08

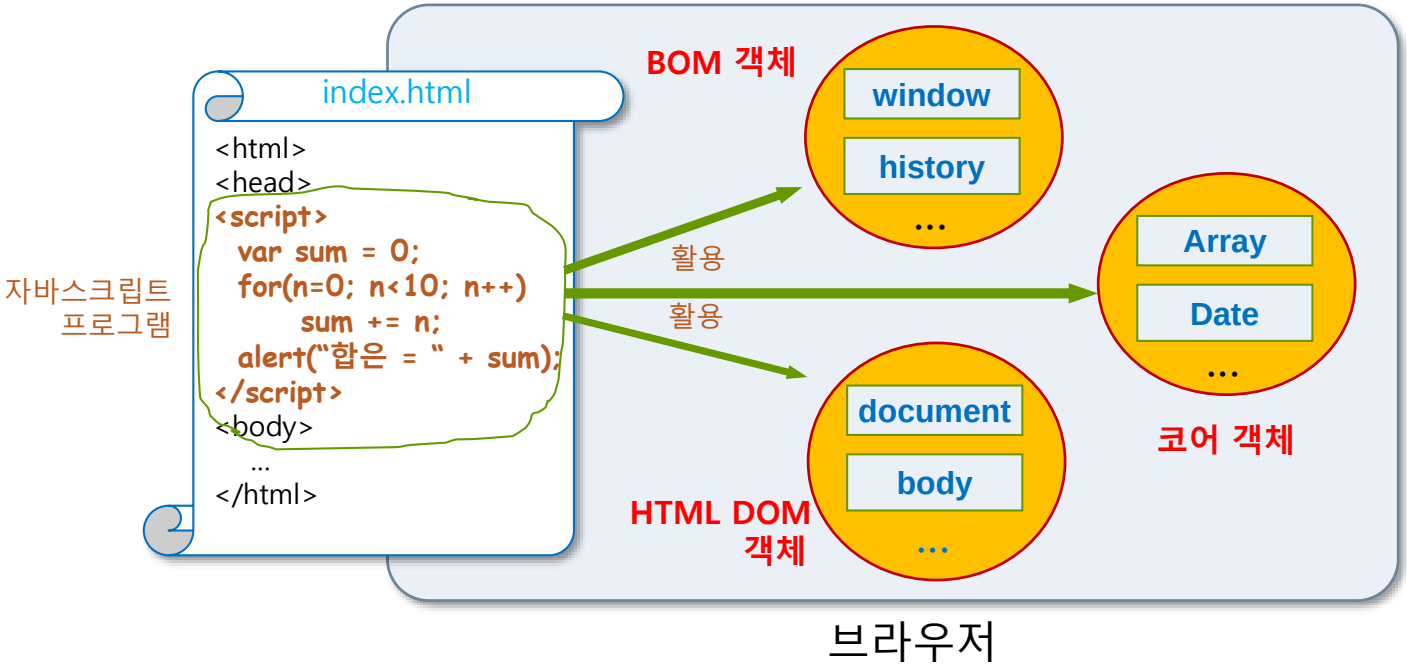
HTML DOM과 Document

강의 목표

1. HTML DOM의 필요성을 이해한다.
2. DOM 트리와 HTML 페이지의 관계를 이해한다.
3. DOM 객체의 구조와 HTML 태그와의 관계를 이해한다.
4. DOM 객체를 통해 HTML 태그의 출력 모양과 콘텐츠를 제어할 수 있다.
5. document 객체를 이해하고, write() 메소드를 활용할 수 있다.
6. createElement() 등을 통해 동적으로 DOM 객체를 웹 페이지에 추가,삭제할 수 있다.

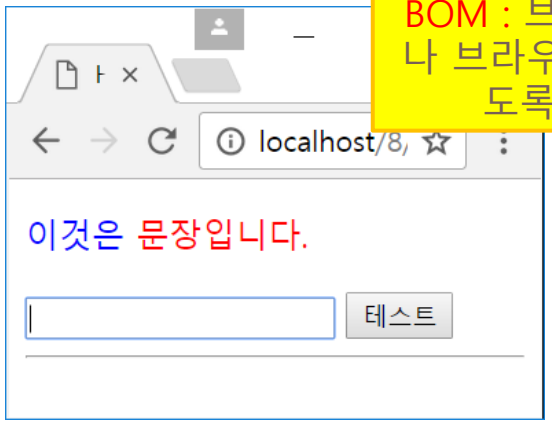
HTML 페이지와 자바스크립트 객체

- 자바스크립트 코드는 브라우저로부터 3 가지 유형의 객체를 제공받아 활용할 수 있다.

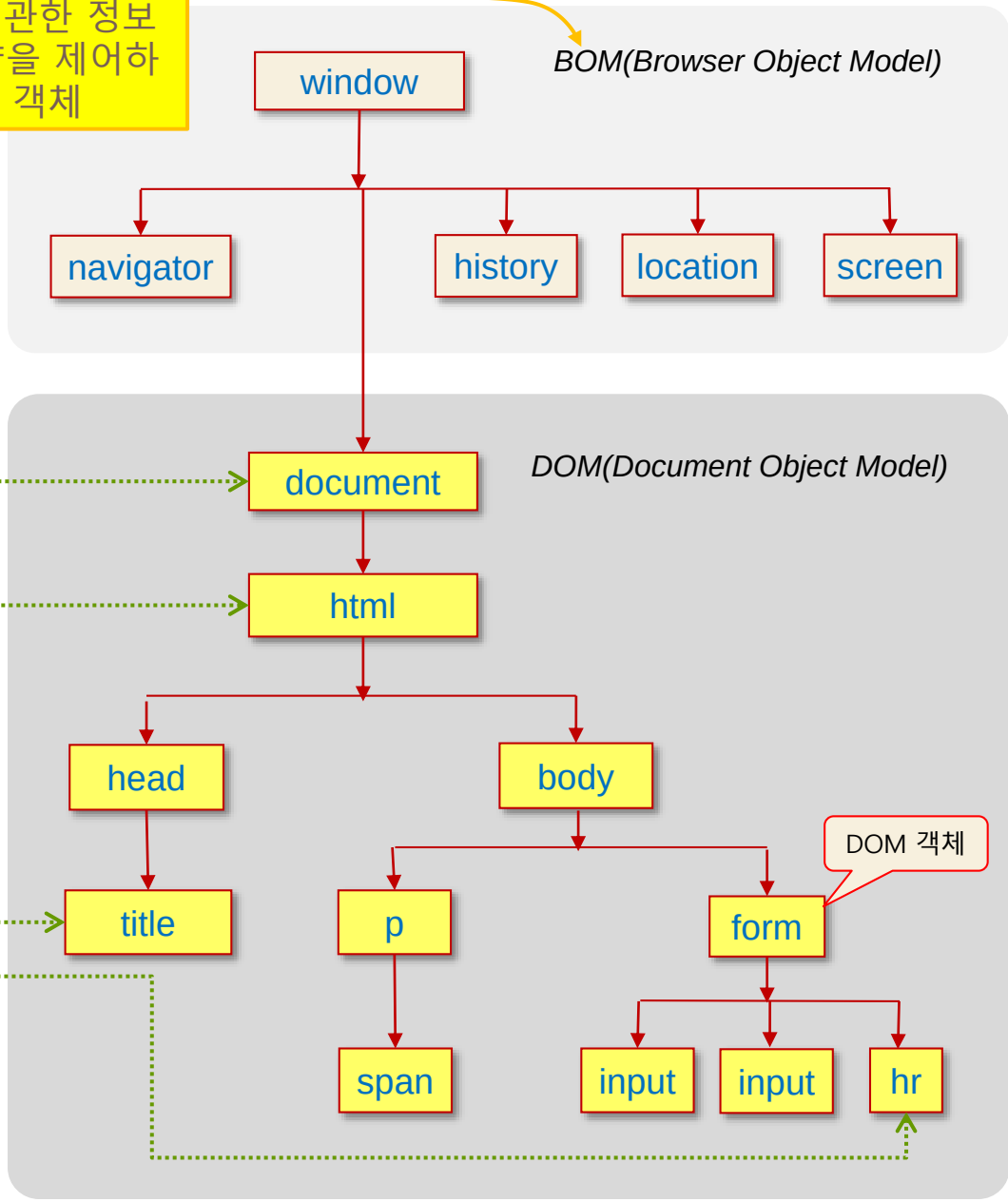


HTML DOM(Document Object Model)

- HTML DOM(간단히 DOM)
  - 웹 페이지에 작성된 **HTML 태그 당 객체**(DOM 객체) 생성
  - 목적
    - HTML 태그가 출력된 모양이나 콘텐츠를 제어하기 위해
      - *DOM 객체를 통해 각 태그의 CSS3 스타일 시트 접근 및 변경*
      - *HTML 태그에 의해 출력된 텍스트나 이미지 변경*
- DOM 트리
  - HTML 태그의 포함관계에 따라 DOM 객체의 트리(tree) 생성
  - DOM 트리는 부모 자식 관계
- DOM 객체
  - DOM 트리의 한 노드
  - HTML 태그 당 하나의 DOM 객체 생성
    - DOM 노드(Node), DOM 엘리먼트(Element) 라고도 불림



BOM : 브라우저에 관한 정보나 브라우저의 모양을 제어하도록 제공하는 객체




```
<!DOCTYPE html>
<html>
<head>
    <title>HTML DOM 트리</title>
</head>
<body>
<p style="color:blue">이것은
    <span style="color:red">문장입니다.
    </span>
</p>
<form>
    <input type="text">
    <input type="button" value="테스트">
    <hr>
</form>
</body>
</html>
```

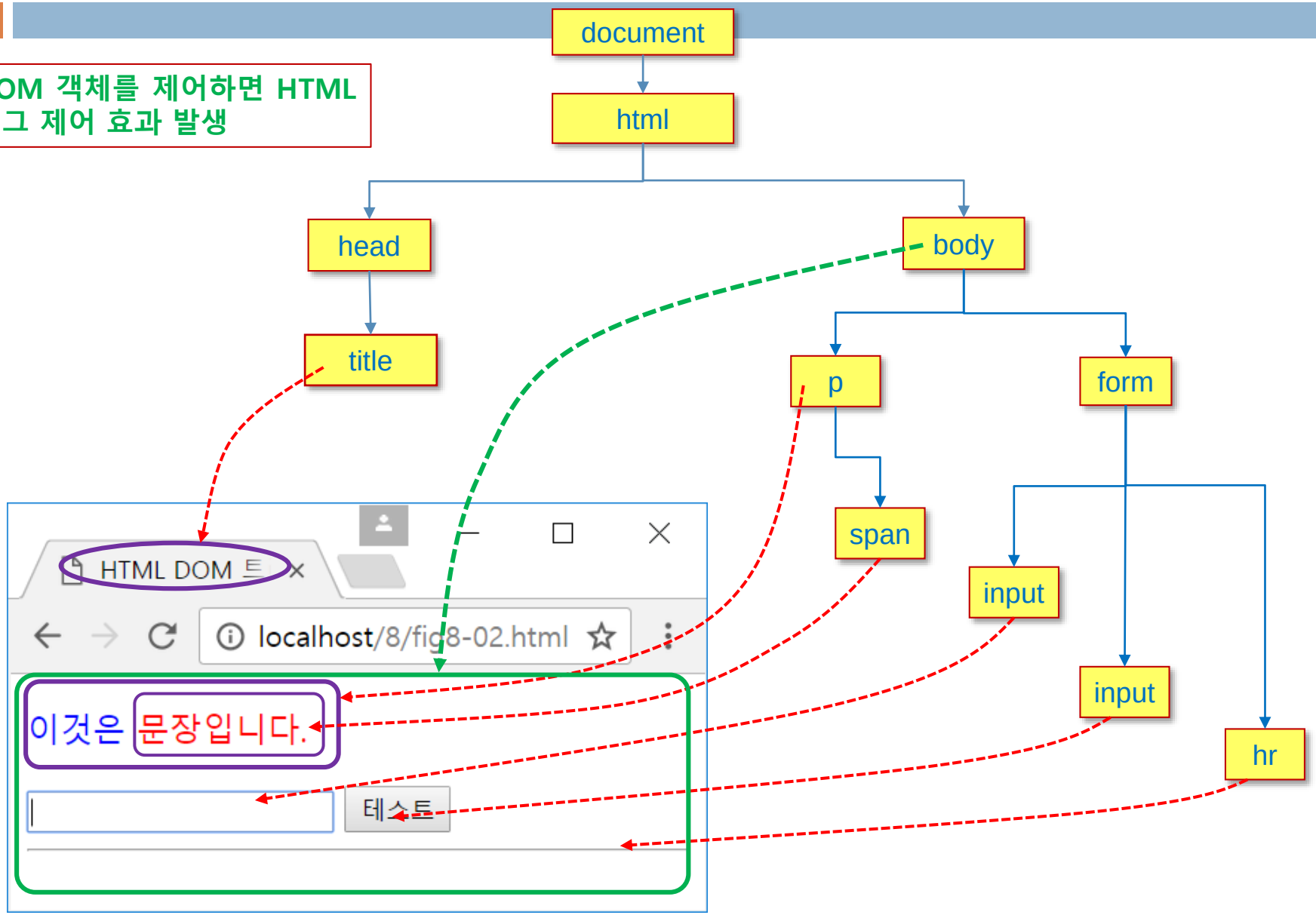
DOM 트리의 특징

- DOM 트리의 특징
  - DOM 트리의 루트는 document 객체
  - DOM 객체의 종류는 HTML 태그 종류만큼
  - HTML 태그 당 DOM 객체가 하나씩 생성
  - HTML 태그의 포함관계에 따라 DOM 트리에 부모 자식 관계

- 브라우저가 HTML 태그를 화면에 그리는 과정
  1. 브라우저가 DOM 트리의 틀(document 객체) 생성
  2. 브라우저가 HTML 태그를 읽고 DOM 트리에 DOM 객체 생성
  3. 브라우저는 DOM 객체를 화면에 출력
  4. HTML 문서 로딩이 완료되면 DOM 트리 완성
  5. DOM 객체 변경 시, 브라우저는 해당 HTML 태그의 출력 모양을 바로 갱신

DOM 객체와 HTML 페이지의 화면 출력

DOM 객체를 제어하면 HTML 태그 제어 효과 발생



HTML 태그의 요소

- HTML 태그
  - 엘리먼트(element)로도 불림
  - 다음 5 가지 요소로 구성
    - 엘리먼트 이름
    - 속성
    - CSS3 스타일
    - 이벤트 리스너
    - 콘텐츠(innerHTML)

DOM 객체의 구성 요소

- DOM 객체는 5 개의 요소 구성
  - 프로퍼티(property)
    - HTML 태그의 속성(attribute) 반영
  - 메소드(method)
    - DOM 객체의 멤버 함수로서, HTML 태그 제어 가능
  - 컬렉션(collection)
    - 자식 DOM 객체들의 주소를 가지는 등 배열과 비슷한 집합적 정보
  - 이벤트 리스너(event listener)
    - HTML 태그에 작성된 이벤트 리스너 반영
    - 약 70여개의 이벤트 리스너를 가질 수 있음
  - CSS3 스타일
    - HTML 태그에 설정된 CSS3 스타일 시트 정보를 반영
    - DOM 객체의 style 프로퍼티를 통해 HTML 태그의 모양 제어 가능

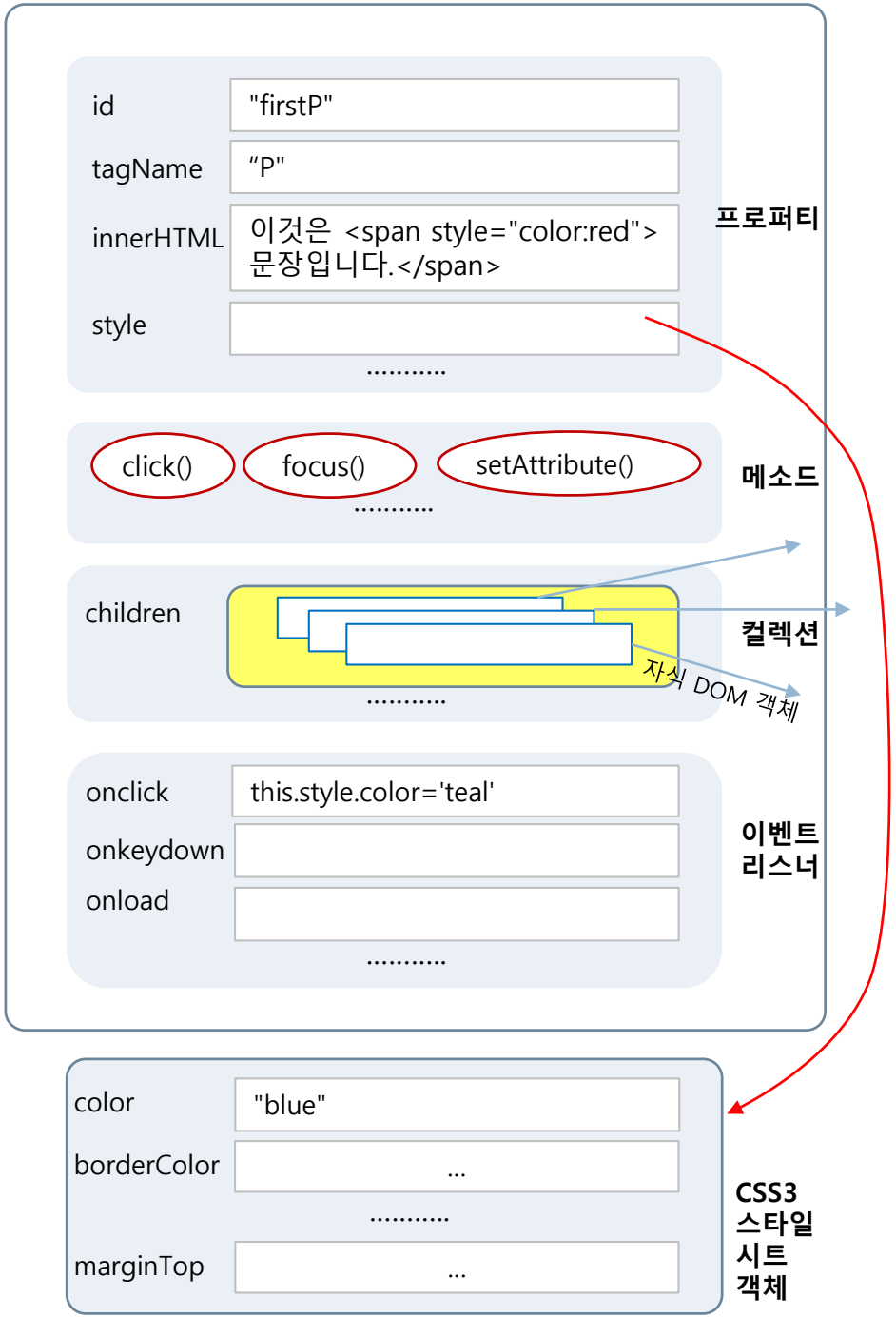
DOM 객체의 구성

- 프로퍼티(property)
- 메소드(method)
- 컬렉션(collection)
- 이벤트 리스너(event listener)
- CSS3 스타일

```
<p id="firstP"
    style="color:blue"
    onclick="this.style.color='teal'">
   이것은
   <span style="color:red">
          문장입니다.
   </span>
</p>
```

** ... 태그**

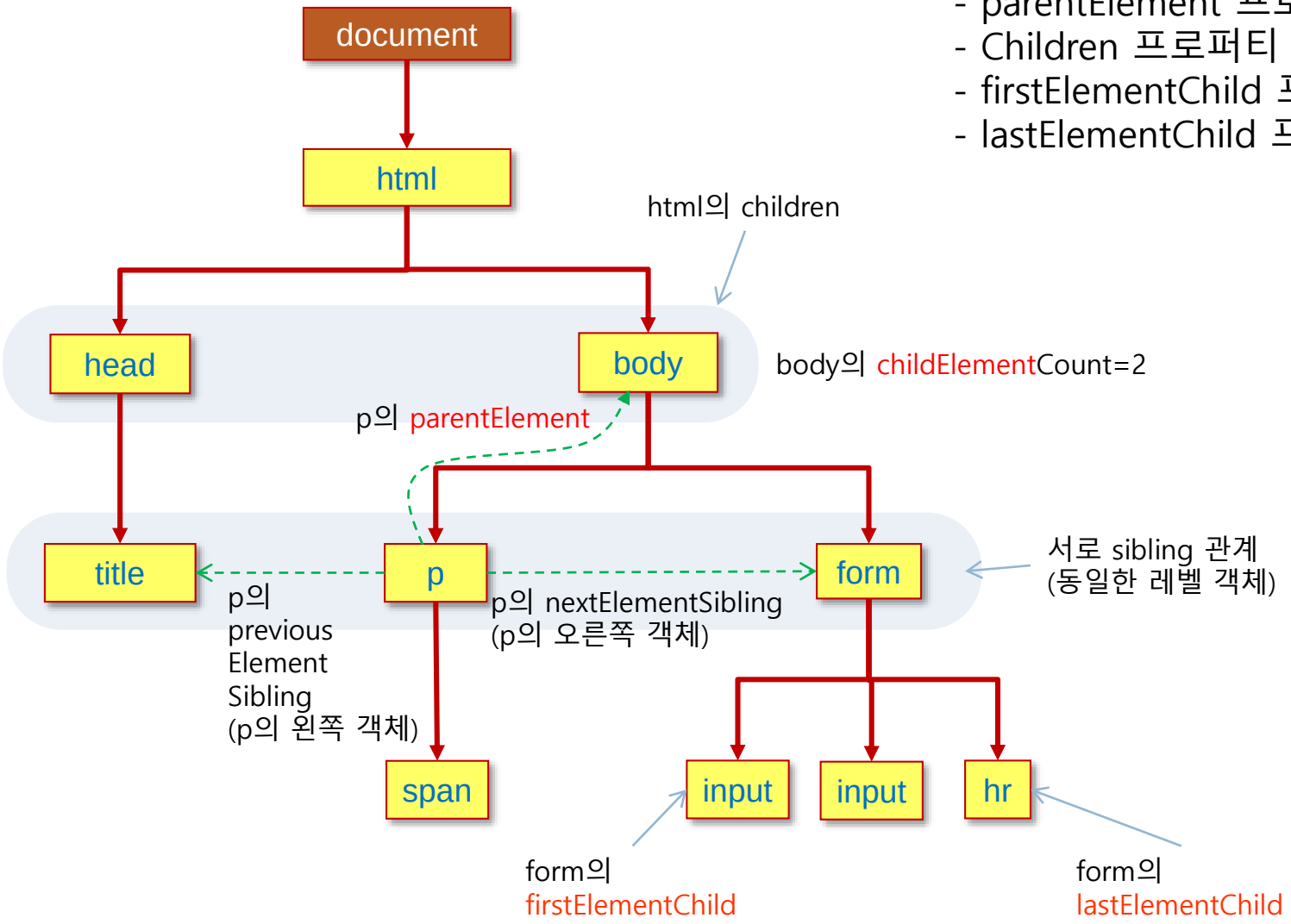
DOM 객체 p



DOM 객체의 구성 요소

DOM 객체의 프로퍼티와 DOM 객체사이의 관계

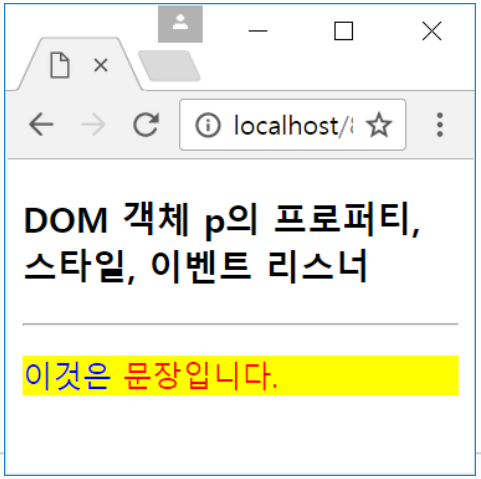
- parentElement 프로퍼티
- Children 프로퍼티
- firstElementChild 프로퍼티
- lastElementChild 프로퍼티

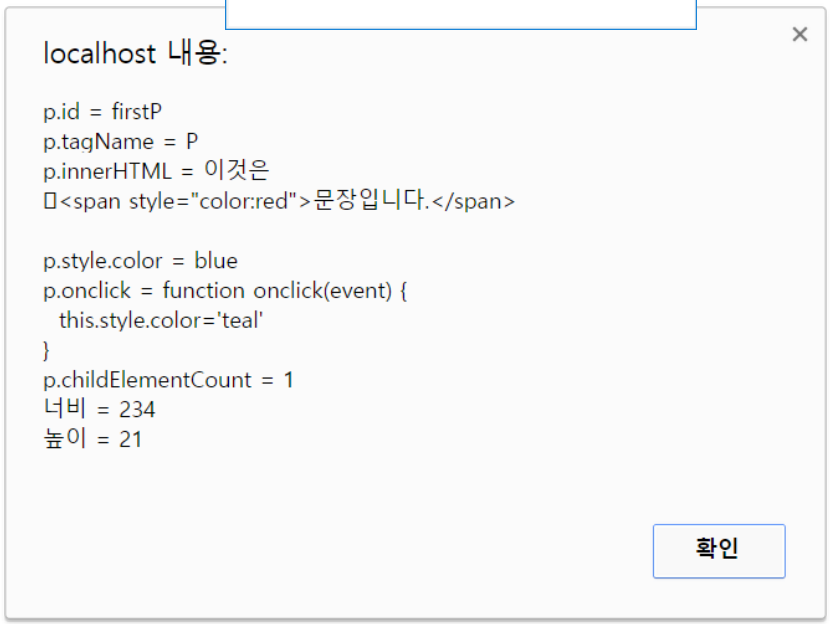


예제 8-1 DOM 객체의 구조 출력 : p 객체 사례


```
<!DOCTYPE html>
<html>
<head><title>HTML DOM 트리</title></head>
<body>
<h3>DOM 객체 p의 프로퍼티, 스타일, 이벤트 리스너</h3>
<hr>
<p id="firstP"
   style="color:blue; background:yellow"
   onclick="this.style.color='teal'">
    이것은 <span style="color:red">문장입니다.
   </span>
</p>
<script>
   var p = document.getElementById("firstP");
   var text = "p.id = " + p.id + "₩n";
   text += "p.tagName = " + p.tagName + "₩n";
   text += "p.innerHTML = " + p.innerHTML + "₩n";
   text += "p.style.color = " + p.style.color + "₩n";
   text += "p.onclick = " + p.onclick + "₩n";
   text += "p.childElementCount = " + p.childElementCount + "₩n";
   text += "너비 = " + p.offsetWidth + "₩n";
   text += "높이 = " + p.offsetHeight + "₩n";
   alert(text);
</script>
</body>
</html>
```

id가 firstP인 태그의 DOM 찾기





DOM 객체 다루기

- DOM 객체 구분 → id 태그 속성

```
<p id="firstP">안녕하세요</p>
```

- DOM 객체 찾기→ document.getElementById()

```
var p = document.getElementById("firstP");  // id 값이 firstP인 DOM 객체 리턴
p.style.color = "red";                      // p 객체의 글자 색을 red로 변경
```

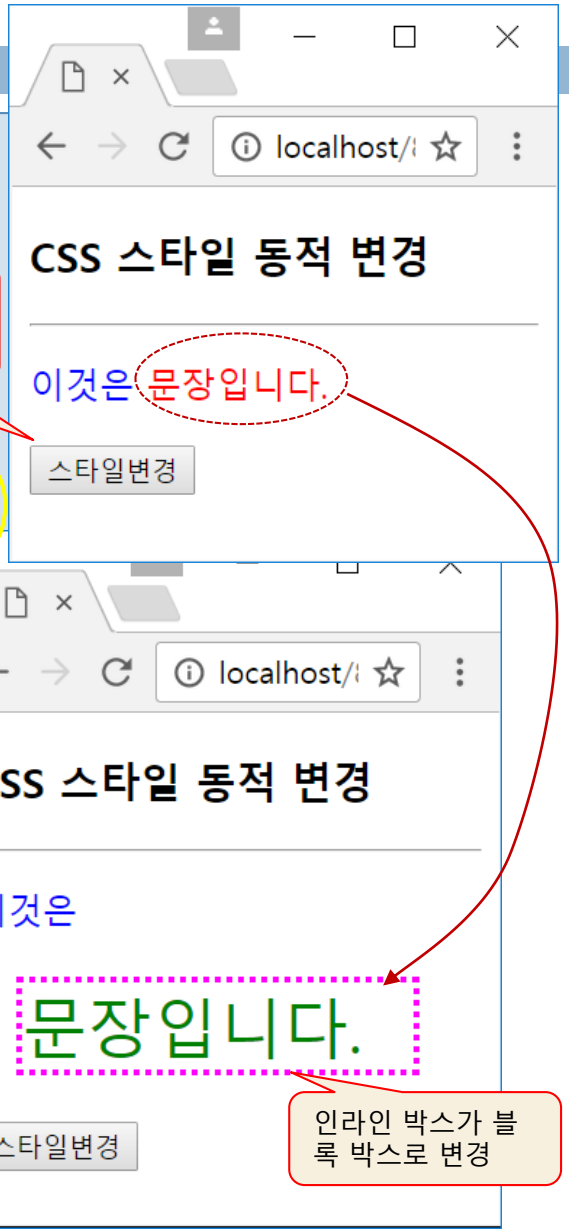
- DOM 객체의 CSS3 스타일 동적 변경
  - CSS3 스타일 프로퍼티는 다음과 같이 사용
    - background-color 스타일 프로퍼티 -> backgroundColor
    - font-size 스타일 프로퍼티 -> fontSize

```
<span id="mySpan" style="color:red">문장입니다.</span>

var span = document.getElementById("mySpan");  // id가 mySpan인 객체 찾기
span.style.color = "green";                    // '문장입니다'의 글자 색을 green으로 변경
span.style.fontSize = "30px";                  // '문장입니다'의 폰트를 30px 크기로 변경
span.style.border = "3px dotted magenta"; // 3픽셀의 magenta 점선 테두리
```

예제 8-2 의 CSS3 스타일 동적 변경


```
<!DOCTYPE html>
<html><head><title>CSS 스타일 동적 변경</title>
<script>
function change() {
    var span = document.getElementById("mySpan");
    span.style.color = "green"; // 글자 색 green
    span.style.fontSize = "30px"; // 글자 크기는 30픽셀
    span.style.display = "block"; // 블록 박스로 변경
    span.style.width = "6em"; // 박스의 폭. 6 글자 크기
    span.style.border = "3px dotted magenta"; // 3픽셀 점선 magenta 테두리
    span.style.margin = "20px"; // 상하좌우 여백 20px
}
</script>
</head>
<body>
<h3>CSS 스타일 동적 변경</h3>
<hr>
<p style="color:blue" >이것은
    <span id="mySpan" style="color:red">문장입니다.</span>
</p>
<input type="button" value="스타일변경" onclick="change()">
</body>
</html>
```

버튼을 클릭하면 change() 함수 호출. 스타일 변경



innerHTML 프로퍼티

- innerHTML 프로퍼티
  - 시작 태그와 종료 태그 사이에 들어 있는 HTML 컨텐츠

```
<p id="firstP" style="color:blue">
  여기에<span style="color:red">
  클릭하세요.</span>
</p>
```

innerHTML

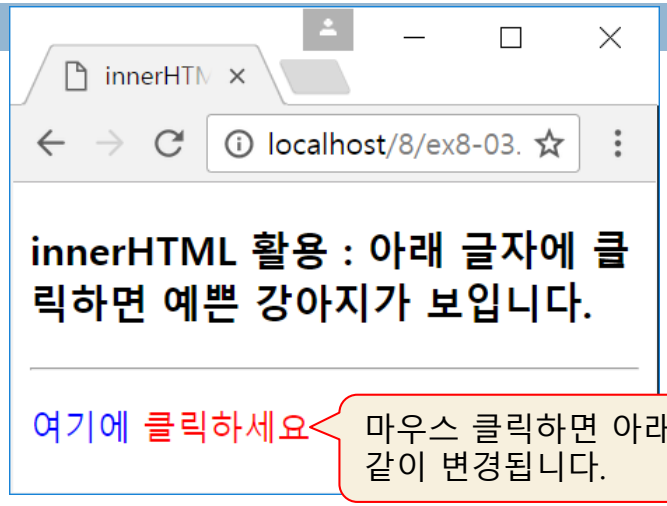
  - innerHTML 프로퍼티 수정 -> HTML 태그의 콘텐트 변경

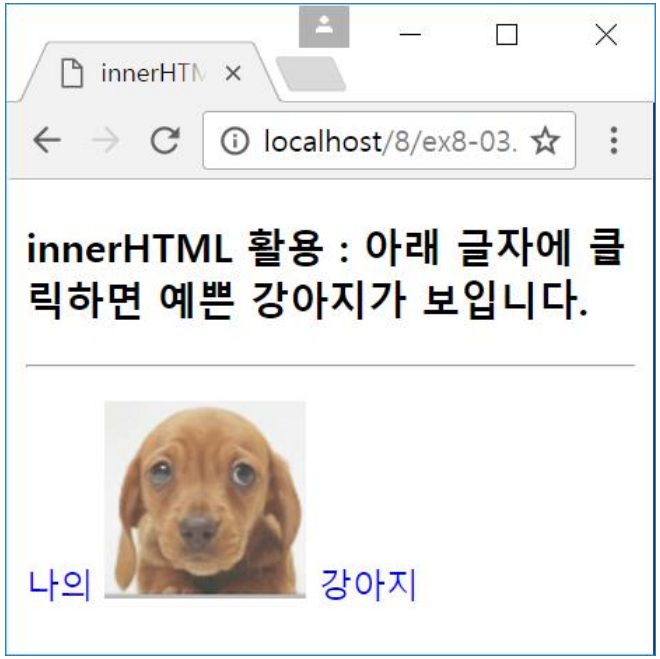
```
var p = document.getElementById("firstP");
p.innerHTML= "나의 <img src='puppy.jpg'>강아지입니다.";
```

```
<p id="firstP" style="color:blue">
  나의 <img src='puppy.jpg'>
  강아지입니다.
</p>
```

예제 8-3 innerHTML을 이용하여 HTML 콘텐츠 동적 변경


```
<!DOCTYPE html>
<html>
<head>
<title>innerHTML 활용</title>
<script>
function change() {
   var p = document.getElementById("firstP");
   p.innerHTML= "나의 <img src='puppy.png'> 강아지";
}
</script>
</head>
<body>
<h3>innerHTML 활용 : 아래 글자에 클릭하면
 예쁜 강아지가 보입니다.</h3>
<hr>
<p id="firstP" style="color:blue"
   onclick="change()">
   여기에<span style="color:red">클릭하세요</span>
</p>
</body>
</html>
```





this

- this 키워드
  - 객체 자신을 가리키는 자바스크립트 키워드
  - DOM 객체에서 객체 자신을 가리키는 용도로 사용
    - 예) 태그 자신의 배경을 orange 색으로 변경

```
<div onclick="this.style.backgroundColor='orange'">
```

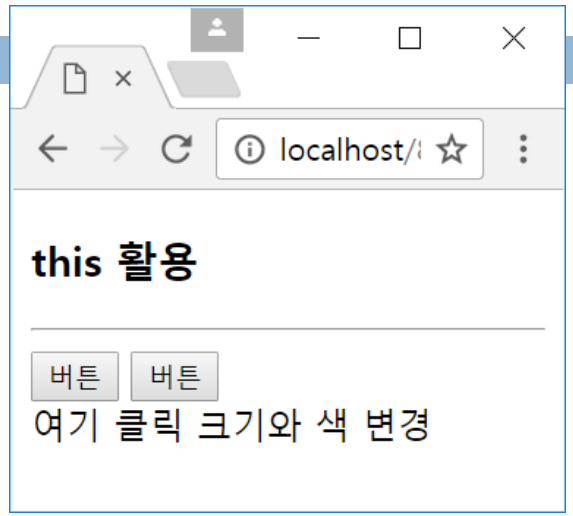
    - 예) 버튼이 클릭되면 자신의 배경색을 orange로 변경

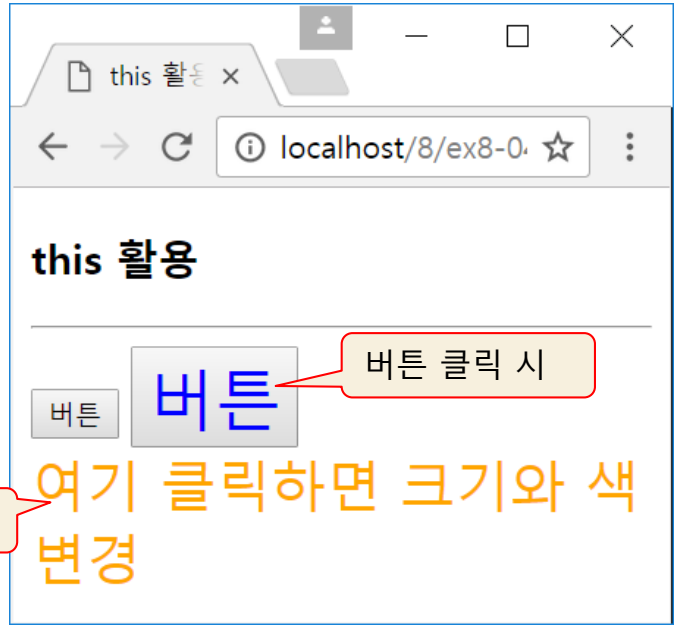
```
<button onclick="this.style.backgroundColor='orange'">
```

예제 8-4 this 활용


```
<!DOCTYPE html>
<html>
<head>
<title>this 활용</title>
<script>
function change(obj, size, color) {
    obj.style.color = color;
    obj.style.fontSize = size;
}
</script>
</head>
<body>
<h3>this 활용</h3>
<hr>
<button onclick="change(this, '30px', 'red')">버튼</button>
<button onclick="change(this, '30px', 'blue')">버튼</button>
<div onclick="change(this, '25px', 'orange')">
    여기 클릭하면 크기와 색 변경
</div>
</body>
</html>
```

this는 이 <button> 객체의 주소





document 객체

- document
  - HTML 문서 전체를 대변하는 객체
    - 프로퍼티 – HTML 문서의 전반적인 속성 내포
    - 메소드 – DOM 객체 검색, DOM 객체 생성, HTML 문서 전반적 제어
  - DOM 객체를 접근하는 경로의 시작점
  - DOM 트리의 최상위 객체
    - 브라우저는 HTML 문서 로드 전, document 객체를 먼저 생성
    - document 객체를 루트로 하여 DOM 트리 생성
- document 객체 접근
  - window.document 또는 document 이름으로 접근
  - document 객체는 DOM 객체가 아님
    - 연결된 스타일 시트가 없음

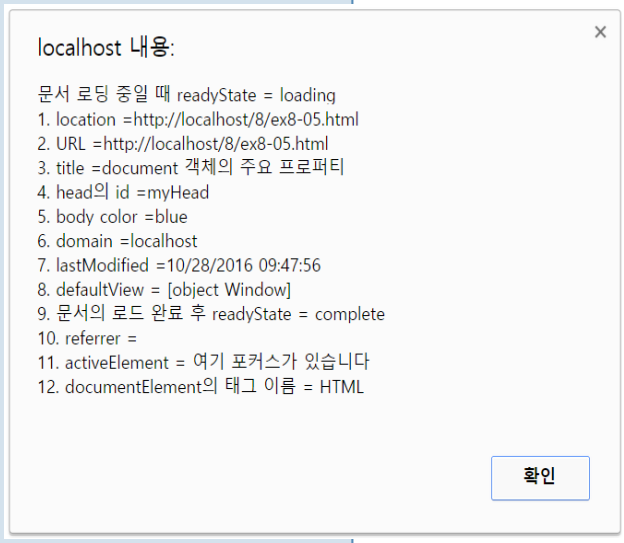
~~document.style.color~~ = "red"; // **오류**. document에는 CSS3 스타일 시트가 연결되지 않음

예제 8-5 document 객체의 프로퍼티 출력

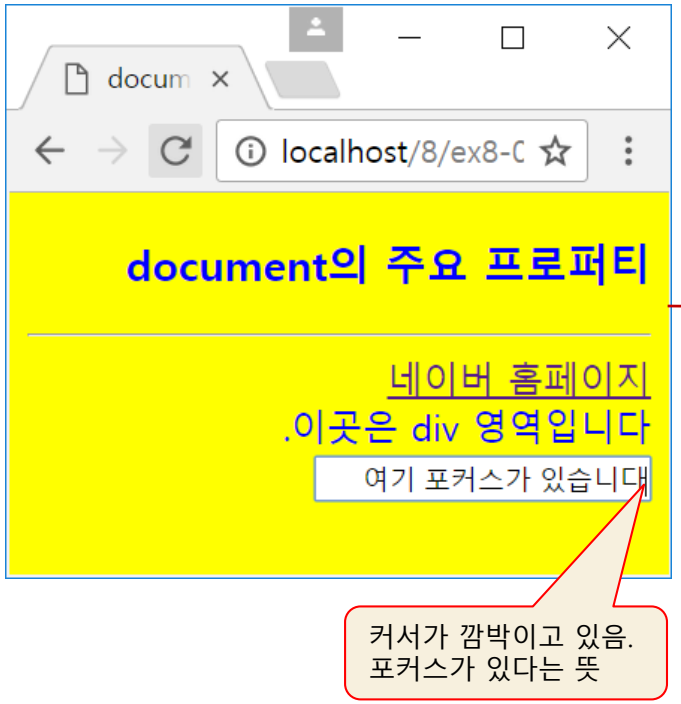
```
<!DOCTYPE html>
<html>
<head id="myHead">
<title>document 객체의 주요 프로퍼티</title>
<script>
   var text = "문서 로딩 중일 때 readyState = " + document.readyState + "₩n";
</script>
</head>
<body style="background-color:yellow; color:blue; direction:rtl"
      onload="printProperties()">
<h3>document의 주요 프로퍼티</h3>
<hr>
<a href="http://www.naver.com">네이버 홈페이지</a>
<div>이곳은 div 영역입니다.</div>
<input id="input" type="text" value="여기 포커스가 있습니다">

<script>
// 문서가 완전히 로드(출력)되었을 때, 현재 document의 프로퍼티 출력
function printProperties() {
   document.getElementById("input").focus(); // <input> 태그에 포커스를 줌

   text += "1. location =" + document.location + "₩n";
   text += "2. URL =" + document.URL + "₩n";
   text += "3. title =" + document.title + "₩n";
   text += "4. head의 id =" + document.head.id + "₩n";
   text += "5. body color =" + document.body.style.color + "₩n";
   text += "6. domain =" + document.domain + "₩n";;
   text += "7. lastModified =" + document.lastModified + "₩n";
   text += "8. defaultView = " + document.defaultView + "₩n";
   text += "9. 문서의 로드 완료 후 readyState = " + document.readyState + "₩n";
   text += "10. referrer = " + document.referrer + "₩n";
   text += "11. activeElement = " + document.activeElement.value + "₩n";
   text += "12. documentElement의 태그 이름 = " + document.documentElement.tagName + "₩n";
   alert(text);
}
</script>
</body>
</html>
```

Direction:rtl → 오른쪽에 붙여서 문서 출력

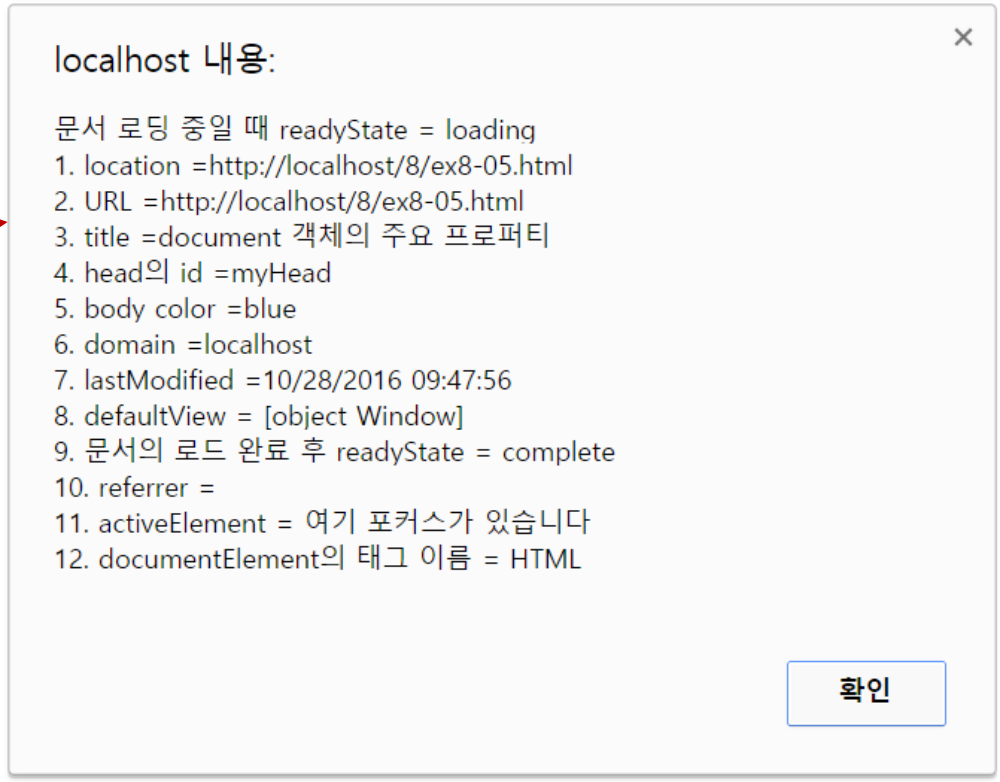
onload: "printProperiees()" → 문서의 로딩이 완료되면 printProperties() 호출



예제 8-5 document 객체의 프로퍼티 출력



로드 후 경고 창 출력



경고창에 document 객체의 주요 프로퍼티 출력

DOM 트리에서 DOM 객체 찾기

태그 이름으로 찾기

- document.getElementsByTagName()
  - 태그 이름이 같은 모든 DOM 객체들을 찾아 컬렉션 리턴
  - 예) 태그의 모든 DOM 객체 찾기

```
var divTags = document.getElementsByTagName("div");
```

```
var n = divTags.length; // 웹 페이지에 있는 <div> 태그의 개수
```

class 속성으로 찾기

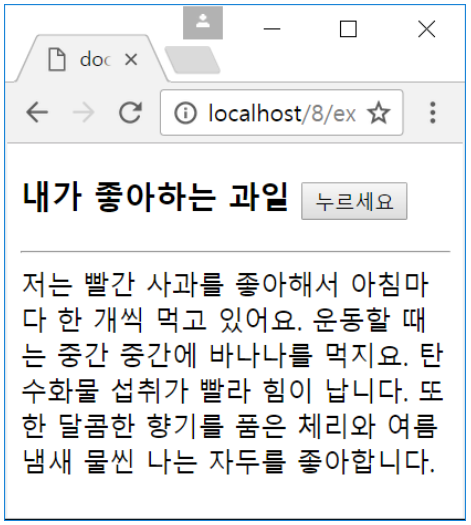
- document.getElementsByClassName()
  - class 속성이 같은 모든 DOM 객체들을 찾아 컬렉션 리턴
  - 예)

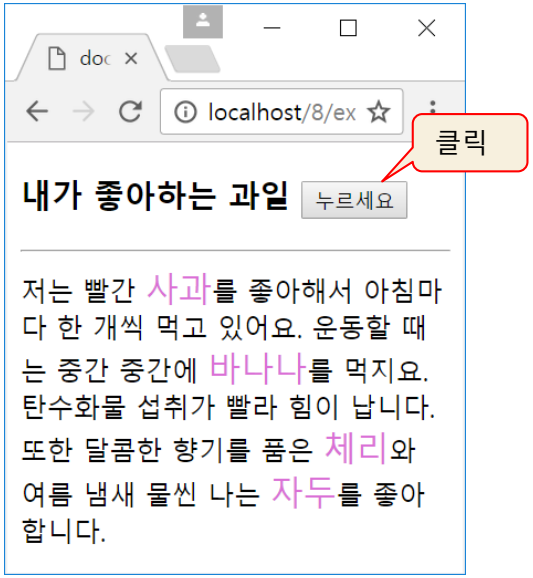
```
<div class="plain">...</div>
<div class="important">...</div>
<div class="plain">...</div>
```

```
var plainClasses = document.getElementsByClassName("plain");
var n = plainClasses.length; // 웹 페이지에 class="plain" 속성을 가진 태그의 개수
```

예제 8-6 태그 이름으로 DOM 객체 찾기, getElementsByTagName()


```
<!DOCTYPE html>
<html>
<head>
<title>document.getElementsByTagName()</title>
<script>
function change() {
    var spanArray = document.getElementsByTagName("span");
    for(var i=0; i<spanArray.length; i++) {
        var span = spanArray[i];
        span.style.color = "orchid";
        span.style.fontSize = "20px";
    }
}
</script>
</head>
<body>
<h3>내가 좋아하는 과일
    <button onclick="change()">누르세요</button>
</h3>
<hr>
저는 빨간 <span>사과</span>를 좋아해서
아침마다 한 개씩 먹고 있어요. 운동할 때는 중간 중간에
<span>바나나</span>를 먹지요. 탄수화물 섭취가 빨라
힘이 납니다. 또한 달콤한 향기를 품은 <span>체리</span>와
여름 냄새 물씬 나는 <span>자두</span>를 좋아합니다.
</body>
</html>
```





document.write()와 document.writeln()

- HTML 페이지 로딩 과정
  1. 브라우저는 HTML 페이지 로드 전 빈 상태 document 생성
  2. 브라우저는 HTML 페이지를 위에서 아래로 해석
  3. HTML 태그들을 document 객체에 담아간다(DOM 객체 생성).
  4. </html> 태그를 만나면 document 객체를 완성하고 닫는다.

- write()
  - document 객체에 담긴 HTML 콘텐츠 마지막에 HTML 태그들을 추가
    - 추가되는 HTML 태그들은 DOM 객체로 바뀌고 DOM 트리에 추가
    - 삽입된 HTML 태그들이 브라우저 화면에 출력
    - 예)

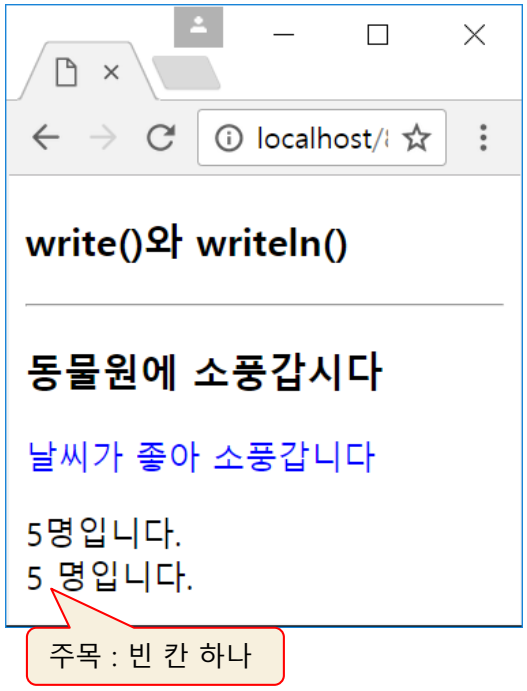
```
document.write("<h3>Welcome to my home</h3>");
document.write(2+3); // 합한 결과 5 출력
document.write("<p>오늘은 " + "sunny day 입니다</p>");
```

- writeln()
  - HTML 텍스트에 '₩n'을 덧붙여 출력. 한 칸 띄는 효과
  - 한줄을 띄려면

```
document.write("<br>");
```

예제 8-7 write()와 writeln() 메소드 활용

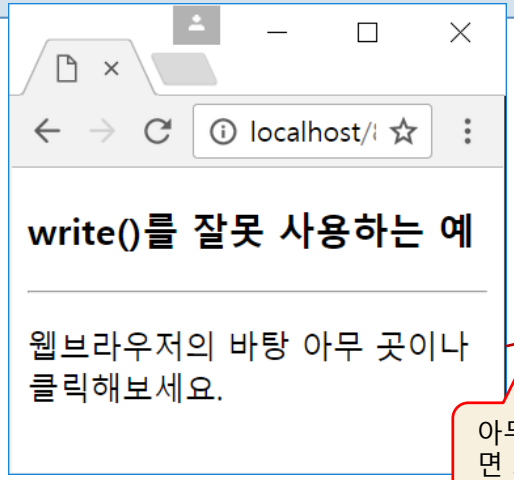

```
<!DOCTYPE html>
<html>
<head>
<title>write()와 writeln() 활용</title>
</head>
<body>
<h3>write()와 writeln() 활용</h3>
<hr>
<script>
   document.write("<h3>동물원에 소풍갑시다</h3>");
   document.write("<p style='color:blue'>날씨가 좋아 ");
   document.write("소풍갑니다</p>");
   document.write(2+3);
   document.write("명입니다.<br>"); // 다음 줄로 넘어가기

   document.writeln(5); // 다음 줄에 넘어가지 못함
   document.writeln("명입니다.<br>");
</script>
</body>
</html>
```



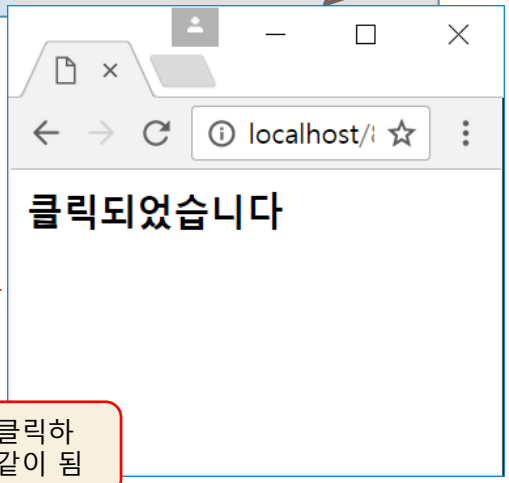
예제 8-8 write()를 잘못 사용하는 예


```
<!DOCTYPE html>
<html>
<head>
<title>write()를 잘못 사용하는 예</title>
</head>
<body onclick="document.write('<h3>클릭되었습니다</h3>')">
<h3>write()를 잘못 사용하는 예</h3>
<hr>
<p>웹브라우저의 바탕 아무 곳이나 클릭해보세요.</p>
</body>
</html>
```

HTML 문서를 모두 로드한 후 document 객체가 닫히기 때문에 새로운 문서가 작성됨



아무 곳이나 클릭하면 오른쪽과 같이 됨



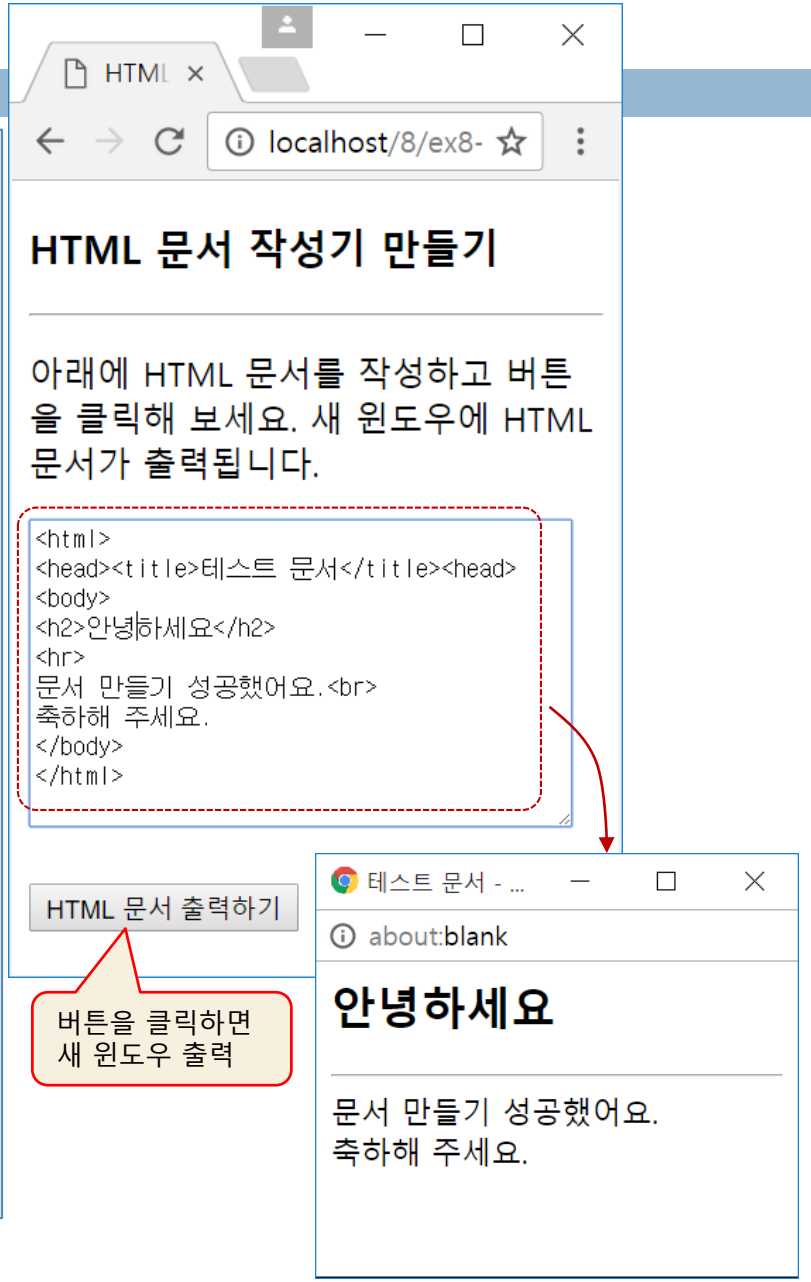
document의 열기와 닫기, open()과 close()

- document.open()
  - 현재 브라우저에 출력된 HTML 콘텐츠를 지우고 새로운 HTML 페이지 시작. 즉 document 객체에 담긴 DOM 트리를 지우고 새로 시작
- document.close()
  - 현재 브라우저에 출력된 HTML 페이지 완성
  - 더 이상 document.write() 할 수 없음
- 예)

```
// 현재 HTML 페이지의 내용을 지우고 다시 시작

document.open();
document.write("<html><head>...<body>안녕하세요.");
document.write(".....");
document.write("</body></html>");
document.close();
```

예제 8-9 HTML 문서 작성 연습 페이지 만들기


```
<!DOCTYPE html>
<html>
<head><title>HTML 문서 작성기 만들기</title>
<script>
var win=null;
function showHTML() {
    if(win == null || win.closed)
        win = window.open("", "outWin", "width=300,height=200");

    var textArea = document.getElementById("srcText");
    win.document.open();
    win.document.write(textArea.value);
    win.document.close();
}
</script>
</head>
<body>
<h3>HTML 문서 작성기 만들기 </h3>
<hr>
<p>아래에 HTML 문서를 작성하고 버튼을 클릭해 보세요.
새 윈도우에 HTML 문서가 출력됩니다.</p>
<textarea id="srcText" rows="10" cols="50"></textarea>
<br>
<br>
<button onclick="showHTML()">HTML 문서 출력하기</button>
</body>
</html>
```



문서의 동적 구성

- DOM 객체 동적 생성: document.createElement("태그이름")
  - 태그이름의 DOM 객체 생성
    - 예)

```
var newDIV = document.createElement("div");
```

```
newDIV.innerHTML = "새로 생성된 DIV입니다.";
```

```
newDIV.setAttribute("id", "myDiv");
newDIV.style.backgroundColor = "yellow";
```

- DOM 트리에 삽입
  - 부모.appendChild(DOM객체);
  - 부모.insertBefore(DOM객체 [, 기준자식]);
    - 예) 생성한 태그를 <p "id=p"> 태그의 마지막 자식으로 추가

```
var p = document.getElementById("p");
p.appendChild(newDiv);
```

- DOM 객체의 삭제
  - var removedObj = 부모.removeChild(떼어내고자하는자식객체);
    - 예)

```
var myDiv = document.getElementById("myDiv");
var parent = myDiv.parentElement;
parent.removeChild(myDiv);     // 부모에서 myDiv 객체 삭제
```

태그의 DOM 객체 동적 생성


```
var newDIV = document.createElement("div");
newDIV.innerHTML = "새로 생성된 DIV입니다.";
newDIV.setAttribute("id", "myDiv");
newDIV.style.backgroundColor = "yellow";
```



```
<div id="myDiv"
        style="background-color:yellow">
        새로 생성된 DIV입니다.
</div>
```

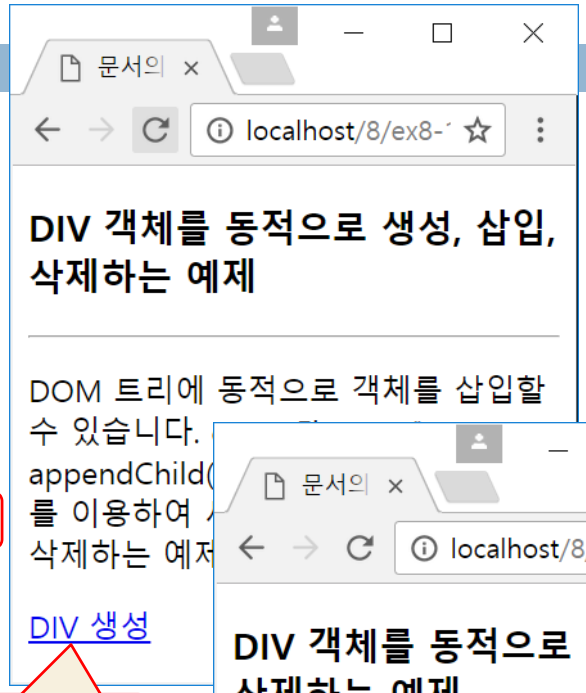
* 이 자바스크립트 코드는 사실상 오른쪽의 태그 정보를 가진 DOM 객체 생성

예제 8-10 HTML 태그의 동적 추가 및 삭제

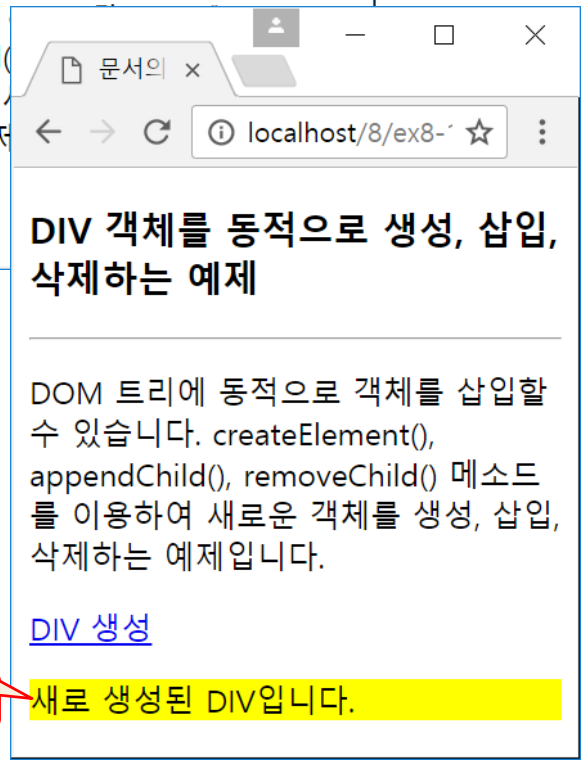

```
<!DOCTYPE html>
<html>
<head><title>문서의 동적 구성</title>
<script>
function createDIV() {
    var obj = document.getElementById("parent");
    var newDIV = document.createElement("div");
    newDIV.innerHTML = "새로 생성된 DIV입니다.";
    newDIV.setAttribute("id", "myDiv");
    newDIV.style.backgroundColor = "yellow";
    newDIV.onclick = function() {
        var p = this.parentElement; // 부모 HTML 태그 요소
        p.removeChild(this); // 자신을 부모로부터 제거
    };
    obj.appendChild(newDIV);
}
</script>
</head>
<body id="parent">
<h3>DIV 객체를 동적으로 생성, 삽입, 삭제하는 예제</h3>
<hr>
<p>DOM 트리에 동적으로 객체를 삽입할 수 있습니다.
createElement(), appendChild(),
removeChild() 메소드를 이용하여 새로운 객체를 생성,
삽입, 삭제하는 예제입니다.</p>
<a href="javascript:createDIV()">DIV 생성</a><p>
<p>
</body>
</html>
```

생성

삭제



클릭하면 아래와 같이 태그가 삽입



클릭하면 삭제