

UI 프론트엔드 프로그래밍 과정

# JavaScript

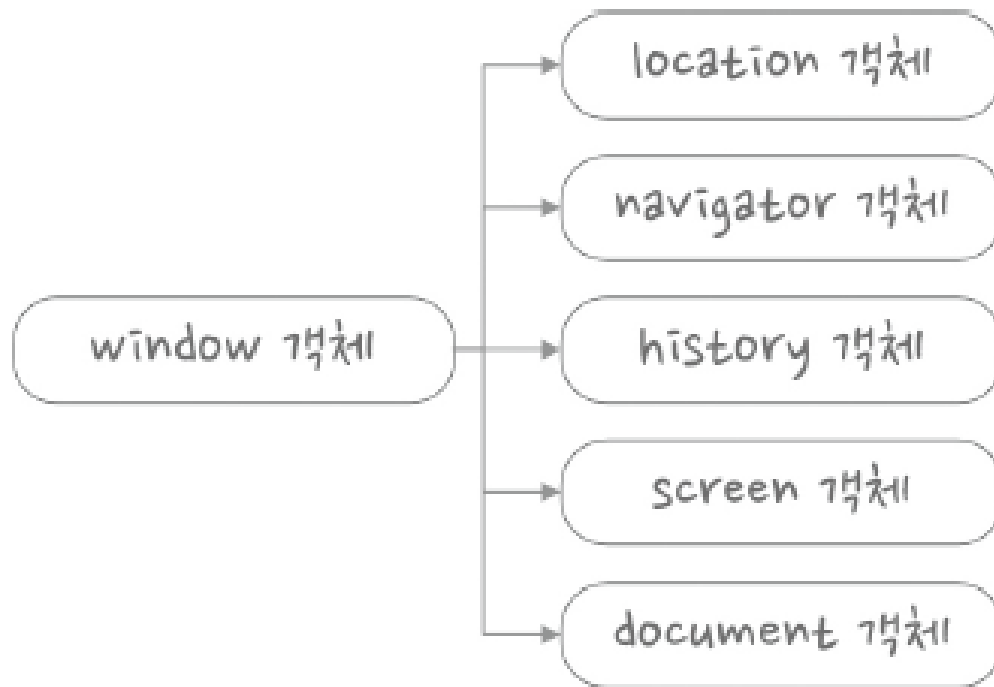
## 윈도우객체와 문서객체

# JavaScript 브라우저 객체 모델

## 9. 브라우저 객체 모델

### ◆ 브라우저 객체 모델

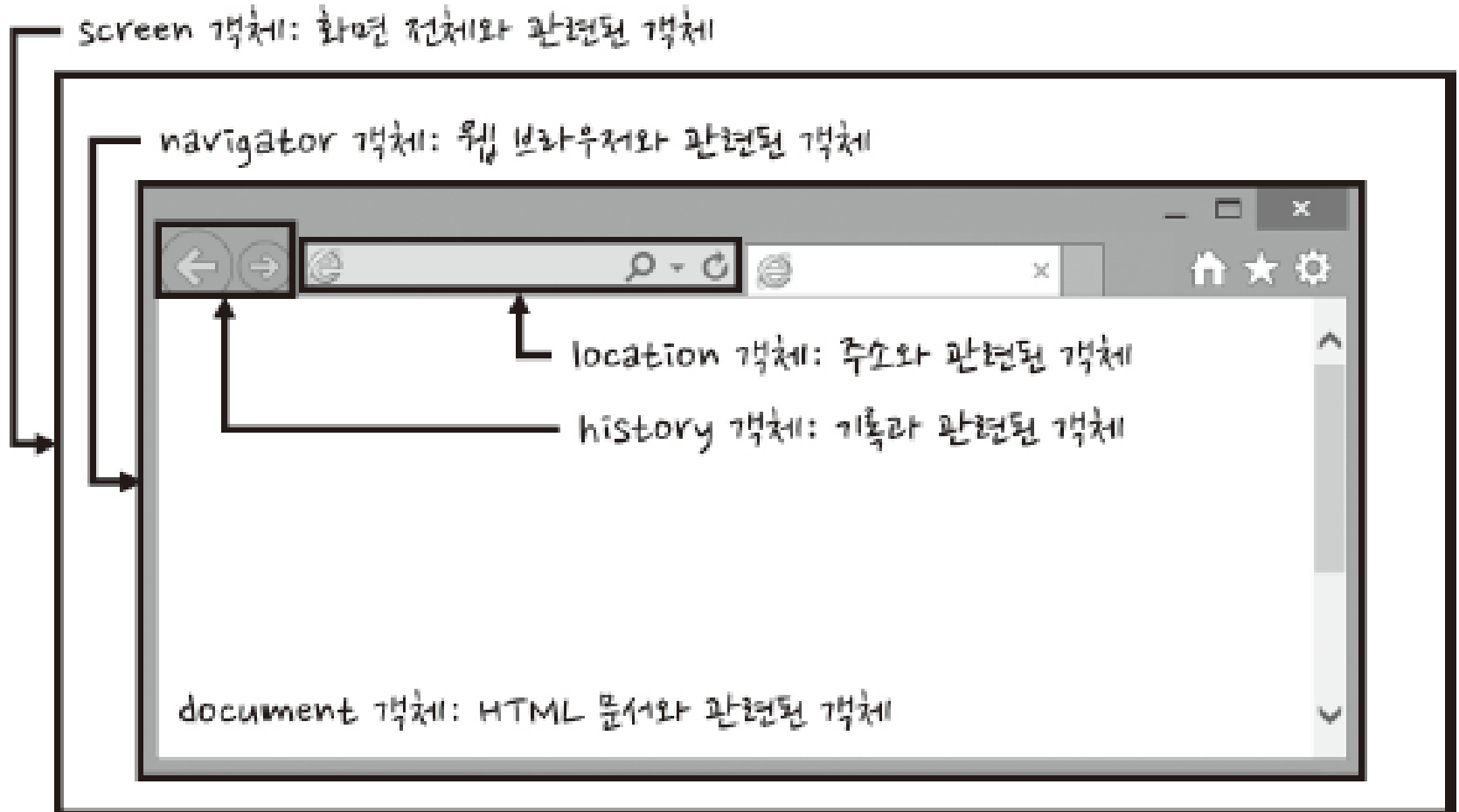
- 웹 브라우저와 관련된 객체의 집합
- 대표적인 브라우저 객체 모델
  - window, location, navigator, history, screen, document 객체



## 9. 브라우저 객체 모델

### ◆ 브라우저 객체 모델

- 브라우저 관련 객체



# 9.1 window 객체

## ◆ window 객체

- window 객체의 속성과 메서드 출력
- 코드를 실행
  - 익스플로러, 크롬에서는 확인 할 수 없음
  - 파이어폭스 4나 오페라 브라우저를 사용하면 결과 확인 가능

---

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

---

# 9.1 window 객체

---

## ◆ window 객체

- window 객체의 속성
- 많은 속성이 있음
- 일부 익스플로러에서는 실행되지 않음

# 9.1 window 객체

---

## ◆ window 객체

- window 객체는 브라우저 기반 자바스크립트의 최상위 객체
- alert( ), prompt ( ) 함수 모두 window 객체의 메서드

## 9.2 새로운 window 객체 생성

### ◆ 새로운 window 객체 생성

#### ■ window 객체 생성

- open ( ) 메서드는 네 개의 매개변수가 있음
- 입력해도 되고 입력하지 않아도 되는 매개변수를 옵션이라고 함
- open ( ) 메서드의 모든 매개변수는 옵션

메서드 이름	설명
open(URL, name, features, replace)	새로운 window 객체를 생성합니다.

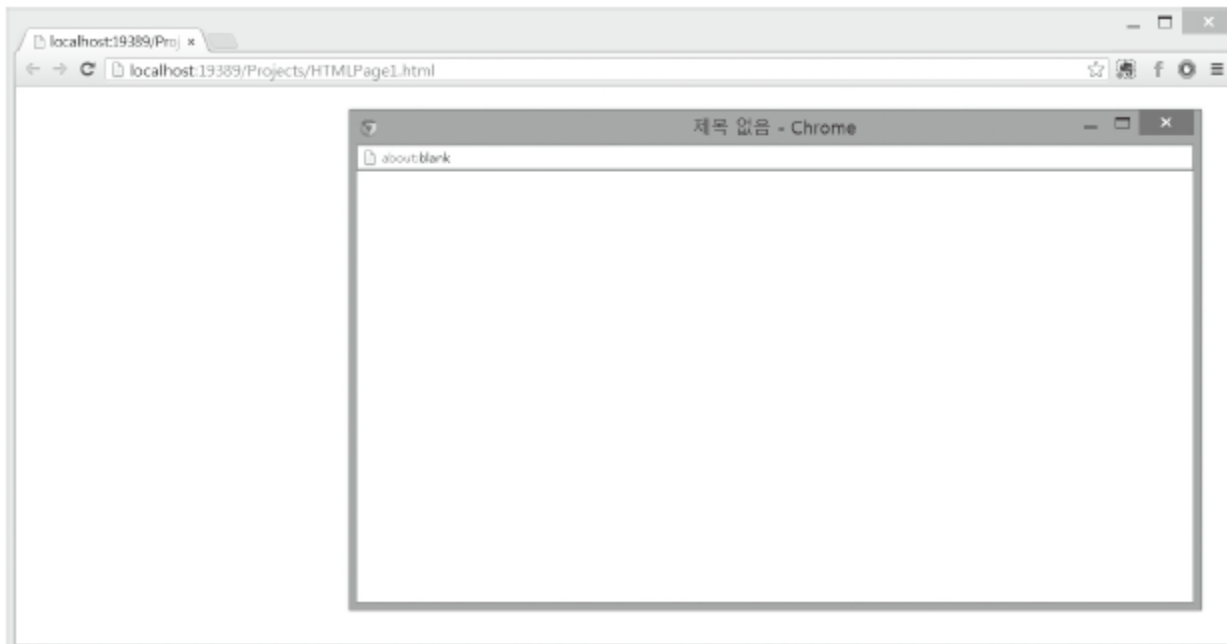


## 9.2 새로운 window 객체 생성

### ◆ 새로운 window 객체 생성

- open( ) 메서드의 옵션을 사용하지 않은 예
- 옵션을 사용한 open( ) 메서드와 차이가 없음

```
<script>  
    window.open();  
</script>
```

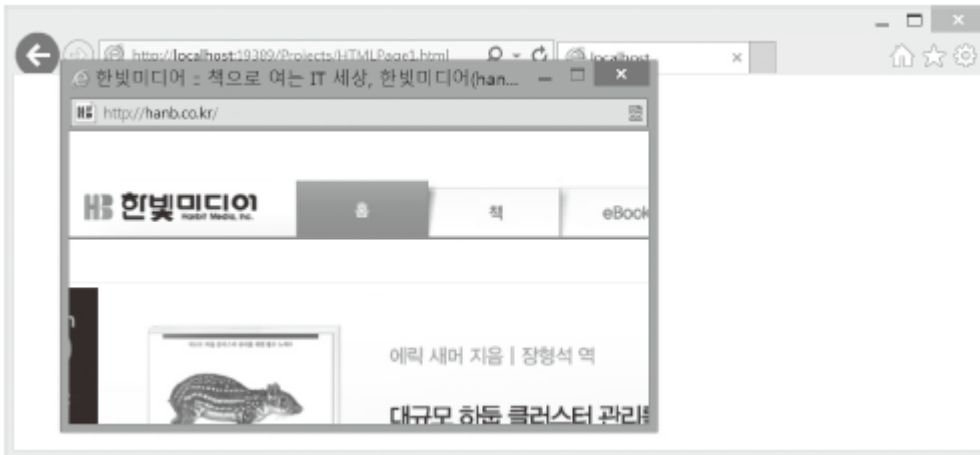


## 9.2 새로운 window 객체 생성

### ◆ 새로운 window 객체 생성

- open( ) 메서드의 옵션을 사용한 예

```
<script>  
    window.open('http://hanb.co.kr', 'child', 'width=600, height=300', true);  
</script>
```



## 9.2 새로운 window 객체 생성

### ◆ 새로운 window 객체 생성

- open( ) 메서드의 첫 번째 매개변수  
→ 열고자 하는 HTML 페이지의 URL
- open( ) 메서드의 두 번째 매개변수  
→ 윈도우 간 통신하는데 사용하는 윈도우 이름
- open( ) 메서드의 세 번째 매개변수  
→ 윈도우 출력 모양을 지정하는 옵션

옵션 이름	설명	입력할 수 있는 값
height	새 윈도우의 높이	픽셀 값
width	새 윈도우의 너비	픽셀 값
location	주소 입력창의 유무	yes, no, 1, 0
menubar	메뉴의 유무	yes, no, 1, 0
resizable	화면 크기 조절 가능 여부	yes, no, 1, 0
status	상태 표시줄의 유무	yes, no, 1, 0
toolbar	상태 표시줄의 유무	yes, no, 1, 0

## 9.2 새로운 window 객체 생성

### ◆ 새로운 window 객체 생성

- open( ) 메서드의 속성과 메서드

---

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

    // 출력합니다.
    child.document.write('<h1>From Parent Window</h1>');
</script>
```

---

## 9.3 window 객체의 기본 메서드

### ◆ window 객체의 기본 메서드

- window 객체는 자신의 형태와 위치를 변경할 수 있도록 메서드 제공

메서드 이름	설명
moveBy(x, y)	윈도의 위치를 상대적으로 이동합니다.
moveTo(x, y)	윈도의 위치를 절대적으로 이동합니다.
resizeBy(x, y)	윈도의 크기를 상대적으로 지정합니다.
resizeTo(x, y)	윈도의 크기를 절대적으로 지정합니다.
scrollBy(x, y)	윈도 스크롤의 위치를 상대적으로 이동합니다.
scrollTo(x, y)	윈도 스크롤의 위치를 절대적으로 이동합니다.
focus( )	윈도에 초점을 맞춥니다.
blur( )	윈도에 초점을 제거합니다.
close( )	윈도를 닫습니다.

## 9.3 window 객체의 기본 메서드

### ◆ window 객체의 기본 메서드

- 상대 이동과 절대 이동
- ○○By ( ) 형태의 메서드 : 현재의 윈도우를 기준으로 상대적으로 속성을 변화
- ○○To ( ) 형태의 메서드 : 절대적인 기준으로 속성을 변화

---

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

    child.moveTo(0, 0);

    // 1초마다 함수를 실행합니다.
    setInterval(function () {
        child.moveBy(10, 10);
    }, 1000);
</script>
```

---

## 9.4 screen 객체

### ◆ screen 객체

- 웹 브라우저의 화면이 아닌 운영체제 화면의 속성을 가지는 객체
- screen 객체의 속성

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

속성 이름	설명
width	화면의 너비
height	화면의 높이
availWidth	실제 화면에서 사용 가능한 너비
availHeight	실제 화면에서 사용 가능한 높이
colorDepth	사용 가능한 색상 수
pixelDepth	한 픽셀당 비트 수

## 9.5 location 객체

### ◆ location 객체

- 브라우저의 주소 표시줄과 관련된 객체
- location 객체는 프로토콜의 종류, 호스트 이름, 문서 위치 등의 정보가 있음

---

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

---



## 9.5 location 객체

### ◆ location 객체

- location 객체의 속성

속성 이름	설명	예
href	문서의 URL 주소	
host	호스트 이름과 포트 번호	localhost:30763
hostname	호스트 이름	localhost
port	포트 번호	30763
pathname	디렉토리 경로	/Projects/Location.htm
hash	앵커 이름(#~)	#beta
search	요청 매개변수	?param=10
protocol	프로토콜 종류	http:

## 9.6 navigator 객체

### ◆ navigator 객체

- navigator 객체는 웹 페이지를 실행하고 있는 브라우저에 대한 정보가 있음
- navigator 속성

속성 이름	설명
appName	브라우저의 코드명
appName	브라우저의 이름
appVersion	브라우저의 버전
platform	사용중인 운영체제의 시스템 환경
userAgent	브라우저의 전체적인 정보

## 9.7 window 객체의 onload 이벤트 속성

### ◆ onload 이벤트 속성

- 문서 객체 속성 중 'on'으로 시작하는 속성을 이벤트 속성이라 부르고 함수를 할당해야 함
- onload 이벤트 속성

---

```
<script>
    window.onload = function () {

        };
</script>
```

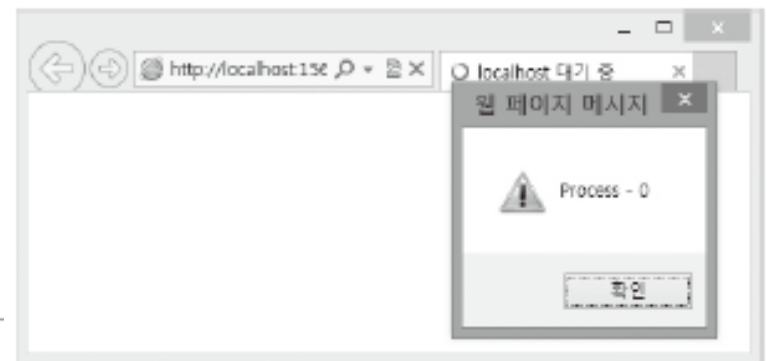
---

## 9.7 window 객체의 onload 이벤트 속성

### ◆ window 객체의 로드 완료

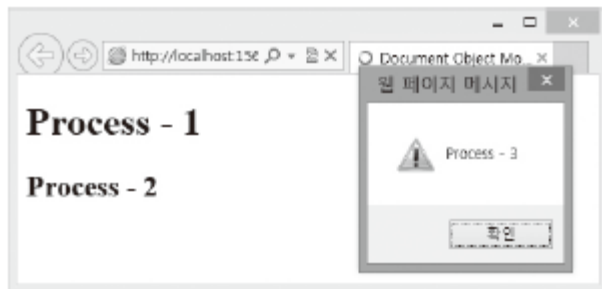
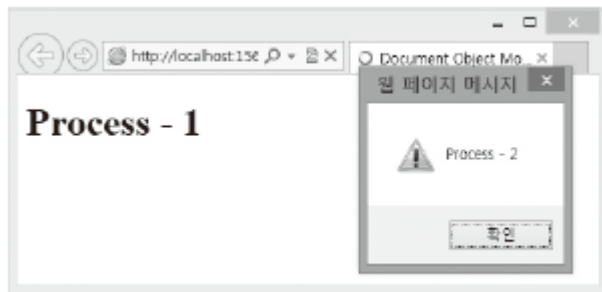
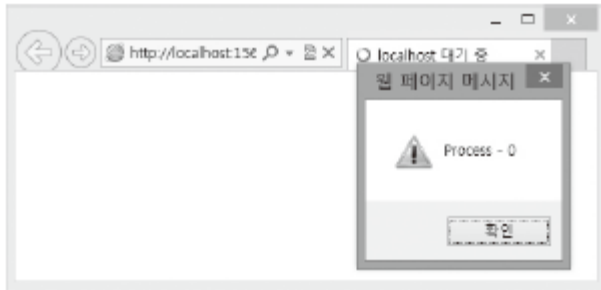
- window 객체 로드가 완료되는 때는?  
→ HTML 페이지에 존재하는 모든 태그가 화면에 올라가는 순간이 로드가 완료되는 순간
- HTML 페이지 생성 순서

```
<!DOCTYPE html>
<html>
<head>
  <title>Document Object Model</title>
  <script>alert('Process - 0')</script>
</head>
<body>
  <h1>Process - 1</h1>
  <script>alert('Process - 2')</script>
  <h2>Process - 2</h2>
  <script>alert('Process - 3')</script>
</body>
</html>
```



## 9.7 window 객체의 onload 이벤트 속성

- ◆ window 객체의 로드 완료
  - HTML 페이지 실행 출력 순서

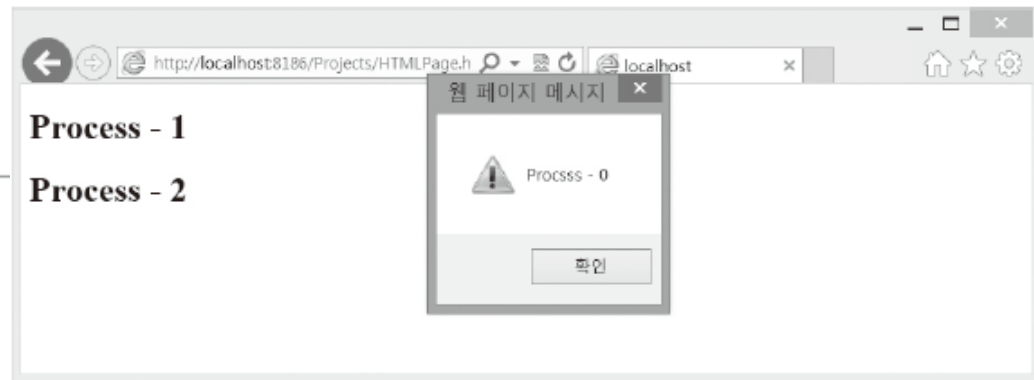


## 9.7 window 객체의 onload 이벤트 속성

### ◆ window 객체의 로드 완료

- onload 이벤트 속성으로 페이지 생성 및 출력 결과

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



# JavaScript 문서 객체 모델

# 10. 문서 객체 모델

---

## ◆ 문서 객체 모델

- 넓은 의미로 웹 브라우저가 HTML 페이지를 인식하는 방식
- 좁은 의미로 document 객체와 관련된 객체의 집합
- 문서 객체 모델을 사용하면 HTML 페이지에 태그를 추가,수정, 제거할 수 있음



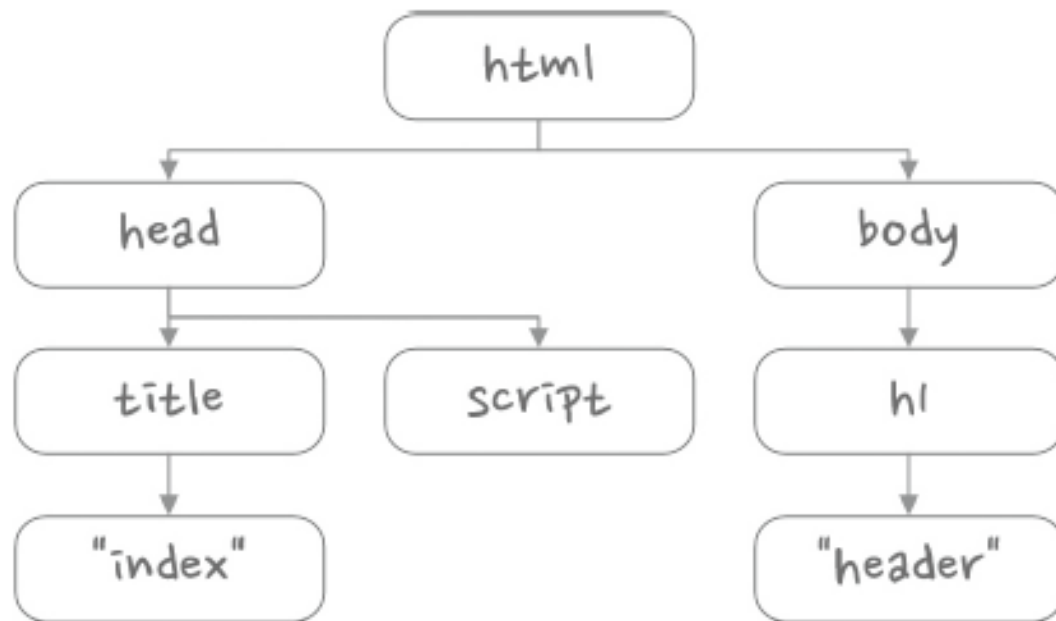
# 10.1 문서 객체 모델과 관련된 용어 정리

## ◆ 태그

- HTML 페이지에 존재하는 html이나 body 태그를 '태그'라고 부름

## ◆ 문서 객체

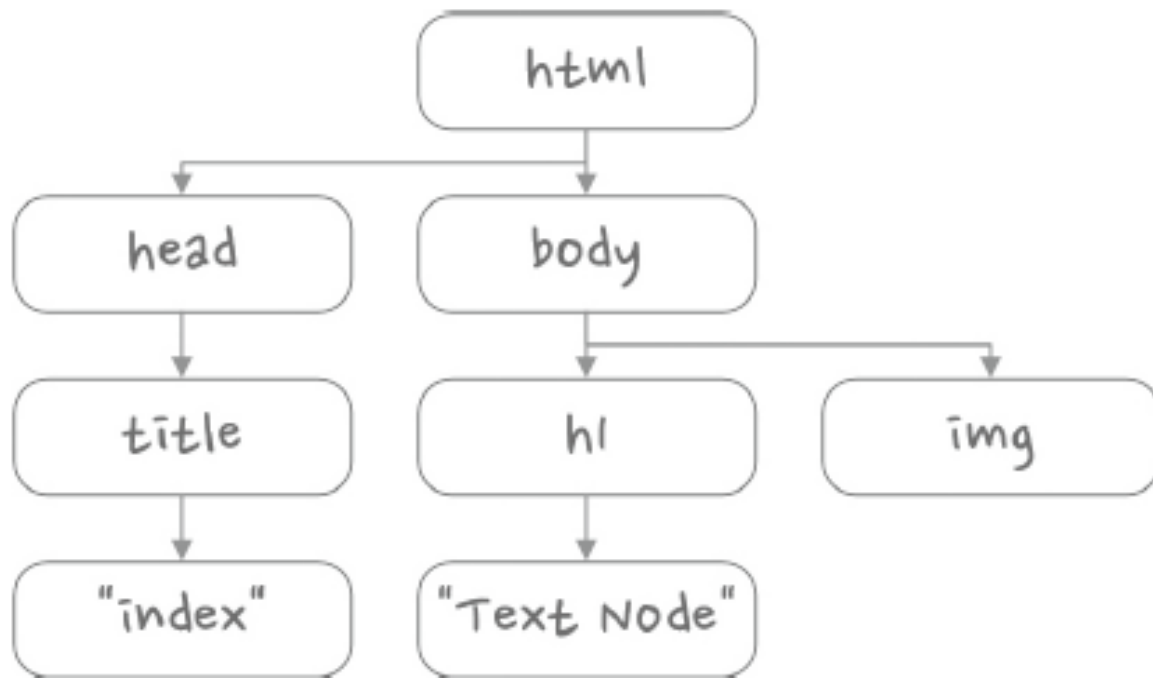
- html이나 body 태그를 자바스크립트에서 이용할 수 있는 객체로 만들면 문서 객체임



# 10.1 문서 객체 모델과 관련된 용어 정리

## ◆ 노드

- 각 요소를 노드라 함
- 요소 노드 : HTML 태그를 의미
- 텍스트 노드 : 요소 노드 안에 들어 있는 글자를 의미



# 10.1 문서 객체 모델과 관련된 용어 정리

---

## ◆ 문서 객체 생성

- 정적으로 문서 객체를 생성 : 처음 HTML 페이지에 적혀 있는 태그들을 읽으며 생성
- 동적으로 문서 객체를 생성 : 자바스크립트로 원래 HTML 페이지에는 없던 문서 객체를 생성

## 10.2 문서 객체 만들기(1)

### ◆ 문서 객체 종류

- 텍스트 노드를 갖는 문서 객체
- 텍스트 노드를 갖지 않는 문서 객체

### ◆ 텍스트 노드를 갖는 문서 객체

- body 태그 구성

---

<body>

</body>

---

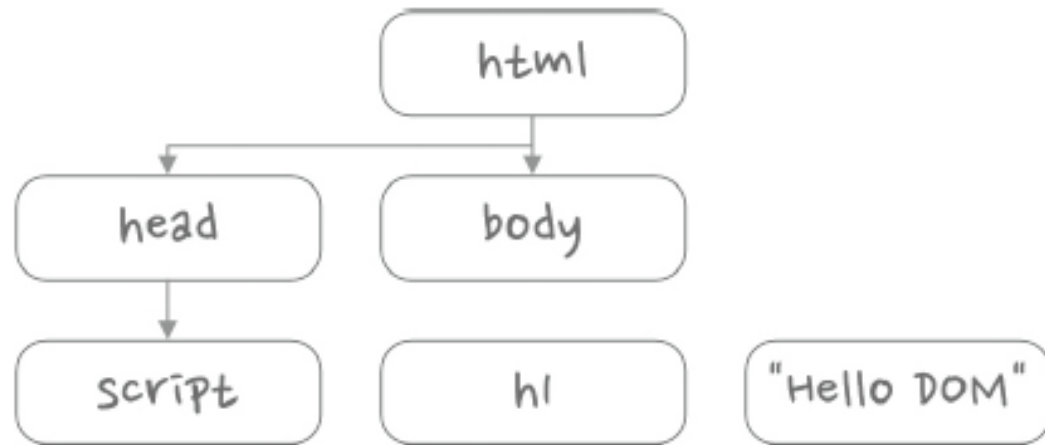
## 10.2 문서 객체 만들기(1)

### ◆ 텍스트 노드를 갖는 문서 객체

- 요소 노드와 텍스트 노드 생성 후 텍스트 노드를 요소 노드에 붙여 줌

메서드 이름	설명
<code>createElement(tagName)</code>	요소 노드를 생성합니다.
<code>createTextNode(text)</code>	텍스트 노드를 생성합니다.

- 현재 상황



## 10.2 문서 객체 만들기(1)

### ◆ 텍스트 노드를 갖는 문서 객체

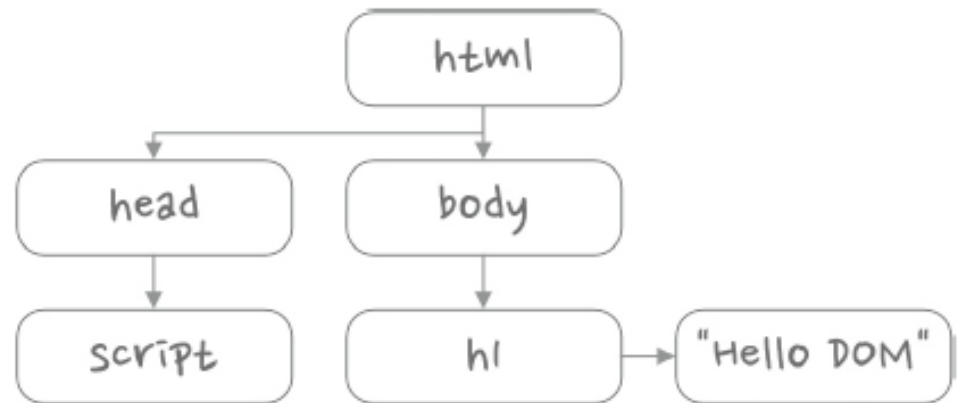
- 화면에 문서를 출력하려면 생성한 문서 객체를 body 문서 객체에 연결
- 노드와 노드를 연결할 때 아래 메서드 사용

메서드 이름	설명
appendChild(node)	객체에 노드를 연결합니다.

## 10.2 문서 객체 만들기(1)

### ◆ 텍스트 노드를 갖는 문서 객체

```
<!DOCTYPE html>
<html>
<head>
  <title>INDEX</title>
  <script>
    window.onload = function () {
      // 변수를 선언합니다.
      var header = document.createElement('h1');
      var textNode = document.createTextNode('Hello DOM');
      // 노드를 연결합니다.
      header.appendChild(textNode);
      document.body.appendChild(header);
    };
  </script>
</head>
<body>
</body>
</html>
```



## 10.3 문서 객체 만들기(2)

### ◆ 텍스트 노드를 갖는 않는 문서 객체

- 텍스트 노드를 갖지 않는 대표적인 HTML 태그는 img
- img 태그는 텍스트 노드 대신에 많은 속성이 있음
- img 태그 생성 후 body 문서 객체에 연결

---

```
<script>
    window.onload = function () {
        // 변수를 선언합니다.
        var img = document.createElement('img');
        // 노드를 연결합니다.
        document.body.appendChild(img);
    };
</script>
```

---



## 10.3 문서 객체 만들기(2)

### ◆ 텍스트 노드를 갖는 애플리케이션 문서 객체

- img 태그에 이미지를 넣으려면 src 속성 지정
- 코드를 실행하면 이미지 출력

---

```
<script>
    window.onload = function () {
        // 변수를 선언합니다.
        var img = document.createElement('img');
        img.src = 'Penguins.jpg';
        img.width = 500;
        img.height = 350;

        // 노드를 연결합니다.
        document.body.appendChild(img);
    };
</script>
```

---

## 10.3 문서 객체 만들기(2)

### ◆ 텍스트 노드를 갖는 않는 문서 객체

- 크롬 요소 검사

```
▼<body>  
    
</body>
```

- 문서 객체의 속성 메서드

메서드 이름	설명
setAttribute(name, value)	객체의 속성을 지정합니다.
getAttribute(name)	객체의 속성을 가져옵니다.

## 10.3 문서 객체 만들기(2)

### ◆ 텍스트 노드를 갖는 애플리케이션 문서 객체

- 문서 객체의 속성 지정

```
<script>
  window.onload = function () {
    // 변수를 선언합니다.
    var img = document.createElement('img');
    img.setAttribute('src', 'Penguins.jpg');
    img.setAttribute('width', 500);
    img.setAttribute('height', 350);

    // setAttribute() 메서드를 사용하지 않으면 불가능합니다.
    img.setAttribute('data-property', 350);

    // 노드를 연결합니다.
    document.body.appendChild(img);
  };
</script>
```

- 크롬 요소 검사

```
▼ <body>
  
</body>
```

## 10.4 문서 객체 만들기(3)

### ◆ innerHTML

- 태그의 내부를 의미하는 속성



```
<script>
  window.onload = function () {
    // 변수를 선언합니다.
    var output = '';

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

## 10.5 문서 객체 가져오기(1)

### ◆ HTML 태그를 자바스크립트로 가져오는 방법

- document 객체가 가지는 표 10-4 메서드를 사용
- document 객체의 getElementById ( ) 메서드는 id 속성을 갖는 태그만 가져올 수 있으므로 id 속성을 입력

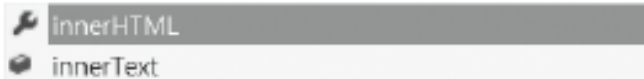
메서드 이름	설명
getElementById(id)	태그의 id 속성이 id와 일치하는 문서 객체를 가져옵니다.

## 10.5 문서 객체 가져오기(1)

### ◆ HTML 태그를 자바스크립트로 가져오는 방법

- header1과 header2는 문서 객체이므로 뒤에 점을 찍어주면 문서 객체의 속성과 메서드를 살펴볼 수 있음

```
window.onload = function () {  
    // 문서 객체를 가져옵니다.  
    var header1 = document.getElementById('header-1');  
    var header2 = document.getElementById('header-2');  
  
    header1.inn  
};
```



## 10.5 문서 객체 가져오기(1)

### ◆ HTML 태그를 자바스크립트로 가져오는 방법

- innerHTML 속성 변경

---

```
<script>
  window.onload = function () {
    // 문서 객체를 가져옵니다.
    var header1 = document.getElementById('header-1');
    var header2 = document.getElementById('header-2');

    // 문서 객체의 속성을 변경합니다.
    header1.innerHTML = 'with getElementById()';
    header2.innerHTML = 'with getElementById()';
  };
</script>
```

---

- 출력

**with getElementById()**  
**with getElementById()**

## 10.6 문서 객체 가져오기(2)

### ◆ 여러 개의 문서 객체 가져오는 방법

- document 객체의 getElementById ( ) 메서드는 한 번에 한 가지 문서 객체만 가져올 수 있음
- 아래 메서드를 이용해서 여러 개의 객체를 가져올 수 있음

메서드 이름	설명
getElementsByName(name)	태그의 name 속성이 name과 일치하는 문서 객체를 배열로 가져옵니다.
getElementsByTagName(tagName)	tagName과 일치하는 문서 객체를 배열로 가져옵니다.



## 10.6 문서 객체 가져오기(2)

### ◆ 여러 개의 문서 객체 가져오는 방법

- 메서드 구현

---

```
<!DOCTYPE html>
<html>
<head>
  <title>Index</title>
  <script>
    window.onload = function () {
      // 문서 객체를 가져옵니다.
      var headers = document.getElementsByTagName('h1');
    };
  </script>
</head>
<body>
  <h1>Header</h1>
  <h1>Header</h1>
</body>
</html>
```

---

## 10.6 문서 객체 가져오기(2)

### ◆ 여러 개의 문서 객체 가져오는 방법

- 문서 객체 배열의 사용

---

```
<script>
    window.onload = function () {
        // 문서 객체를 가져옵니다.
        var headers = document.getElementsByTagName('h1');

        headers[0].innerHTML = 'with getElementsByTagName()';
        headers[1].innerHTML = 'with getElementsByTagName()';
    };
</script>
```

---

## 10.6 문서 객체 가져오기(2)

### ◆ 여러 개의 문서 객체 가져오는 방법

- `getElementsByTagName()` 메서드

---

```
<script>
    window.onload = function () {
        // 문서 객체를 가져옵니다.
        var headers = document.getElementsByTagName('h1');
        for (var i = 0; i < headers.length; i++) {
            // 문서 객체의 속성을 변경합니다.
            headers[i].innerHTML = 'with getElementsByTagName()';
        }
    };
</script>
```

---

- 출력

**with getElementById()**

**with getElementById()**

## 10.7 문서 객체 가져오기(3)

### ◆ HTML 5에서 추가된 메서드

메서드 이름	설명
<code>document.querySelector(선택자)</code>	선택자로 가장 처음 선택되는 문서 객체를 가져옵니다.
<code>document.querySelectorAll(선택자)</code>	선택자로 선택되는 문서 객체를 배열로 가져옵니다.

## 10.7 문서 객체 가져오기(3)

### ◆ HTML 5에서 추가된 메서드

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Basic</title>
  <script>
    window.onload = function () {
      // 문서 객체를 가져옵니다.
      var header1 = document.querySelector('#header-1');
      var header2 = document.querySelector('#header-2');

      // 문서 객체의 속성을 변경합니다.
      header1.innerHTML = 'with getElementById()';
      header2.innerHTML = 'with getElementById()';
    };
  </script>
</head>
<body>
  <h1 id="header-1">Header</h1>
  <h1 id="header-2">Header</h1>
</body>
</html>
```

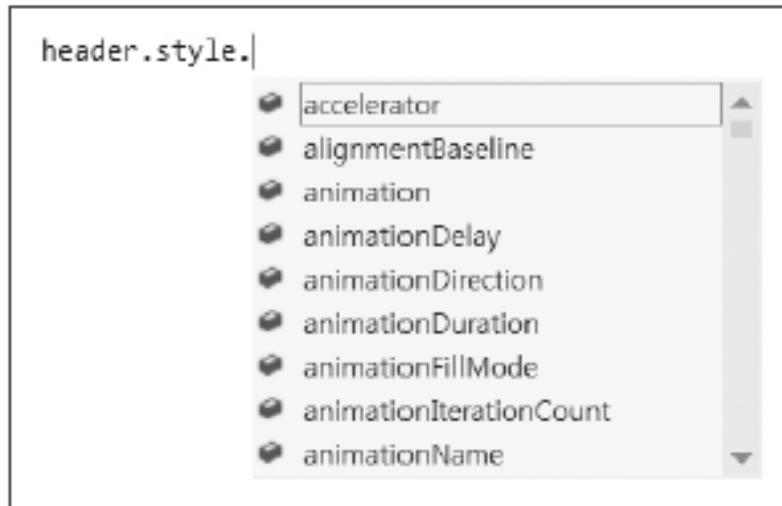
**with getElementById()**

**with getElementById()**

## 10.8 문서 객체의 스타일 조작

### ◆ 문서 객체의 스타일 변경

- style 속성 사용
- getElementById ( ) 메서드로 문서 객체를 가져옴
- style 속성에 있는 border, color, fontFamily 속성을 지정
- CSS에 입력하는 것과 같은 형식으로 입력



## 10.9 문서 객체 제거

### ◆ 문서 객체 제거

- 제거 메서드 사용

메서드 이름	설명
<code>removeChild(child)</code>	문서 객체의 자식 노드를 제거합니다.

- h1 태그 제거

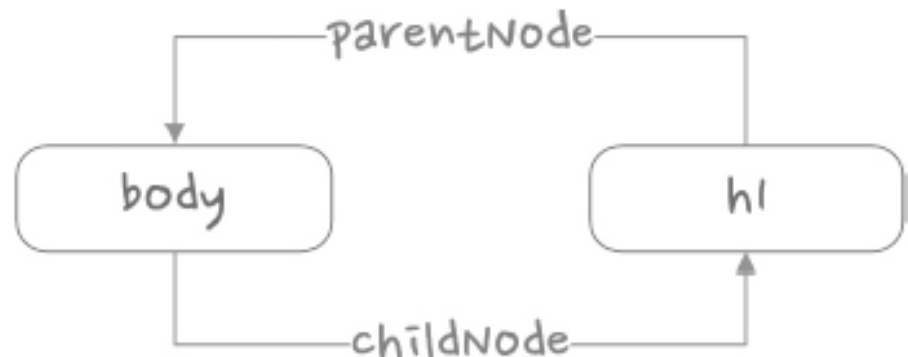
```
<body>  
  <h1 id="will-remove">Header</h1>  
</body>
```

## 10.9 문서 객체 제거

### ◆ 문서 객체 제거

- 문서 객체의 제거

```
<script>  
  window.onload = function () {  
    // 문서 객체를 가져옵니다.  
    var willRemove = document.getElementById('will-remove');  
  
    // 문서 객체를 제거합니다.  
    document.body.removeChild(willRemove);  
  };  
</script>
```





## 10.10 문서 객체를 사용한 시계

### ◆ 시계를 만들어 보며 문서 객체 연습

- body 태그 구성

---

```
<body>
  <h1 id="clock"></h1>
</body>
```

---

- 문서 객체 설정

---

```
<script>
  window.onload = function () {
    // 변수를 선언합니다.
    var clock = document.getElementById('clock');
  };
</script>
```

---

## 10.10 문서 객체를 사용한 시계

### ◆ 시계를 만들어 보며 문서 객체 연습

- 현재 시간 표시

---

```
<script>
  window.onload = function () {
    // 변수를 선언합니다.
    var clock = document.getElementById('clock');

    // 매 1초마다 함수를 실행합니다.
    setInterval(function () {
      clock.innerHTML = new Date().toString();
    }, 1000);
  };
</script>
```

---

- 출력

**Tue Jul 16 20:30:39 UTC+0900 2013**

# JavaScript 이벤트

# 11. 이벤트

---

## ◆ 이벤트란?

- 키보드를 이용해 버튼을 입력하거나 마우스 클릭과 같이 다른 것에 영향을 미치는 것
- 애플리케이션 사용자가 발생시킬 수도 있고 애플리케이션이 스스로 발생시킬 수도 있음
- 자바 스크립트 이벤트 종류

- 마우스 이벤트
- 키보드 이벤트
- HTML 프레임 이벤트
- HTML 입력 양식 이벤트
- 유저 인터페이스 이벤트
- 구조 변화 이벤트
- 터치 이벤트

# 11.1 이벤트 관련 용어 정리

## ◆ 이벤트 연결

- window 객체의 onload 속성에 함수 자료형을 할당하는 것을 "이벤트를 연결한다"고 함
- load를 이벤트 이름 또는 이벤트 타입이라 함
- onload를 이벤트 속성이라고 함
- 이벤트 속성에 할당한 함수를 이벤트 리스너 또는 이벤트 핸들러라 함

---

```
<script>  
    window.onload = function () { };  
</script>
```

---

# 11.1 이벤트 관련 용어 정리

## ◆ 이벤트 용어 연습

- 이벤트 이름, 이벤트 속성, 이벤트 리스너 찾아보기

---

```
<script>
  window.onload = function () {
    // 변수를 선언합니다.
    var header = document.getElementById('header');

    // 이벤트를 연결합니다.
    function whenClick() { alert('CLICK'); }
    header.onclick = whenClick;
  };
</script>
```

---

# 11.1 이벤트 관련 용어 정리

---

## ◆ 이벤트 연결

- 이벤트 모델 : 문서 객체에 이벤트를 연결하는 방법
- 이벤트 모델 분류
  - DOM Level 단계에 따라 두 가지로 분류
  - 분류된 두 가지가 각기 두 가지로 나뉘어 총 네 가지 방법으로 이벤트 연결

## 11.2 고전 이벤트 모델

### ◆ 고전 이벤트 모델

- 자바스크립트에서 문서 객체의 이벤트 속성으로 이벤트를 연결하는 방법
- 이름은 고전이지만 현대에서도 많이 사용

---

```
<body>  
  <h1 id="header">Click</h1>  
</body>
```

---



## 11.2 고전 이벤트 모델

### ◆ 고전 이벤트 모델

- 고전 이벤트 모델을 사용한 이벤트 연결
- `getElementById ( )` 메서드로 문서 객체를 가져오고 `click` 이벤트를 연결

---

```
<script>
    window.onload = function () {
        // 변수를 선언합니다.
        var header = document.getElementById('header');

        // 이벤트를 연결합니다.
        header.onclick = function () {
            alert('클릭');
        };
    };
</script>
```

---


# 11.2 고전 이벤트 모델

## ◆ 고전 이벤트 모델

### ■ 이벤트 속성

그림 11-1 이벤트 속성

```
var header = document.getElementById('header');  
header.on
```



- onchange
- onclick
- oncontextmenu
- oncontrolselect
- oncopy
- oncuechange
- oncut
- ondataavailable
- ondatasetchanged

## 11.2 고전 이벤트 모델

### ◆ 고전 이벤트 모델

- 이벤트 제거

---

```
<script>
  window.onload = function () {
    // 변수를 선언합니다.
    var header = document.getElementById('header');

    // 이벤트를 연결합니다.
    header.onclick = function () {
      alert('클릭');

      // 이벤트를 제거합니다.
      header.onclick = null;
    };
  };
</script>
```

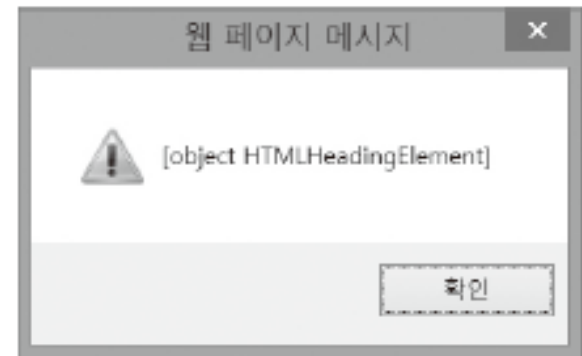
---

## 11.3 이벤트 발생 객체와 이벤트 객체

### ◆ this 키워드

- 이벤트를 발생한 객체를 찾을 수 있음

```
<!DOCTYPE html>
<html>
<head>
  <script>
    window.onload = function () {
      document.getElementById('header').onclick = function () {
        alert(this);
      };
    };
  </script>
</head>
<body>
  <h1 id="header">Click</h1>
</body>
</html>
```



## 11.3 이벤트 발생 객체와 이벤트 객체

### ◆ this 키워드

- this 키워드의 스타일을 바꾸는 것은 이벤트가 발생한 객체의 스타일을 변경하는 것

---

```
<script>
  window.onload = function () {
    document.getElementById('header').onclick = function () {
      this.style.color = 'orange';
      this.style.backgroundColor = 'red';
    };
  };
</script>
```

---

## 11.3 이벤트 발생 객체와 이벤트 객체

### ◆ this 키워드

- 이벤트 객체 : '누가'와 관련된 정보

---

```
<script>
  window.onload = function () {
    document.getElementById('header').onclick = function (e) {
      // 이벤트 객체를 설정합니다.
      var event = e || window.event;

      document.body.innerHTML = '';
      for (var key in event) {
        document.body.innerHTML += '<p>' + key + ': ' + event[key] + '</p>';
      }
    };
  };
</script>
```

---

## 11.4 이벤트 강제 실행

### ◆ 이벤트 강제 실행

- 메서드를 호출하는 것처럼 이벤트 속성을 호출하면 이벤트가 강제로 실행

---

```
header.onclick()
```

---

---

```
<body>
  <button id="button-a">ButtonA</button>
  <button id="button-b">ButtonB</button>
  <h1>Button A - <span id="counter-a">0</span></h1>
  <h1>Button B - <span id="counter-b">0</span></h1>
</body>
```

---

## 11.4 이벤트 강제 실행

### ◆ 이벤트 강제 실행

- 이벤트 연결

---

```
<script>
    window.onload = function () {
        // 문서 객체를 가져옵니다.
        var buttonA = document.getElementById('button-a');
        var buttonB = document.getElementById('button-b');
        var counterA = document.getElementById('counter-a');

        var counterB = document.getElementById('counter-b');

        // 이벤트를 연결합니다.
        buttonA.onclick = function () { };
        buttonB.onclick = function () { };
    };
</script>
```

---



## 11.4 이벤트 강제 실행

### ◆ 이벤트 강제 실행

- 클릭 횟수 증가

---

// 이벤트를 연결합니다.

```
buttonA.onclick = function () {  
    counterA.innerHTML = Number(counterA.innerHTML) + 1;  
};  
buttonB.onclick = function () {  
    counterB.innerHTML = Number(counterB.innerHTML) + 1;  
};
```

---

```
buttonB.onclick = function () {  
    counterB.innerHTML = Number(counterB.innerHTML) + 1;  
    counterA.innerHTML = Number(counterA.innerHTML) + 1;  
};
```

---

## 11.4 이벤트 강제 실행

### ◆ 이벤트 강제 실행

- 간단한 이벤트 강제 실행

---

```
// 이벤트를 연결합니다.  
buttonA.onclick = function () {  
    counterA.innerHTML = Number(counterA.innerHTML) + 1;  
};  
buttonB.onclick = function () {  
    counterB.innerHTML = Number(counterB.innerHTML) + 1;  
    buttonA.onclick();  
};
```

---

ButtonA

ButtonB

**Button A - 25**

**Button B - 9**

# 11.5 인라인 이벤트 모델

## ◆ 인라인 이벤트 모델

- HTML 페이지의 가장 기본적인 이벤트 연결 방법

---

```
<body>  
  <h1>Click</h1>  
</body>
```

---

- h1 태그에 onclick 속성을 줌

---

```
<body>  
  <h1 onclick="">Click</h1>  
</body>
```

---

# 11.5 인라인 이벤트 모델

## ◆ 인라인 이벤트 모델

- h1 태그를 클릭할 때 onclick 속성의 자바스크립트 코드를 실행

---

```
<body>
  <h1 onclick="alert('클릭')">Click</h1>
</body>
```

---

- 여러 줄의 자바스크립트 코드 사용

---

```
<body>
  <h1 onclick="var alpha=10;alert(alpha);">Click</h1>
</body>
```

---

# 11.5 인라인 이벤트 모델

## ◆ 인라인 이벤트 모델

- script 태그 안에 함수를 만들고 이를 호출하는 방식

---

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function whenClick(e) {
      alert('클릭');
    }
  </script>
</head>
<body>
  <h1 onclick="whenClick(event)">Click</h1>
</body>
</html>
```

---

## 11.6 디폴트 이벤트 제거

### ◆ 디폴트 이벤트

- 일부 HTML 태그에 이미 이벤트 리스너가 있는 것

---

```
<body>
  <form id="my-form">
    <label for="name">이름</label><br/>
    <input type="text" name="name" id="name" /><br/>
    <label for="pass">비밀번호</label><br/>
    <input type="password" name="pass" id="pass" /><br/>
    <label for="pass-check">비밀번호 확인</label><br/>
    <input type="password" id="pass-check" /><br/>
    <input type="submit" value="제출" />
  </form>
</body>
```

---

## 11.6 디폴트 이벤트 제거

### ◆ 디폴트 이벤트

- submit 이벤트 연결
- 이벤트 리스너에서 false를 리턴

---

```
<script>
    window.onload = function () {
        // 이벤트를 연결합니다.
        document.getElementById('my-form').onsubmit = function () {
            return false;
        };
    };
</script>
```

---

# 11.6 디폴트 이벤트 제거

## ◆ 디폴트 이벤트

- 입력 양식의 유효성 검사

---

```
<script>
    window.onload = function () {
        // 이벤트를 연결합니다.
        document.getElementById('my-form').onsubmit = function () {
            // 변수를 선언합니다.
            var pass = document.getElementById('pass').value;
            var passCheck = document.getElementById('pass-check').value;

            // 비밀번호가 같은지 확인합니다.
            if (pass == passCheck) {
                alert('성공');
            } else {
                alert('다시 입력해주세요. ');
                return false;
            }
        };
    };
</script>
```

---



## 11.6 디폴트 이벤트 제거

### ◆ 디폴트 이벤트

- form 태그의 onsubmit 이벤트 속성에 "return 함수()"형태를 입력

---

```
<script>
    function whenSubmit() {
        // 유효성 검사
        return false;
    }
</script>
<form onsubmit="return whenSubmit()">
    <!-- 생략 -->
</form>
```

---

## 11.7 이벤트 전달

### ◆ 이벤트 전달

- 네 개의 태그, 네 개의 이벤트

---

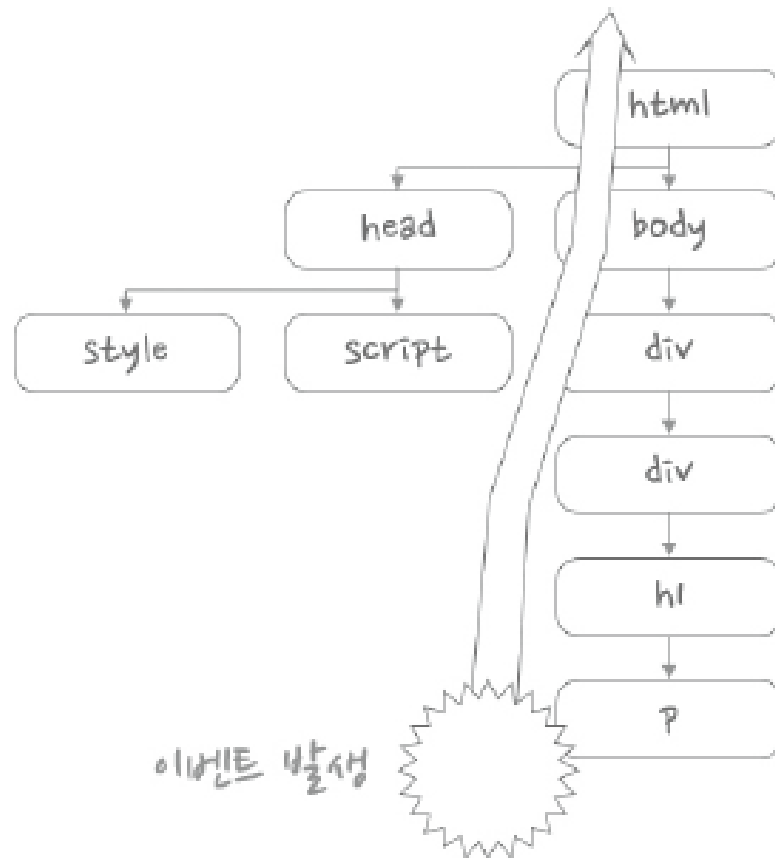
```
<!DOCTYPE html>
<html>
<head>
  <style>
    * { border: 3px solid black; }
  </style>
  <script>
  </script>
</head>
<body>
  <div onclick="alert('outer-div')">
    <div onclick="alert('inner-div')">
      <h1 onclick="alert('header')">
        <p onclick="alert('pagagraph')">Pagagraph</p>
      </h1>
    </div>
  </div>
</body>
</html>
```

---

# 11.7 이벤트 전달

## ◆ 이벤트 버블링

- 이벤트 버블링은 자식 노드에서 부모 노드 순으로 이벤트를 실행하는 것을 의미
- 이벤트 전달 순서는 이벤트 버블링을 따름



# 11.7 이벤트 전달

## ◆ 이벤트 전달을 막는 방법

---

```
<body>
  <h1 id="header">
    <p id="paragraph">Paragraph</p>
  </h1>
</body>
```

---

```
<script>
  window.onload = function () {
    // 이벤트를 연결합니다.
    document.getElementById('header').onclick = function () {
      alert('header');
    };
    document.getElementById('paragraph').onclick = function () {
      alert('paragraph');
    };
  };
</script>
```

---

## 11.7 이벤트 전달

### ◆ 이벤트 전달을 막는 방법

- 인터넷 익스플로러와 그외 브라우저가 이벤트 전달을 막는 방법
  - 인터넷 익스플로러 이벤트 객체의 `cancelBubble` 속성을 `true`로 변경합니다.
  - 그 이외의 브라우저 이벤트 객체의 `stopPropagation()` 메서드를 사용합니다.

---

```
document.getElementById('paragraph').onclick = function (e) {  
    // 이벤트 객체를 처리합니다.  
    var event = e || window.event;  
  
    // 이벤트 발생을 알립니다.  
    alert('paragraph');  
  
    // 이벤트 전달을 제거합니다.  
    event.cancelBubble = true;  
    if (event.stopPropagation) {  
        event.stopPropagation();  
    }  
};
```

---

## 11.8 인터넷 익스플로러 이벤트 모델

---

### ◆ 인터넷 익스플로러 이벤트 모델

- 두 가지 메서드로 이벤트를 연결하거나 제거할 수 있음
- 첫 번째 매개변수에 이벤트 속성을 사용

---

```
attachEvent(eventProperty, eventListener);  
detachEvent(eventProperty, eventListener);
```

---

# 11.8 인터넷 익스플로러 이벤트 모델

## ◆ 인터넷 익스플로러 이벤트 모델

### ■ 이벤트 연결

---

```
<!DOCTYPE html>
<html>
<head>
  <script>
    // 윈도우가 로드될 때
    window.attachEvent('onload', function () {
      // my-header를 가져옵니다.
      var header = document.getElementById('my-header');

      // 이벤트를 연결합니다.
      header.attachEvent('onclick', function () { alert('클릭'); });
      header.attachEvent('onclick', function () { alert('클릭'); });
      header.attachEvent('onclick', function () { alert('클릭'); });
    });
  </script>
</head>
<body>
  <h1 id="my-header">Click</h1>
</body>
</html>
```

---

## 11.8 인터넷 익스플로러 이벤트 모델

### ◆ 인터넷 익스플로러 이벤트 모델

- 이벤트 제거

---

```
<!DOCTYPE html>
<html>
<head>
  <script>
    window.onload = function () {
      var header = document.getElementById('my-header');
      var handler = function () { alert('클릭'); };

      header.attachEvent('onclick', handler);
      header.detachEvent('onclick', handler);
    };
  </script>
</head>
<body>
  <h1 id="my-header">Click</h1>
</body>
</html>
```

---



## 11.9 표준 이벤트 모델

### ◆ 표준 이벤트 모델

- 표준 이벤트 모델은 웹 표준을 만드는 단체인 W3C에서 공식적으로 지정한 DOM Level 2 이벤트 모델
- 인터넷 익스플로러 이벤트 모델과 달리 이벤트 캡처링을 지원
- 이벤트 이름을 매개변수로 입력

---

```
addEventListener(eventName, handler, useCapture)  
removeEventListener(eventName, handler)
```

---

## 11.9 표준 이벤트 모델

### ◆ 표준 이벤트 모델

- 이벤트 연결

---

```
<!DOCTYPE html>
<html>
<head>
  <script>
    window.onload = function () {
      var header = document.getElementById('my-header');

      header.addEventListener('click', function () {
        this.innerHTML += '+';
      });
    };
  </script>
</head>
<body>
  <h1 id="my-header">Click</h1>
</body>
</html>
```

---

# 11.9 표준 이벤트 모델

## ◆ 표준 이벤트 모델

### ■ 이벤트 모델의 통합적 사용

---

```
<script>
  window.onload = function () {
    var header = document.getElementById('my-header');

    if (header.attachEvent) {
      // 인터넷 익스플로러의 경우 실행합니다.
      var handler = function () {
        window.event.srcElement.style.color = 'red';
        window.event.srcElement.detachEvent('onclick', handler);
      };
      header.attachEvent('onclick', handler);
    } else {
      // 그 이외의 브라우저에서 실행합니다.
      var handler = function () {
        this.style.color = 'red';
        this.removeEventListener('click', handler);
      };
      header.addEventListener('click', handler);
      header.addEventListener('click', handler);
    }
  };
</script>
```

---