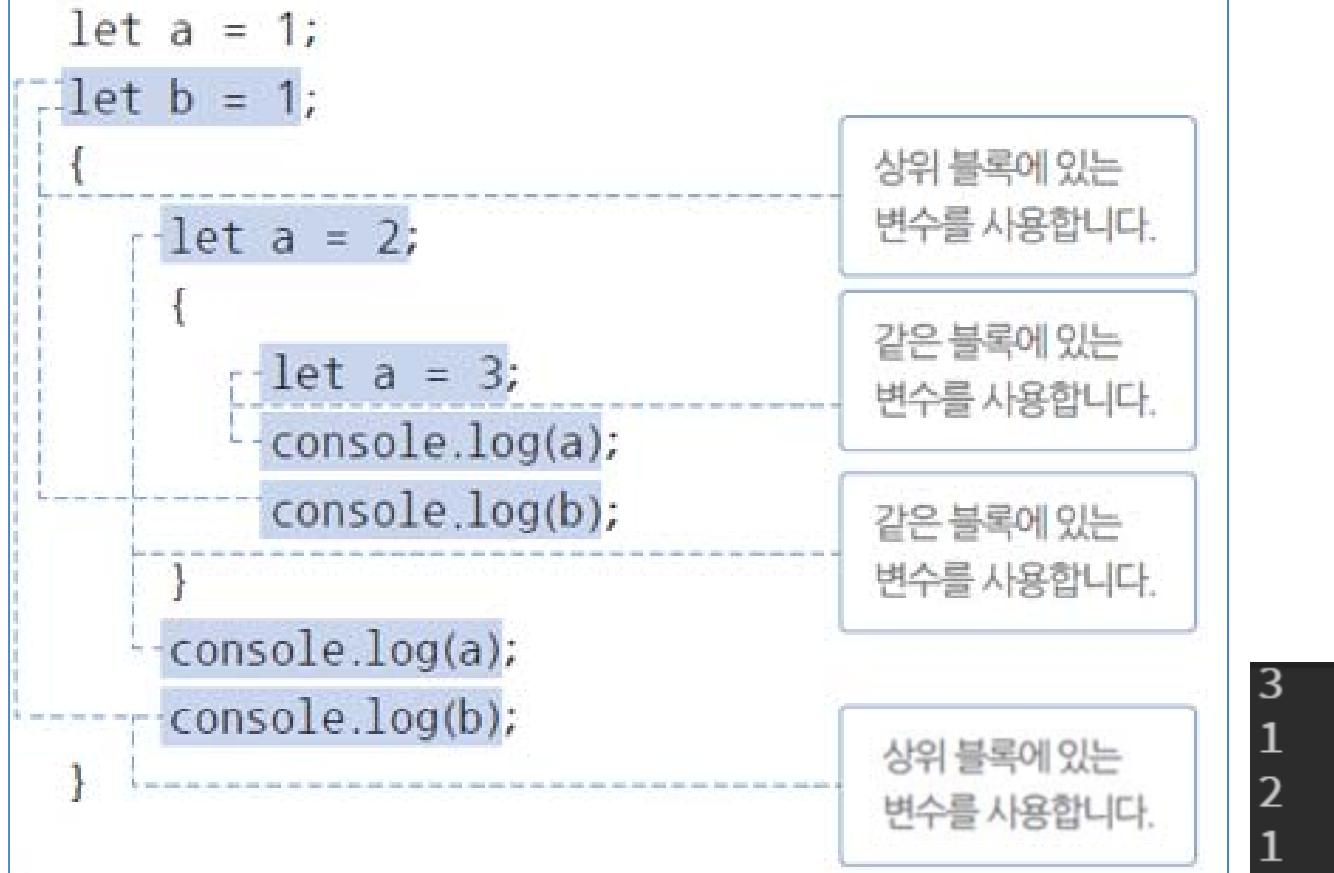
```
for (let i = 0; i < 3; i++) {  
    console.log(i);  
}
```

```
console.log(i);
```

변수 i는 해당 블록 외부에서  
사용할 수 없습니다.

# scope

## ❖ 변수의 적용 범위



# 함수

## ❖ 익명 함수

- 이름을 붙이지 않고 함수 생성

```
let <함수 이름> = function () { };
```

```
let 함수 = function () {
    console.log("함수의 첫 번째 줄");
    console.log("함수의 두 번째 줄");
};
```

```
함수();
console.log(함수);
```

함수를 생성합니다.

함수를 호출합니다.

함수 자체를 출력합니다.

```
함수의 첫 번째 줄
함수의 두 번째 줄
[Function: 함수]
```

# 함수

## ❖ 선언적 함수

- 이름을 붙여서 함수 생성

```
function <함수 이름>() { }
```

```
function 함수() {  
    console.log("함수의 첫 번째 줄");  
    console.log("함수의 두 번째 줄");  
};
```

```
함수();  
console.log(함수);
```

함수를 생성합니다.

함수를 호출합니다.

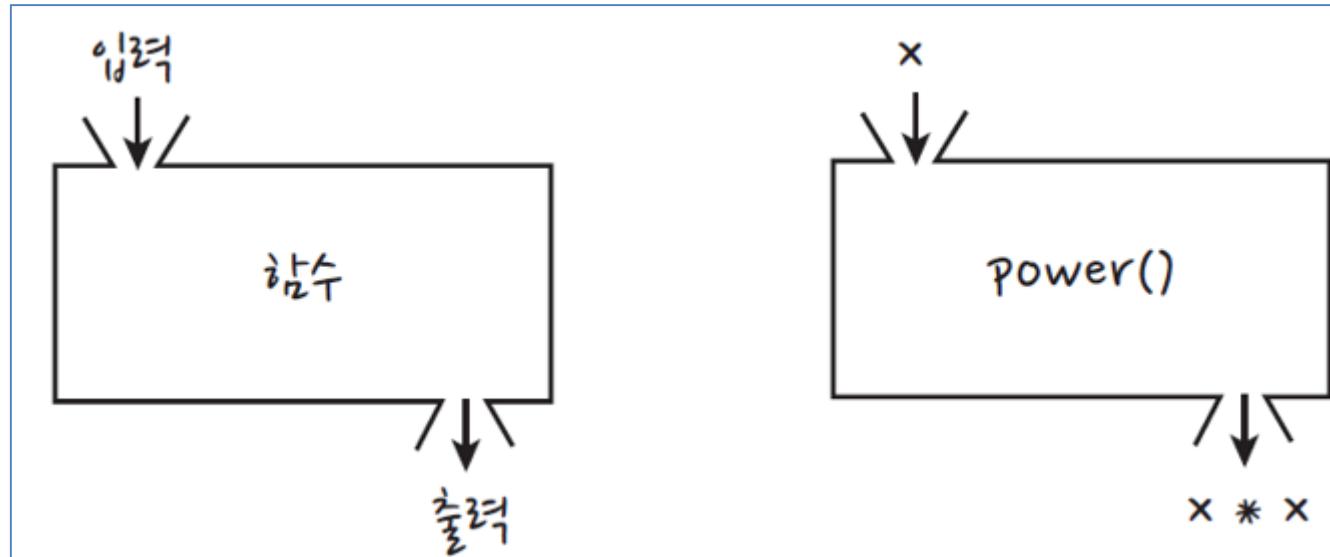
함수를 출력합니다.

```
함수의 첫 번째 줄  
함수의 두 번째 줄  
[Function: 함수]
```

# 함수

## ❖ 매개변수와 리턴 값

```
function <함수 이름>(<매개 변수>) {  
    <함수 코드>  
    return <리턴 값>  
}
```



# 함수

## ❖ 매개변수와 리턴 값

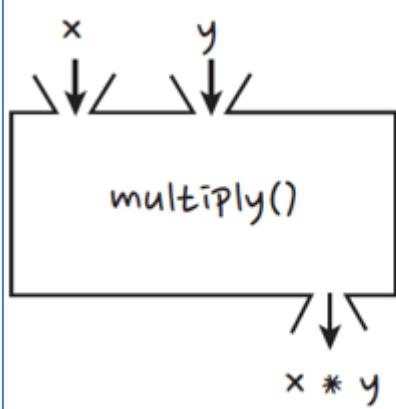
```
function <함수 이름>(<매개 변수>) {  
    <함수 코드>  
    return <리턴 값>  
}
```

```
function power(x) {  
    return x * x;  
}
```

```
console.log(power(10));  
console.log(power(20));
```

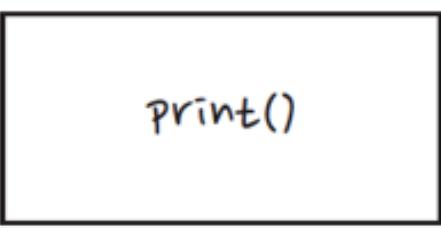
# 함수

## ❖ 매개변수가 여러 개인 함수



```
function multiply(x, y) {  
    return x * y;  
}  
  
console.log(multiply(52, 273));  
console.log(multiply(103, 32));
```

## ❖ 리턴 값이 없는 함수



```
function print(message) {  
    console.log(`"${message}"(이)라고 말했습니다!`);  
}  
  
print("안녕하세요");  
print("뿌잉뿌잉");
```

# 함수

- ❖ return 키워드

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

- ✓ 함수가 실행되는 도중에 return 문을 만나면 함수를 호출한 곳으로 돌아간다.

# 콜백 함수

- ❖ 함수의 매개 변수로 전달되는 함수
    - 함수도 하나의 자료형이므로 매개변수로 전달 할 수 있음

```
<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>
```

수수수수수수수수수수

# 콜백 함수

- ❖ 함수의 매개 변수로 전달되는 함수
    - 익명 함수를 매개변수로 전달

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

    // 함수를 호출합니다 (익명함수를 매개변수로 전달).
    callTenTimes(function () {
        alert('함수 호출');
    });
</script>
```

호호호호호호호호호호

# 표준 내장 함수

## ❖ 숫자 변환 함수

- 문자열을 숫자로 변환

| 함수           | 설명              |
|--------------|-----------------|
| parseInt()   | 문자열을 정수로 변환합니다. |
| parseFloat() | 문자열을 실수로 변환합니다. |

```
// 변수를 선언합니다.  
let inputA = "52";  
let inputB = "52.273";  
let inputC = "1401동"  
  
// parseInt() 함수의 기본적인 사용  
console.log(parseInt(inputA))  
  
// parseInt() 함수와 parseFloat() 함수의 차이  
console.log(parseInt(inputB))  
console.log(parseFloat(inputB))  
  
// 문자열 뒤에 숫자가 아닌 문자가 포함되어 있을 때  
console.log(parseInt(inputC));
```

52  
52  
52.273  
1401

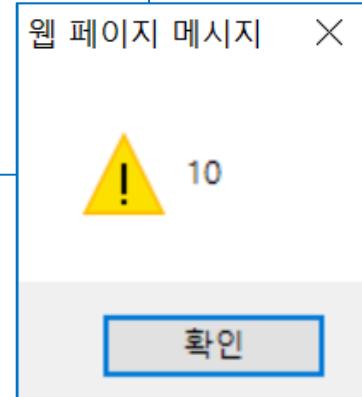
# 표준 내장 함수

## ❖ 코드 실행 함수

- 문자열을 자바스크립트 코드로 실행

eval(string)

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



# 표준 내장 함수

## ❖ 타이머 함수

- 특정 시간 후에 또는 특정 시간마다 어떤 일을 할 때 사용
- 시간은 밀리초로 지정

| 함수                  | 설명                  |
|---------------------|---------------------|
| setTimeout(함수, 시간)  | 특정 시간 후에 함수를 실행합니다. |
| setInterval(함수, 시간) | 특정 시간마다 함수를 실행합니다.  |

```
// 1초 후에
setTimeout(function () {
    console.log("1초가 지났습니다.");
}, 1000);

// 1초마다
setInterval(function () {
    console.log("1초 마다 호출됩니다.");
}, 1000);
```

1초가 지났습니다.  
1초 마다 호출됩니다.  
1초 마다 호출됩니다.  
1초 마다 호출됩니다.  
^C

# 표준 내장 함수

## ❖ 타이머 제거 함수

| 함수                 | 설명                         |
|--------------------|----------------------------|
| clearInterval(아이디) | 특정 시간마다 실행하던 함수 호출을 정지합니다. |

```
// 1초마다
let id = setInterval(function () {
    console.log("출력합니다.");
}, 1000);

// 3초 후에
setTimeout(function () {
    // 타이머를 제거합니다.
    clearInterval(id);
}, 3000);
```

# 배열

## ■ 배열

- 여러 개의 자료를 한꺼번에 다룰 수 있는 자료형
- 대괄호 내부의 각 자료는 ,로 구분
- 배열에는 여러 자료형이 섞여 있을 수 있음
- 요소 : 배열 안에 들어 있는 각 자료

```
let 이름 = [자료, 자료, 자료, 자료, 자료]
```

- 배열 요소

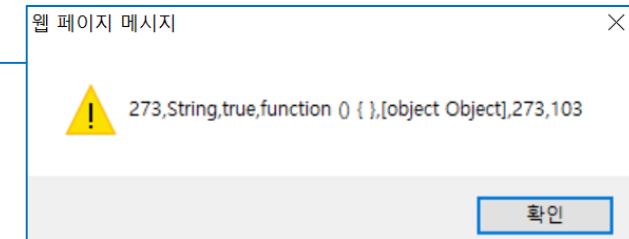
배열[인덱스]  
└─> 요소

```
<script>
    // 배열을 선언합니다.
    var array = [273, 32, 103, 57, 52];
</script>
```

# 배열

## ■ 배열

```
<script>
    // 배열을 선언합니다.
    var array = [273, 'String', true, function () { }, {}, [273, 103]];
    // 출력합니다.
    alert(array);
</script>
```



```
<script>
    // 배열을 선언합니다.
    var array = [273, 32, 103, 57, 52];
    // 출력합니다.
    alert(array[0]);
    alert(array[2]);
    alert(array[4]);
</script>
```

# 배열

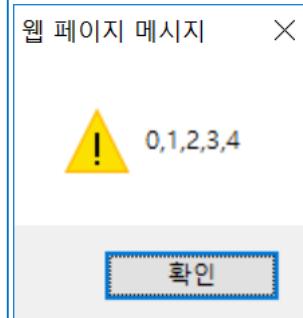
- 배열

- 배열의 크기

```
<script>
    var arrayA = [0, 1, 2, 3];
    var arrayB = [0, 1, 2, 3, 4, 5, 6];
    alert("length of A: " + arrayA.length);
    alert("length of B: " + arrayB.length);
</script>
```

- 배열 요소 추가

```
<script>
    // 배열을 생성합니다.
    var array = [0, 1];
    // 배열에 요소를 추가합니다.
    array.push(2);
    array.push(3);
    array.push(4);
    // 출력합니다.
    alert(array);
</script>
```



# 객체

## ❖ 객체 선언

```
<script>
    // 객체를 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                      |

# 객체

## ❖ 객체 접근

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

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



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

# 객체

## ❖ 객체 생성 및 접근

| 속성(키) | 값     |
|-------|-------|
| name  | '바나나' |
| price | 1200  |

```
// 객체를 선언합니다.  
let object = {  
    name: '바나나',  
    price: 1200  
};  
  
// 출력합니다.  
console.log(object.name);  
console.log(object.price);
```

# 객체

## ❖ 속성과 메서드

- 속성 : 객체 내부에 있는 값 하나하나
- 메서드 : 속성 중 함수 자료형인 속성
- 객체는 다양한 자료형을 가질 수 있음

```
<script>
    // 객체는 다양한 자료형을 가질 수 있음
    var object = {
        number: 273,
        string: 'RintIanTta',
        boolean: true,
        array: [52, 273, 103, 32],
        method: function () {
            ...
        }
    };
</script>
```

# 객체

## ❖ 속성과 메서드

```
let object = {  
    name: '바나나', } 속성  
    price: 1200,  
    print: function () { 메서드  
        console.log(`#${this.name}의 가격은 ${this.price}원입니다.`)  
    }  
};
```

## ❖ 메서드 호출

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

# 객체

## ❖ 속성과 메서드

- 메서드 내부에서 this 키워드

```
// 객체를 선언합니다.  
let object = {  
    name: '바나나',  
    price: 1200,  
    print: function () {  
        console.log(` ${this.name}의 가격은 ${this.price}원입니다.`)  
    }  
};  
  
// 메소드를 호출합니다.  
object.print();
```

# 객체

## ❖ 객체와 반복문

### ▪ for in문

```
//객체 선언
let product = {
    제품명 : '갤럭시노트 8',
    유형   : '스마트폰 노트',
    제조사 : '삼성전자',
    가격    : 1200000
};
```

```
//for in문
for (let key in product) {
    console.log(`#${key}: ${product[key]}`);
}
```

제품명: 갤럭시노트 8  
유형: 스마트폰 노트  
제조사: 삼성전자  
가격: 1200000

# 객체

## ❖ 객체와 반복문

### ▪ for of 문

```
//배열과 객체를 사용하면 여러개의 데이터를 쉽게 다룰 수 있음
let products = [
    {name:'바나나', price:1200},
    {name:'사과', price:1500},
    {name:'배', price:2000}
];
//함수를 외부에 만든 경우
function printProduct(key) {
    console.log(` ${key.name}의 가격은 ${key.price}원입니다.`)
}

for (let key of products) {
    printProduct(key);
}
```

바나나의 가격은 1200원입니다.  
사과의 가격은 1500원입니다.  
배의 가격은 2000원입니다.

# 객체

## ❖ 객체와 반복문

### ▪ for of 문

```
//메소드를 가진 객체의 배열
let productList = [
    {
        name: '바나나',
        price: 1200,
        print: function() {
            console.log(`#${this.name}의 가격은 ${this.price}입니다.`)
        }
    },
    {
        name: '사과',
        price: 1500,
        print: function() {
            console.log(`#${this.name}의 가격은 ${this.price}입니다.`)
        }
    },
    {
        name: '배',
        price: 2500,
        print: function() {
            console.log(`#${this.name}의 가격은 ${this.price}입니다.`)
        }
    },
    {
        name: '봉숭아',
        price: 3500,
        print: function() {
            console.log(`#${this.name}의 가격은 ${this.price}입니다.`)
        }
    }
];
```

```
//반복 출력
for (let key of productList) {
    key.print();
}
```

바나나의 가격은 1200입니다.  
사과의 가격은 1500입니다.  
배의 가격은 2500입니다.  
봉숭아의 가격은 3500입니다.

# 객체

## ❖ 객체 관련 키워드

- with 키워드
  - 동적 코드 생성시 코드를 짧게 줄여주는 키워드

```
<script>
    // 객체를 선언합니다.
    var student = {
        이름: '홍길동',
        국어: 97, 수학: 98,
        영어: 96, 과학: 99
    };
    // 출력합니다.
    var output = '';
    output += '이름: ' + student.이름 + '\n';
    output += '국어: ' + student.국어 + '\n';
    output += '수학: ' + student.수학 + '\n';
    output += '영어: ' + student.영어 + '\n';
    output += '과학: ' + student.과학 + '\n';
    output += '총점: ' + (student.국어 + student.수학 + student.영어 + student.과학);
    alert(output);
</script>
```

이름: 홍길동  
국어: 97  
수학: 98  
영어: 96  
과학: 99  
총점: 390

확인

# 객체

## ❖ 객체 관련 키워드

- with 키워드
- 동적 코드 생성시 코드를 짧게 줄여주는 키워드

```
<script>
    // 객체를 선언합니다.
    var student = {
        이름: '홍길동',
        국어: 97, 수학: 98,
        영어: 96, 과학: 99
    };
    // 출력합니다.
    var output = '';

    with (student) {
        output += '이름: ' + 이름 + '\n';
        output += '국어: ' + 국어 + '\n';
        output += '수학: ' + 수학 + '\n';
        output += '영어: ' + 영어 + '\n';
        output += '과학: ' + 과학 + '\n';
        output += '총점: ' + (국어 + 수학 + 영어 + 과학);
    }
    alert(output);
</script>
```

```
with(객체명) {
    code
}
```

이름: 홍길동  
국어: 97  
수학: 98  
영어: 96  
과학: 99  
총점: 390

확인

# 객체

## ❖ 객체 속성 추가

- 동적으로 속성 추가

```
<script>
    // 객체 선언
    var student = {};

    // 객체에 속성 추가
    student.이름 = '홍길동';
    student.취미 = '운동';
    student.특기 = '프로그래밍';
    student.장래희망 = '컨설턴트';

    // toString() 메서드 : 객체에 있는 속성을 출력하는 메서드
    student.toString = function () {
        var output = '';
        for (var key in this) {
            // toString() 메서드는 출력하지 않게 합니다.
            if (key != 'toString') {
                output += key + ':' + this[key] + '\n';
            }
        }
        return output;
    };
    // 출력
    alert(student.toString());
</script>
```

이름 : 홍길동  
취미 : 운동  
특기 : 프로그래밍  
장래희망 : 컨설턴트

확인

# 객체

## ❖ 객체 속성 삭제

- 동적으로 속성 삭제 : delete 키워드

```
<script>
    // 변수 선언
    var student = {};

    // 객체에 속성 추가
    student.이름 = '홍길동';
    student.취미 = '등산';
    student.특기 = '프로그래밍';
    student.장래희망 = '컨설턴트';

    // toString() 메서드를 만듭니다.
    student.toString = function () {
        var output = '';
        for (var key in this) {
            // toString() 메서드는 출력하지 않게 합니다.
            if (key != 'toString') {
                output += key + ':' + this[key] + '\n';
            }
        }
        return output;
    };
    // 출력
    alert(student.toString());

    // 속성 제거
    delete (student.장래희망);

    // 출력
    alert(student);
    // (toString() 메서드를 사용하지 않고, student 객체 출력)
</script>
```

이름 : 홍길동  
취미 : 등산  
특기 : 프로그래밍  
장래희망 : 컨설턴트

확인

이름 : 홍길동  
취미 : 등산  
특기 : 프로그래밍

확인

# 객체

## ❖ 객체에서 배열을 이용한 데이터 관리

```
<script>
    // 학생 정보 배열 객체 생성
    var students = [];

    // 배열 객체에 학생 정보 추가
    students.push({ 이름: '홍길동', 국어: 87, 수학: 98, 영어: 88, 과학: 95 });
    students.push({ 이름: '손지수', 국어: 92, 수학: 98, 영어: 96, 과학: 98 });
    students.push({ 이름: '이정은', 국어: 76, 수학: 96, 영어: 94, 과학: 90 });
    students.push({ 이름: '장하나', 국어: 98, 수학: 92, 영어: 96, 과학: 92 });
    students.push({ 이름: '고진영', 국어: 95, 수학: 98, 영어: 98, 과학: 98 });

    // student 배열의 각각의 객체에 메서드를 추가
    for (var i in students) {
        // 총점을 구하는 메서드를 추가
        students[i].getSum = function () {
            return this.국어 + this.수학 + this.영어 + this.과학;
        };
        // 평균을 구하는 메서드를 추가
        students[i].getAverage = function () {
            return this.getSum() / 4;
        };
    }

    // 출력
    var output = '이름      총점      평균\n';
    for (var i in students) {
        with (students[i]) {
            output += 이름 + '      ' + getSum() + '      ' + getAverage() + '\n';
        }
    }
    alert(output);
</script>
```

| 이름  | 총점  | 평균    |
|-----|-----|-------|
| 홍길동 | 368 | 92    |
| 손지수 | 384 | 96    |
| 이정은 | 356 | 89    |
| 장하나 | 378 | 94.5  |
| 고진영 | 389 | 97.25 |

# 생성자 함수

## ❖ 생성자 함수

- 객체를 생성할 때 사용하는 함수
- 인스턴스 : 생성자 함수를 사용하여 만든 객체
- 생성자 함수 이름은 대문자로 시작

```
<script>
    // 생성자 함수 선언
    function Product(name, no) {
        this.name = name;
        this.no = no;
    }

    // 객체(instance) 생성
    let productInstance = new Product('홍길동', 2017001);

    // 출력
    alert('이름 : ' + productInstance.name + '\n' +
          '학번 : ' + productInstance.no);
</script>
```

이름 : 홍길동  
학번 : 2017001

# 생성자 함수

## ❖ 생성자 함수

```
<script>
    // 생성자 함수 선언
    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 + ' ' + this.getSum() + ' ' + this.getAverage();
        };
    }

    // 학생 정보 배열 객체 생성
    var students = [];

    // 생성자 함수를 사용하여 객체를 생성하여 학생정보 배열에 추가
    students.push(new Student('홍길동', 87, 98, 88, 95));
    students.push(new Student('손지수', 92, 98, 96, 98));
    students.push(new Student('이정은', 76, 96, 94, 90));
    students.push(new Student('장하나', 98, 92, 96, 92));
    students.push(new Student('고진영', 95, 98, 98, 98));

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

| 이름  | 총점  | 평균    |
|-----|-----|-------|
| 홍길동 | 368 | 92    |
| 손지수 | 384 | 96    |
| 이정은 | 356 | 89    |
| 장하나 | 378 | 94.5  |
| 고진영 | 389 | 97.25 |

# 프로토타입

## ❖ 프로토타입

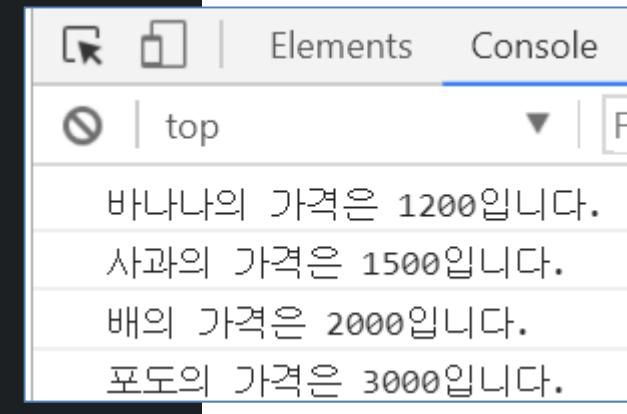
- 생성자 함수로 생성한 객체들이 공동으로 갖는 공간
- 일반적으로 메서드를 프로토타입으로 선언하여 공동으로 사용

```
<script>
    // 생성자 함수 선언
    function ProductConst(name, price) {
        this.name = name;
        this.price = price;
    }

    //프로토타입에 메소드 선언
    ProductConst.prototype.print = function() {
        console.log(`#${this.name}의 가격은 ${this.price}입니다.`);
    };

    //상품 목록을 선언(객체 생성)
    let productsList = [
        new ProductConst('바나나', 1200),
        new ProductConst('사과', 1500),
        new ProductConst('배', 2000),
        new ProductConst('포도', 3000)
    ];

    //for()으로 출력
    for (let key of productsList) {
        key.print();
    }
</script>
```



# 내장 객체

## ❖ Object 객체

- 자바스크립트의 최상위 객체
  - `toString()` 메서드 : 객체를 문자열로 변환하는 메서드

```
<script>
    // 객체 선언
    var student = {
        name: '홍길동',
        grade: '3학년',
        //모든 객체는 toString()메서드를 갖는데 재선언함
        toString: function () {
            return this.name + ' : ' + this.grade;
        }
    };
    // 출력
    alert(student);
</script>
```

홍길동 : 3학년

# 내장 객체

## ❖ Number 객체

- 자바스크립트에서 숫자를 표현할 때 사용
  - toFixed() 메서드 : 소수점 자리를 자르는 메서드

```
<script>
    // 변수를 선언합니다.
    var number = 273.5210332;
    // 출력합니다.
    var output = '';
    output += number.toFixed(1) + '\n';
    output += number.toFixed(4);
    alert(output);
</script>
```

273.5  
273.5210

# 내장 객체

## ❖ String 객체

- length 속성 : 문자열의 길이를 나타냄

```
<script>
    // 변수를 선언합니다.
    var characters = prompt('사용할 비밀번호를 입력해주세요.', '6글자 이상');
    // 출력합니다.
    if (characters.length < 6) {
        alert('6글자 이상으로 입력하세요.');
    } else {
        alert('잘했어요!');
    }
</script>
```

사용할 비밀번호를 입력해주세요.

6글자 이상

확인

취소

잘했어요!

# 내장 객체

## ❖ String 객체

- 자바스크립트에서 가장 많이 사용하는 객체

| 메소드                                 | 설명                                 |
|-------------------------------------|------------------------------------|
| charAt(position)                    | position에 위치하는 문자를 리턴합니다.          |
| charCodeAt(position)                | position에 위치하는 문자의 유니코드 번호를 리턴합니다. |
| concat(args)                        | 매개 변수로 입력한 문자열을 이어 리턴합니다.          |
| indexOf(searchString, position)     | 앞에서부터 일치하는 문자열의 위치를 리턴합니다.         |
| lastIndexOf(searchString, position) | 뒤에서부터 일치하는 문자열의 위치를 리턴합니다.         |
| match(regExp)                       | 문자열 안에 regExp가 있는지 확인합니다.          |
| replace(regExp, replacement)        | regExp를 replacement로 바꾼 후 리턴합니다.   |
| search(regExp)                      | regExp와 일치하는 문자열의 위치를 리턴합니다.       |
| slice(start, end)                   | 특정 위치의 문자열을 추출해 리턴합니다.             |
| split(separator, limit)             | separator로 문자열을 잘라 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.               |

# 내장 객체

## ❖ String 객체

- `indexOf()` 메소드 : 특정 문자열이 있는지 확인  
문자열이 포함되어 있지 않을 때는 -1을 리턴

```
// 변수를 선언합니다.  
let string = '안녕하세요. 좋은 아침입니다.';  
  
// 문자열 내부에 "아침"이라는 문자열이 있는지 확인합니다.  
if (string.indexOf('아침') >= 0) {  
    console.log('좋은 아침이에요...!');  
}
```

좋은 아침이에요...!

# 내장 객체

## ❖ String 객체

- `split()` 메소드 : 특정한 기호를 기반으로 문자열을 분해

```
// 변수를 선언합니다.  
let string = '감자,고구마,바나나,사과';  
  
// 문자열을 쉼표로 자르고 출력합니다.  
let array = string.split(',');  
console.log(array);
```

[ '감자', '고구마', '바나나', '사과' ]

# 내장 객체

## ❖ String 객체

- trim() 메소드 : 문자열 양쪽 끝의 공백 제거

```
<script>
    // 변수를 선언
    var text = ' text ';

    // trim() —로 공백제거
    var output = '';
    output += '++' + text + '++\n';
    output += '++' + text.trim() + '++';

    // 출력
    alert(output);
</script>
```

++ text ++  
++text++

# 내장 객체

## ❖ Array 객체

- length 속성 : 요소의 개수를 알아냄

```
<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>
```

# 내장 객체

## ❖ Array 객체

### ▪ Array 객체의 메서드

| 메소드        | 설명                                      |
|------------|---|
| concat()   | 매개 변수로 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.      |

\*표시된 메소드는 자기 자신을 변화시킵니다.

# 내장 객체

## ❖ Array 객체

- sort() 메서드

```
<script>
    // 배열 선언
    var array = [52, 273, 103, 32];
    //오름차순 소트
    array.sort();
    // 출력
    alert(array);
</script>
```

103,273,32,52

# 내장 객체

## ❖ Array 객체

- sort() 메서드

```
<script>
    // 배열 선언
    var array = [52, 273, 103, 32];

    //오름 차순 소트(문자열 오름차순)
    array.sort();

    // 출력
    alert(array);
</script>
```

103,273,32,52

32,52,103,273

# 내장 객체

## ❖ Array 객체

- sort() 메서드 – 오름차순 소트

```
<script>
    // 배열 선언
    var array = [52, 273, 103, 32];

    // 오름 차순 sort
    array.sort(function (left, right) {
        return left - right;
    });

    // 출력
    alert(array);
</script>
```

32,52,103,273

# 내장 객체

## ❖ Array 객체

- sort() 메서드 – 내림차순 소트

```
<script>
    // 배열 선언
    var array = [52, 273, 103, 32];

    // 내림 차순 sort
    array.sort(function (left, right) {
        return right - left;
    });
    |
    // 출력
    alert(array);
</script>
```

273,103,52,32

# 내장 객체

## ❖ Array 객체

- 요소 제거

```
<script>
    // Array 생성자 함수의 프로토타입에 remove() 메서드를 추가
    Array.prototype.remove = function (index) { this.splice(index, 1); }

    // 배열 선언
    var array = [52, 273, 5, 103, 32, 274, 129, 10, 20];

    // 반복문과 조건문으로 100보다 큰 요소를 제거
    for (var i = array.length - 1; i >= 0; i--) {
        if (array[i] > 100) {
            array.remove(i);
        }
    }
    // 출력
    alert(array);
</script>
```

52,5,32,10,20

# 내장 객체

## ❖ Array 객체

### ▪ 객체 탐색 메서드

```
<script>
    // 배열 선언
    var array = [1, 2, 3, 4, 5, 5, 4, 3, 2, 1];

    /* 검색메서드를 사용
     - indexof() : 특정 요소를 앞쪽부터 검색
     - lastIndexof() : 특정 요소를 앞쪽부터 검색
     * 내부에 검색하려는 객체가 있으면 해당 객체의 인덱스 리턴
     없으면 -1 리턴
    */
    var output1 = array.indexof(4);
    var output2 = array.indexof(8);
    var output3 = array.lastIndexof(4);
    var output4 = array.lastIndexof(1);

    // 출력
    var output = '';
    output += output1 + ' : ' + output2 + '\n';
    output += output3 + ' : ' + output4;
    alert(output);
</script>
```

3 : -1  
6 : 9

# 내장 객체

## ❖ Array 객체

- forEach() 반복 메소드

```
<script>
    // 배열 선언
    var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

    // forEach : 특정함수를 for in 반복문처럼 실행
    var sum = 0;
    var output = '';

    //element: 배열의 요소, index:인덱스, array: 배열
    array.forEach(function (element, index, array) {
        sum += element;
        output += index + ': ' + element + ' → ' + sum + '\n';
    });
    //
    // 출력
    alert(output);
</script>
```

0: 1 → 1  
1: 2 → 3  
2: 3 → 6  
3: 4 → 10  
4: 5 → 15  
5: 6 → 21  
6: 7 → 28  
7: 8 → 36  
8: 9 → 45  
9: 10 → 55

# 내장 객체

## ❖ Array 객체

- map() 메서드 : 배열의 각 요소를 변경해 새로운 배열을 리턴

```
<script>
    // 배열 선언.
    var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

    // map() 메서드 사용
    var output = array.map(function (element) {
        return element * element;//요소를 제곱해 리턴
    });
    // 출력
    alert(output);
</script>
```

1,4,9,16,25,36,49,64,81,100

# 내장 객체

## ❖ Array 객체

- filter() 메서드 : 특정 조건을 만족하는 요소를 추출해 새로운 배열 생성

```
<script>
    // 배열 선언
    var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

    // filter 메서드 사용
    array = array.filter(function (element, index, array) {
        return element <= 5; // 5보다 적거나 같은 요소만 리턴
    });
    // 출력
    alert(array);
</script>
```



# 내장 객체

## ❖ Date 객체

| 생성자 함수  | 설명  |
|---|---|
| <code>new Date()</code>   | 현재 시간으로 Date 객체를 생성합니다.                                     |
| <code>new Date(&lt;유닉스 타임&gt;)</code>   | 유닉스 타임(1970년 1월 1일 00시 00분 00초부터 경과한 밀리초)으로 Date 객체를 생성합니다. |
| <code>new Date(&lt;시간 문자열&gt;)</code>   | 문자열로 Date 객체를 생성합니다.  |
| <code>new Date(&lt;년&gt;, &lt;월 - 1&gt;, &lt;일&gt;, &lt;시간&gt;, &lt;분&gt;, &lt;초&gt;, &lt;밀리초&gt;)</code> | 시간 요소(년, 월 - 1, 일, 시간, 분, 초, 밀리초)를 기반으로 Date 객체를 생성합니다.     |

- Month를 나타내는 '월'은 0부터 시작,  
 $0 \rightarrow 1\text{월}, 11 \rightarrow 12\text{월}$

# 내장 객체

## ❖ Date 객체

```
<script>
    // 현재 시간을 기준으로 Date객체 생성
    var dateA = new Date();
    // 출력
    console.log(dateA);

    // 문자열을 기반으로 Date 객체 생성
    var dateB = new Date('September 23, 2017');
    var dateC = new Date('Septembe 23, 2017 21:24:23');
    console.log(dateB);
    console.log(dateC);

    // 숫자를 사용한 Date 객체 생성
    var dateD = new Date(2017, 9, 23);
    var dateE = new Date(2017, 9, 23, 21, 24, 23);
    console.log(dateD);
    console.log(dateE);
</script>
```

|                                   |
|-----------------------------------|
| Sat Sep 23 2017 21:31:22 GMT+0900 |
| Sat Sep 23 2017 00:00:00 GMT+0900 |
| Sat Sep 23 2017 21:24:23 GMT+0900 |
| Mon Oct 23 2017 00:00:00 GMT+0900 |
| Mon Oct 23 2017 21:24:23 GMT+0900 |

# 내장 객체

## ❖ JSON 객체

- JSON.stringify() : 자바스크립트 객체를 JSON 문자열로 변환

```
<script>
    // 객체 선언
    var object = {
        name: '홍길동',
        region: '경기도 성남시'
    };

    // 출력.
    alert(JSON.stringify(object));
</script>
```

{ "name": "홍길동", "region": "경기도 성남시"}

# 내장 객체

## ❖ JSON 객체

- JSON.parse() : JSON문자열을 자바스크립트 객체로 변환

```
<script>
    // 객체 선언
    var object = {
        name: '홍길동',
        region: '경기도 성남시'
    };

    var copy = JSON.parse(JSON.stringify(object));

    // 출력
    alert(copy.name + ' : ' + copy.region);
</script>
```

홍길동 : 경기도 성남시

# 브라우저 관련 객체

## ❖ window 객체

- window 객체 생성
- open(URL, name, features, replace) : 새로운 window 객체 생성

```
<script>
    /* window 객체 생성
       URL : 열고자하는 페이지 URL
       name : 원도우 이름
       features : 원도우 출력 옵션,
       replace : true/false
       (true일 땐 새 문서가 이전의 문서와 교체,
        false이거나 지정되지 않으면 새 문서는 창의 브라우징 히스토리에 새 항목으로 추가)
    */
    window.open('http://www.gachon.ac.kr', 'child', 'width=900, height=600', true);
</script>
```

# 브라우저 관련 객체

## ❖ screen 객체

- 운영체제 화면의 속성을 갖는 객체

```
<script>
    // 변수를 선언합니다.
    var child = window.open('', '', 'width=300, height=200');
    var width = screen.width;//화면 너비
    var height = screen.height;//화면 높이

    child.moveTo(0, 0);//윈도우 위치를 0,0로 이동
    child.resizeTo(width, height);//윈도우 크기를 지정(절대적 크기)

    // 1초마다 윈도우 크기와 위치를 변경
    setInterval(function () {
        child.resizeBy(-20, -20);//윈도우 크기를 상대적으로 지정
        child.moveBy(10, 10);//윈도우 위치를 상대적으로 이동
    }, 1000);
</script>
```

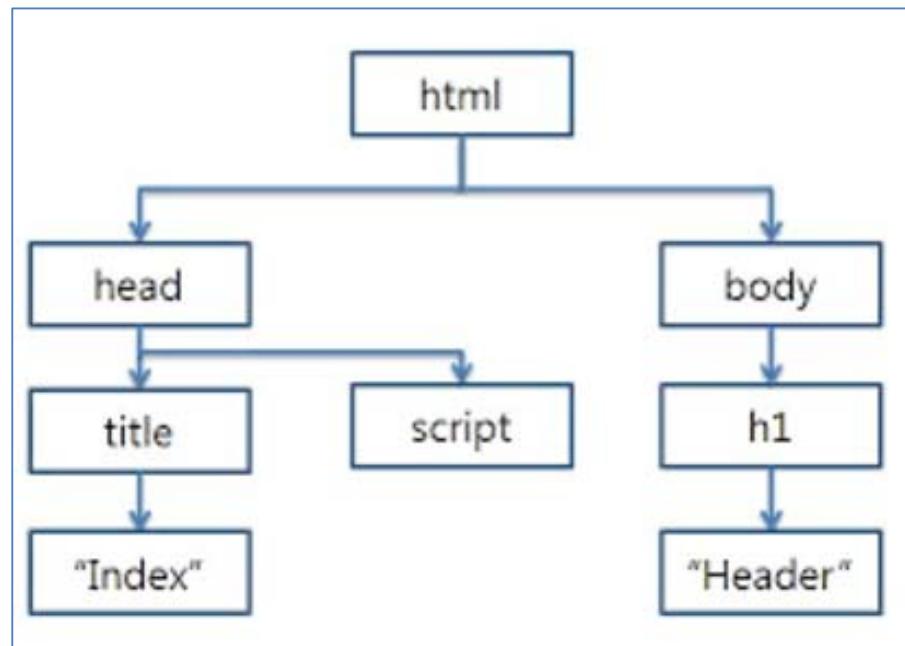
# 브라우저 관련 객체

- ❖ window 객체의 onload 이벤트 속성
  - 윈도우 객체의 로드가 완료되면 자동으로 함수 실행

```
<!DOCTYPE html>
<html>
<head>
    <script>
        //window 객체의 로드가 완료되면 자동으로 함수 실행
        window.onload = function () {
            alert('Process - 0');
        };
    </script>
</head>
<body>
    <h1>Process - 1</h1>
    <h1>Process - 2</h1>
</body>
</html>
```

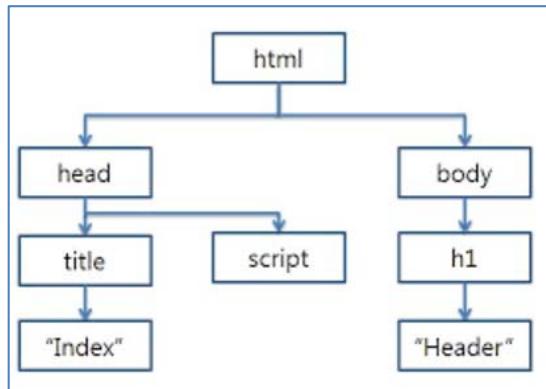
# 문서 객체 모델

- ❖ 문서 객체 모델(DOM : Document Object Model)
  - 문서 객체 모델은 HTML, XML 문서의 프로그래밍 인터페이스
  - DOM은 웹 문서의 구조화된 표현을 제공하며, 프로그래밍 언어가 DOM 구조에 접근할 수 있는 방법을 제공
  - DOM은 웹 페이지를 스크립트 또는 프로그래밍에서 사용할 수 있게 연결시켜주는 역할을 담당



# 문서 객체 모델

## ❖ 문서 객체 모델(DOM : Document Object Model)



```
1  <!DOCTYPE html>
2  ▼ <html>
3  ▼ <head>
4      <title>Index</title>
5  ▼     <script>
6  ▼         window.onload = function () {
7  ▼             /* document 객체의 getElementById() 메서드로 'header' 문서 객체를
8                 자바스크립트로 가져와서 조작할 수 있도록 한다.
9             */
10            var header = document.getElementById('header');
11        };
12    </script>
13  </head>
14 ▼ <body>
15      <h1 id="header">Header</h1>
16  </body>
17  </html>
```

/\* document 객체의 getElementById() 메서드로 'header' 문서 객체를  
자바스크립트로 가져와서 조작할 수 있도록 한다.

# 문서 객체 모델

## ❖ 문서 객체 생성(1)

```
1  <!DOCTYPE html>
2 ▼ <html>
3 ▼ <head>
4 ▼   <script>
5 ▼     window.onload = function () {
6         // 문서객체 생성
7         // 요소노드(객체) 생성
8         var header = document.createElement('h1');
9         //텍스트 노드 생성
10        var textNode = document.createTextNode('Hello DOM');
11
12        // 객체에 노드를 연결
13        header.appendChild(textNode);
14        // 화면에 문서객체(header)를 출력하기 위해 body 문서객체에 연결
15        document.body.appendChild(header);
16    };
17  </script>
18 </head>
19 ▼ <body>
20
21 </body>
22 </html>
```



# 문서 객체 모델

## ❖ 문서 객체 생성(2)

```
<script>
    window.onload = function () {
        // img 태그(문서객체)를 만들어 body에 연결
        var img = document.createElement('img');

        img.src = 'Penguins.jpg';
        img.width = 500;
        img.height = 350;

        /* 웹브라우저가 지원하지 않는 태그의 속성지정
         - setAttribute() 메서드를 사용하여 지정해야 함 */
        /*
        img.setAttribute('src', 'Penguins.jpg');
        img.setAttribute('width', 500);
        img.setAttribute('height', 350); */

        // setAttribute(name, value) 메서드를 사용하여 객체의 속성 지정
        img.setAttribute('data-property', 350);

        // 화면에 문서객체(img)를 출력하기 위해 body 문서객체에 연결
        document.body.appendChild(img);
    };
</script>
```

# 문서 객체 모델

## ❖ 문서 객체 생성(3)

- innerHTML 속성 사용 – 가장 많이 사용

```
<script>
    window.onload = function () {
        // 변수를 선언
        var output = '';
        output += '<ul>';
        output += '    <li>JavaScript</li>';
        output += '    <li>jQuery</li>';
        output += '    <li>Ajax</li>';
        output += '</ul>';

        // body 문서객체의 innerHTML 속성에 문자열을 할당
        document.body.innerHTML = output;
    };
</script>
```

- JavaScript
- jQuery
- Ajax

# 문서 객체 모델

## ❖ 문서 객체 가져오기(1)

- getElementById(id) : 태그의 id속성이 id 매개변수와 일치하는 문서 객체를 가져옴
- getElementById(id)는 한 번에 한 가지 문서 객체만 가져옴

```
1  <!DOCTYPE html>
2 ▼ <html>
3 ▼ <head>
4 ▼   <script>
5 ▼     window.onload = function () {
6         // 문서 객체를 가져옵니다.
7         var header1 = document.getElementById('header-1');
8         var header2 = document.getElementById('header-2');
9
10        // 문서 객체의 속성을 변경합니다.
11        header1.innerHTML = 'with getElementById(header-1)';
12        header2.innerHTML = 'with getElementById(header-2)';
13    };
14  </script>
15 </head>
16 ▼ <body>
17    <h1 id="header-1">Header</h1>
18    <h1 id="header-2">Header</h1>
19 </body>
20 </html>
```

**with getElementById(header-1)**  
**with getElementById(header-2)**

# 문서 객체 모델

## ❖ 문서 객체 가져오기(2)

- getElementByName(name): 태그의 name속성이 name매개변수와 일치하는 문서 객체를 배열로 가져옴
- getElementByTagName(tagName) : tagName 매개변수와 일치하는 문서 객체를 배열로 가져옴

```
1  <!DOCTYPE html>
2 ▼ <html>
3 ▼ <head>
4      <title>Index</title>
5 ▼      <script>
6 ▼          window.onload = function () {
7              // 문서 객체를 가져옵니다.
8              var headers = document.getElementsByTagName('h1');
9
10             //for in문을 사용하면 문서객체 [이외의 속성에도 접근함으로 사용하지 말것
11            for (var i = 0; i < headers.length; i++) {
12                // 문서 객체의 속성을 변경합니다.
13                headers[i].innerHTML = 'with getElementsByTagName(' + i + ')';
14            }
15        };
16    </script>
17  </head>
18 ▼ <body>
19      <h1>Header</h1>
20      <h1>Header</h1>
21  </body>
22  </html>
```

**with getElementsByTagName(0)**

**with getElementsByTagName(1)**

# 문서 객체 모델

## ❖ 문서 객체 가져오기(3)

- querySelector(선택자): 선택자로 가장 처음 선택되는 문서 객체를 가져옴
- querySelectorAll(선택자) : 선택자를 통해 선택되는 문서객체를 배열로 가져옴

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>DOM Basic</title>
5  <script>
6  window.onload = function () {
7      // 문서 객체를 가져옵니다.
8      var header1 = document.querySelector('#header-1');
9      var header2 = document.querySelector('#header-2');
10     var navs = document.querySelectorAll('.navItem');
11
12     // 문서 객체의 속성을 변경합니다.
13     header1.innerHTML = 'with getElementById(header-1)';
14     header2.innerHTML = 'with getElementById(header-2)';
15
16     //for in문을 사용하면 문서객체 이외의 속성에도 접근함으로 사용하지 말것
17     for (var i = 0; i < navs.length; i++) {
18         // 문서 객체의 속성을 변경합니다.
19         navs[i].innerHTML = 'querySelectorAll-navItem(' + i + ')';
20     }
21 };
22 </script>
23 </head>
24 <body>
25     <h1 id="header-1">Header-1</h1>
26     <h1 id="header-2">Header-2</h1>
27     <ul>
28         <li class='navItem'></li>
29         <li class='navItem'></li>
30         <li class='navItem'></li>
31     </ul>
32 </body>
33 </html>
```

**with getElementById(header-1)**

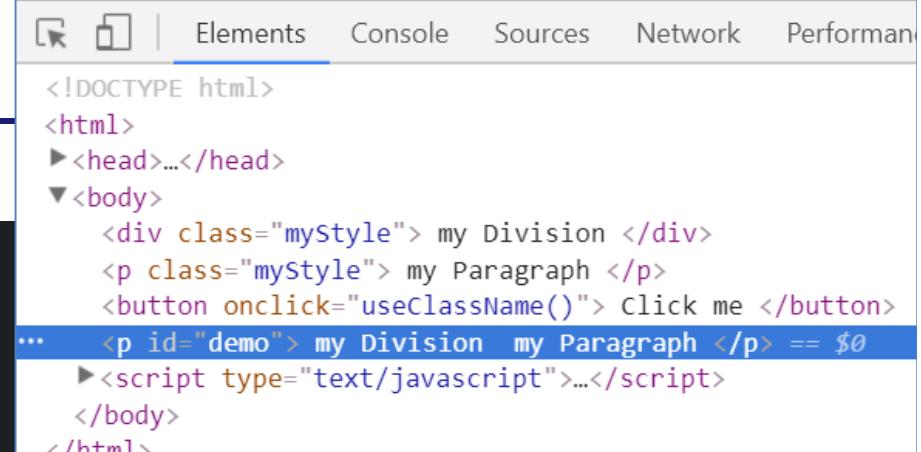
**with getElementById(header-2)**

- querySelectorAll.navItem(0)
- querySelectorAll.navItem(1)
- querySelectorAll.navItem(2)

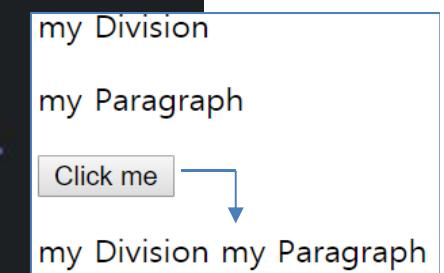
# 문서 객체 모델

## ❖ 문서 객체 가져오기(4)

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>DOM Basic</title>
5  </head>
6  <body>
7      <div class="myStyle"> my Division </div>
8      <p class="myStyle"> my Paragraph </p>
9
10     <button onclick="useClassName()"> Click me </button>
11
12     <p id="demo"></p>
13
14     <script type="text/javascript">
15         function useClassName (){
16             var tags = document.getElementsByClassName( "myStyle" );
17             var text = '';
18             text += tags[0].innerHTML;
19             text += tags[1].innerHTML;
20
21             //아이디가 demo인 문서객체의 innerHTML로 text를 넣는다.
22             document.getElementById("demo").innerHTML = text;
23         }
24     </script>
25 </body>
26 </html>
```



```
<!DOCTYPE html>
<html>
    <head>...</head>
    <body>
        <div class="myStyle"> my Division </div>
        <p class="myStyle"> my Paragraph </p>
        <button onclick="useClassName()"> Click me </button>
        ... <p id="demo"> my Division my Paragraph </p> == $0
        <script type="text/javascript">...</script>
    </body>
</html>
```



# 문서 객체 모델

## ❖ 문서 객체 가져오기(5)

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>DOM Basic</title>
5  </head>
6  <body>
7      <div class="myStyle"></div>
8      <div class="divStyle"></div>
9
10     <button onclick="getClassName_with_tagName()"> Click me </button>
11     <p id="demo"></p>
12
13     <script type="text/javascript">
14         function getClassName_with_tagName (){
15             // div 태그 문서객체를 배열로 가져옴
16             var tags = document.getElementsByTagName("div");
17
18             // tags[0]의 클래스 이름을 가져옴
19             var className = tags[0].className;
20
21             //아이디가 demo인 문서객체의 innerHTML로 className을 넣는다.
22             document.getElementById("demo").innerHTML = className;
23         }
24     </script>
25
26     </body>
27     </html>
```

```
<body>
    <div class="myStyle"></div>
    <div class="divStyle"></div>
    <button onclick="getClassName_with_tagName()"> Click me </button>
    <p id="demo">myStyle</p> == $0
    <script type="text/javascript">...</script>
```



# 문서 객체 모델

## ❖ 문서 객체 가져오기(6)

```
1  <!DOCTYPE html>
2  ▼ <html>
3  ▼ <head>
4      <title>DOM Basic</title>
5  </head>
6  ▼ <body>
7      <P class="myStyle"></P>
8      <DIV class="test myStyle"></DIV>
9      <DIV class="myStyle test example"></DIV>
10
11     <DIV id="mydiv"></DIV>
12
13     <button onclick="total_of_sameClass()"> Click me </button>
14     <p id="demo"></p>
15
16    ▼ <script type="text/javascript">
17        function total_of_sameClass (){
18            var tags = document.getElementsByClassName("myStyle");
19            var total = tags.length;
20
21            document.getElementById ("demo").innerHTML = total;
22        }
23    </script>
24
25  </body>
26  </html>
```

```
<body>
<p class="myStyle"></p>
<div class="test myStyle"></div>
<div class="myStyle test example"></div>
<div id="mydiv"></div>
<button onclick="total_of_sameClass()"> Click me </button>
<p id="demo">3</p> == $0
▶<script type="text/javascript">...</script>
</body>
```

Click me

3

# 문서 객체 모델

## ❖ 문서 객체 속성 조작

- setAttribute(속성명, 값) : 속성값 설정

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>DOM</title>
6  <style>
7      #box{
8          width: 100px;
9          height: 100px;
10         border: 3px solid black;
11     }
12     .box{
13         background: orange;
14     }
15  </style>
16 </head>
17 <body>
18
19     <div id='box' onclick='changeColor()'>Hello box</div>
20
21     <script>
22         function changeColor(){
23             // 문서객체를 가져온다.
24             var box = document.getElementById('box');
25
26             // box 문서객체에 class='box'를 설정
27             box.setAttribute('class', 'box');
28         }
29     </script>
30 </body>
31 </html>
```

▼<body>  
  <div id="box" onclick="changeColor()" class="box">Hello box</div>  
  ►<script>...</script>  
  </body>

Hello box

Hello box

# 문서 객체 모델

## ❖ 문서 객체 속성 조작

- `getAttribute(속성명)` : 속성 읽기

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>DOM</title>
6  <style>
7      #box{
8          width: 100px;
9          height: 100px;
10         border: 3px solid black;
11     }
12     .box{
13         background: orange;
14     }
15 </style>
16 </head>
17 <body>
18
19     <div id='box' onclick='changeColor()'>Hello box</div>
20
21 <script>
22     function changeColor(){
23         // 문서객체를 가져온다.
24         var box = document.getElementById('box');
25
26         // box 문서객체에 class='box'를 설정
27         box.setAttribute('class', 'box');
28
29         // 'class' 속성값 읽기
30         var getAttr = box.getAttribute('class');
31         alert('class=' + getAttr);
32     }
33 </script>
34 </body>
35 </html>
```

class=box

확인

# 문서 객체 모델

## ❖ 문서 객체 속성 조작

- removeAttribute(속성명) : 속성 제거

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>DOM</title>
6  <style>
7      #box{
8          width: 100px;
9          height: 100px;
10         border: 3px solid black;
11     }
12     .box{
13         background: orange;
14     }
15 </style>
16 </head>
17 <body>
18
19     <div id='box' class='box' onclick='changeColor()>Hello box</div>
20
21 <script>
22     function changeColor(){
23         // 문서객체를 가져온다.
24         var box = document.getElementById('box');
25
26         // box 문서객체의 class='box'를 제거
27         box.removeAttribute('class');
28     }
29 </script>
30 </body>
31 </html>
```

# 문서 객체 모델

## ❖ 문서 객체 스타일 조작

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>DOM Basic</title>
5  <script>
6      window.onload = function () {
7          // 문서 객체를 가져옵니다.
8          var header = document.getElementById('header');
9
10         // 문서 객체의 스타일을 바꿔줍니다.
11         header.style.border = '2px solid black';
12         header.style.color = 'orange';
13         header.style.fontFamily = 'helvetica';
14     };
15     </script>
16 </head>
17 <body>
18     <h1 id="header">Header</h1>
19 </body>
20 </html>
```

Header

# 문서 객체 모델

## ❖ 문서 객체 제거

- removeChild(child) : 문서 객체의 자식 노드를 삭제

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>DOM Basic</title>
5  <script>
6      window.onload = function () {
7          // 문서 객체를 가져옵니다.
8          var willRemove = document.getElementById('will-remove');
9
10         // 문서 객체를 제거합니다.
11         document.body.removeChild(willRemove);
12     };
13 </script>
14 </head>
15 <body>
16     <h1 id="will-remove">Header</h1>
17 </body>
18 </html>
```

▶ <head>...</head>  
· <body></body> == \$0  
</html>

# 문서 객체 모델

## ❖ 문서 객체를 사용한 시계

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>DOM Basic</title>
5  <script>
6      window.onload = function () {
7          // 문서객체를 가져온다.
8          var clock = document.getElementById('clock');
9
10         // 1초마다 함수를 실행.
11         setInterval(function () {
12             clock.innerHTML = new Date().toString();
13         }, 1000);
14     };
15 </script>
16 </head>
17 <body>
18     <h1 id="clock"></h1>
19 </body>
20 </html>
```

Sun Sep 24 2017 08:06:04 GMT+0900 (대한민국 표준시)

# 이벤트 처리

## ❖ 고전 이벤트 모델 – 이벤트 연결

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script>
5      window.onload = function () {
6          // 문서객체를 가져온다.
7          var header = document.getElementById('header');
8
9          // 문서객체에 이벤트 연결
10         header.onclick = function () {
11             alert('클릭이벤트가 발생했습니다.');
12         };
13     };
14 </script>
15 </head>
16 <body>
17     <h1 id="header">이곳을 Click하세요</h1>
18 </body>
19 </html>
```

이곳을 Click하세요

클릭이벤트가 발생했습니다.

# 이벤트 처리

## ❖ 고전 이벤트 모델 – 이벤트 제거

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script>
5  window.onload = function () {
6      // 문서객체를 가져옴
7      var header = document.getElementById('header');
8
9      // 이벤트를 연결합니다.
10     header.onclick = function () {
11         alert('클릭이벤트가 발생했습니다. \n 이벤트를 제거합니다.');
12         // 이벤트를 제거합니다.
13         header.onclick = null;
14     };
15 }
16 </script>
17 </head>
18 <body>
19     <h1 id="header">이곳을 Click하세요.</h1>
20 </body>
21 </html>
```

이곳을 Click하세요

클릭이벤트가 발생했습니다.  
이벤트를 제거합니다.

# 이벤트 처리

## ❖ 표준 이벤트 모델

```
1  <!DOCTYPE html>
2 ▼ <html>
3 ▼ <head>
4 ▼   <script>
5 ▼     window.onload = function () {
6       // 문서객체를 가져온다.
7       var header = document.getElementById('my-header');
8
9       // 표준이벤트 모델 방식으로 이벤트 연결
10      header.addEventListener('click', function () {
11        this.innerHTML += '+';
12      });
13    };
14  </script>
15 </head>
16 ▼ <body>
17   <h1 id="my-header">이곳을 Click하세요. </h1>
18 </body>
19 </html>
```

이곳을 Click하세요.

이곳을 Click하세요. + + +

# 예외 처리

## ❖ 예외처리

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script>
5  try {
6      willExcept.byebye();
7  } catch (exception) {
8      alert('예외가 발생했습니다.');
9  } finally {
10     alert('예외 발생 가능 부분을 통과했습니다.');
11 }
12 </script>
13 </head>
14 <body>
15
16 </body>
17 </html>
```

예외가 발생했습니다.

확인

예외 발생 가능 부분을 통과했습니다.

확인