JavaScript

◆ HTML DOM과 Document

정수아

Contents

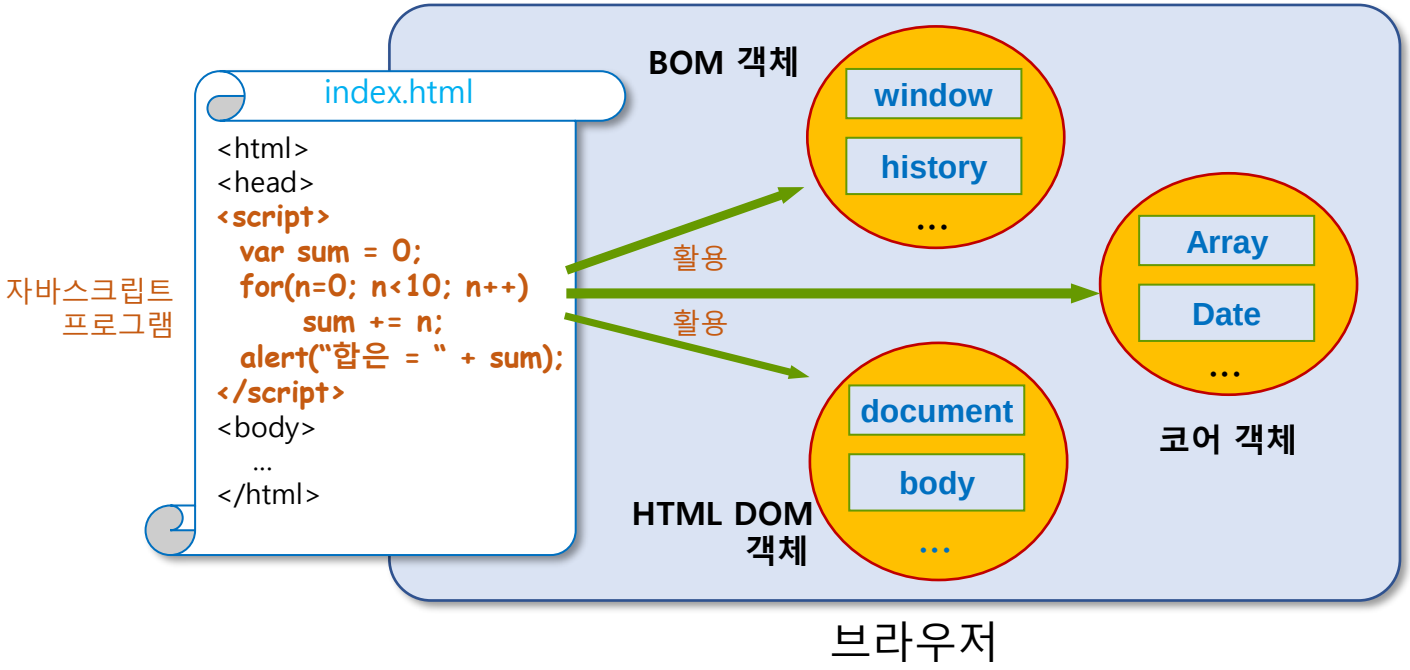
01

HTML DOM과 Document

HTML 페이지와 자바스크립트 객체

❖ 3가지 유형의 객체

- 자바스크립트 코드는 브라우저로부터 3가지 유형의 객체를 제공받아 활용



HTML DOM(Document Object Model)

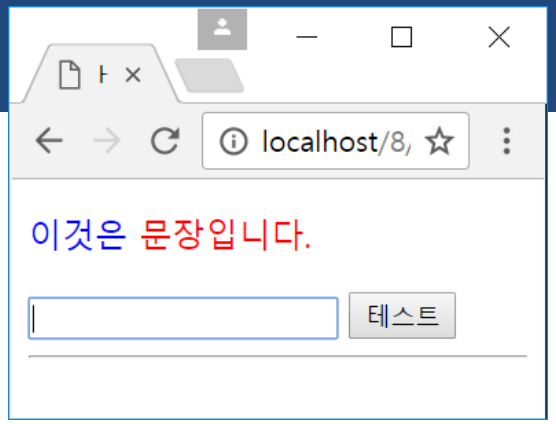
❖ **HTML DOM**

- 간단하게 DOM이라고도 부름
- 웹 페이지에 작성된 HTML 태그마다 객체(DOM 객체) 생성

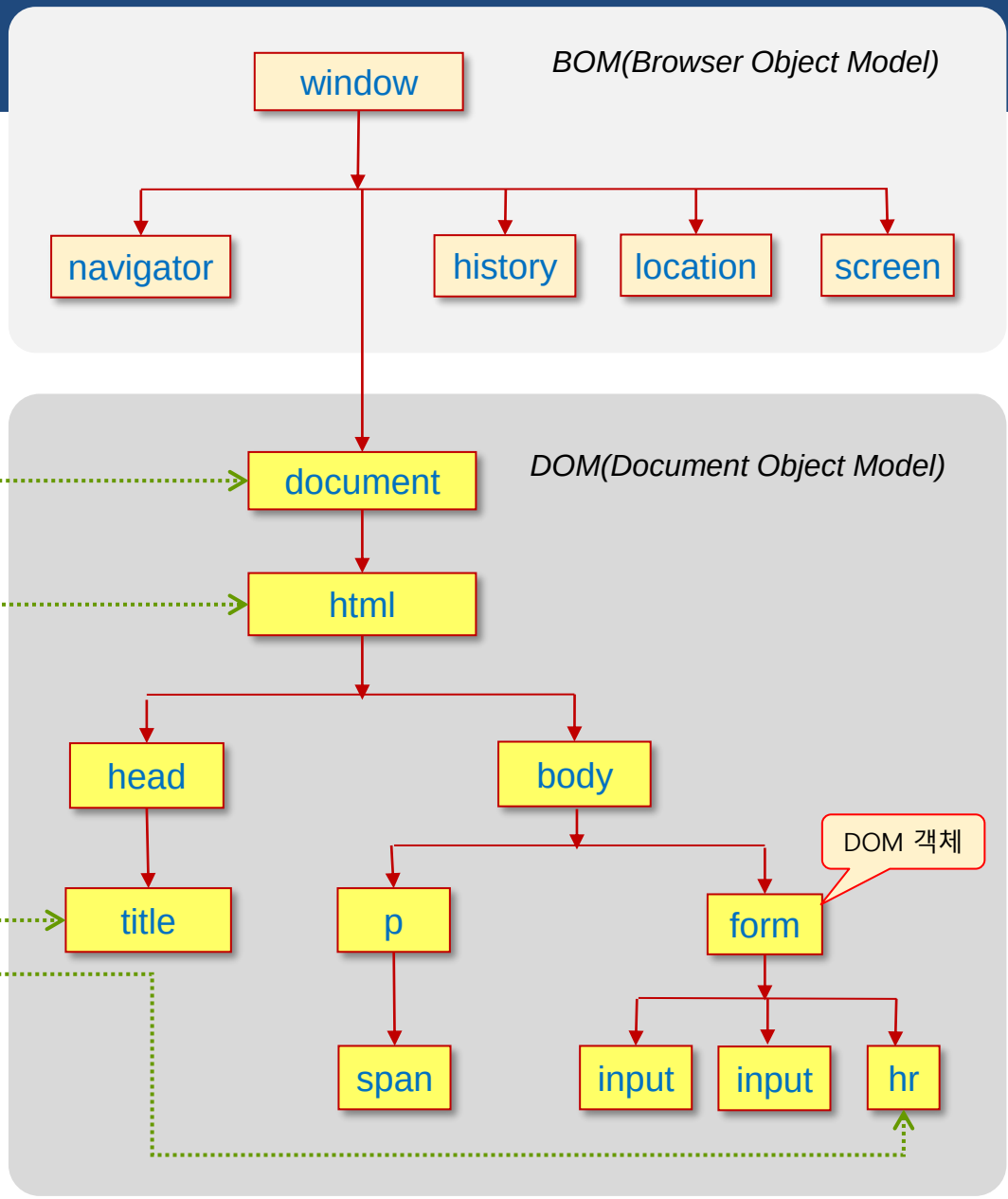
- 목적
  - HTML 태그가 출력된 모양이나 컨텐츠를 제어
  - DOM 객체를 통해 각 태그의 CSS 스타일 시트 접근 및 변경
  - HTML 태그에 의해 출력된 텍스트나 이미지 변경

- DOM 트리
  - HTML 태그의 포함 관계에 따라 DOM 객체의 트리(tree) 생성
  - DOM 트리는 부모 자식 관계로 구성

- DOM 객체
  - DOM 트리의 한 노드
  - HTML 태그 당 하나의 DOM 객체 생성
    - 각 노드는 DOM 노드 또는 DOM 엘리먼트라고 부름




```
<!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**

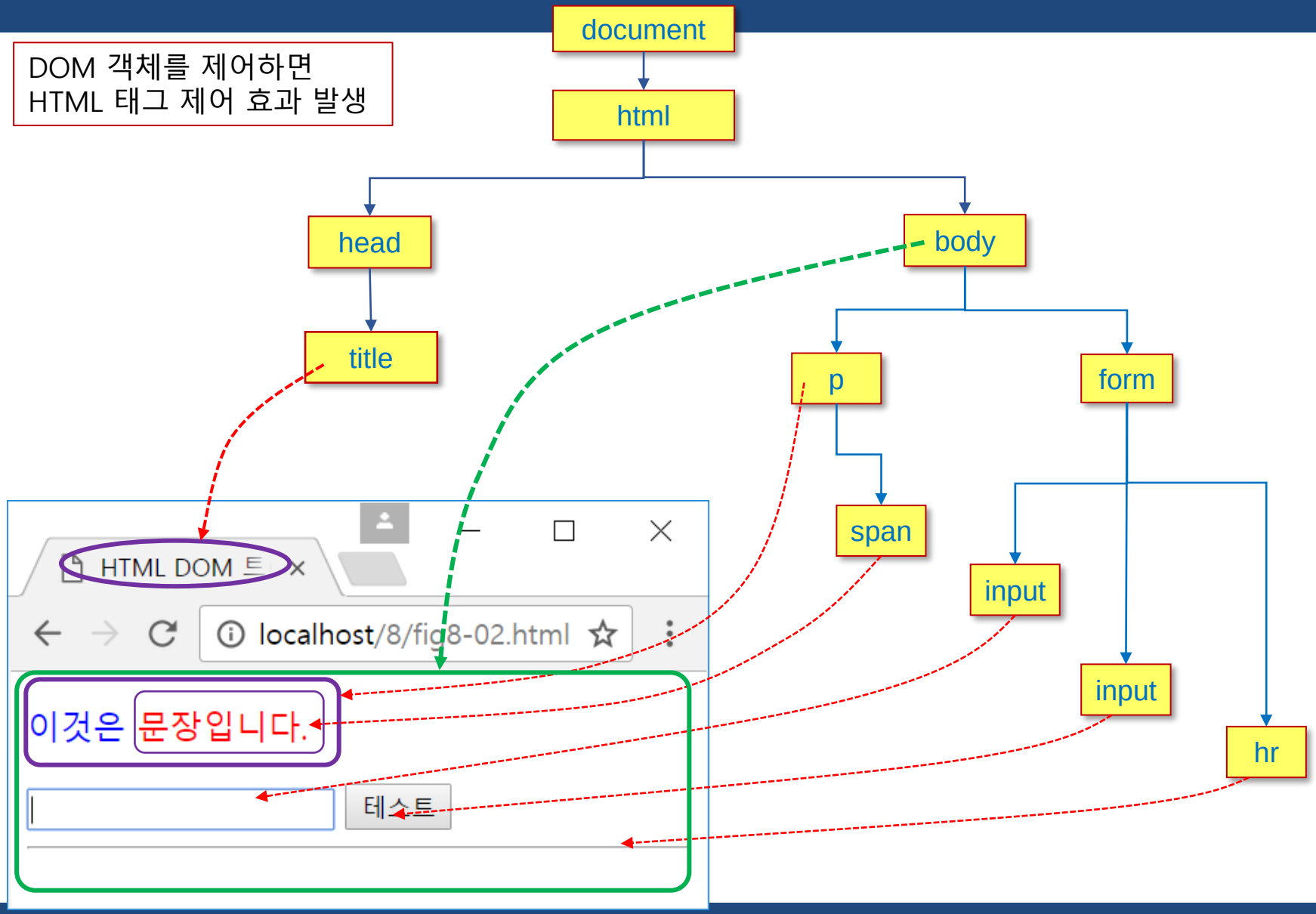
- DOM 트리의 루트는 document 객체
- DOM 객체의 종류는 HTML 태그 종류만큼 있음
  - HTML 태그 당 DOM 객체가 하나씩 생성
- HTML 태그의 포함관계에 따라 DOM 트리에 부모 자식 관계로 구성

❖ **브라우저가 HTML 태그를 화면에 그리는 과정**

- 브라우저가 DOM 트리의 틀(document 객체) 생성
- 브라우저가 HTML 태그를 읽고 DOM 트리에 DOM 객체 생성
- 브라우저는 DOM 객체를 화면에 출력
- HTML 문서 로딩이 완료되면 DOM 트리 완성
- 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



DOM 객체의 구성 요소

❖ **DOM 객체는 5개의 요소로 구성**

- 프로퍼티(property)
  - HTML 태그의 속성(attribute) 반영
- 메서드(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

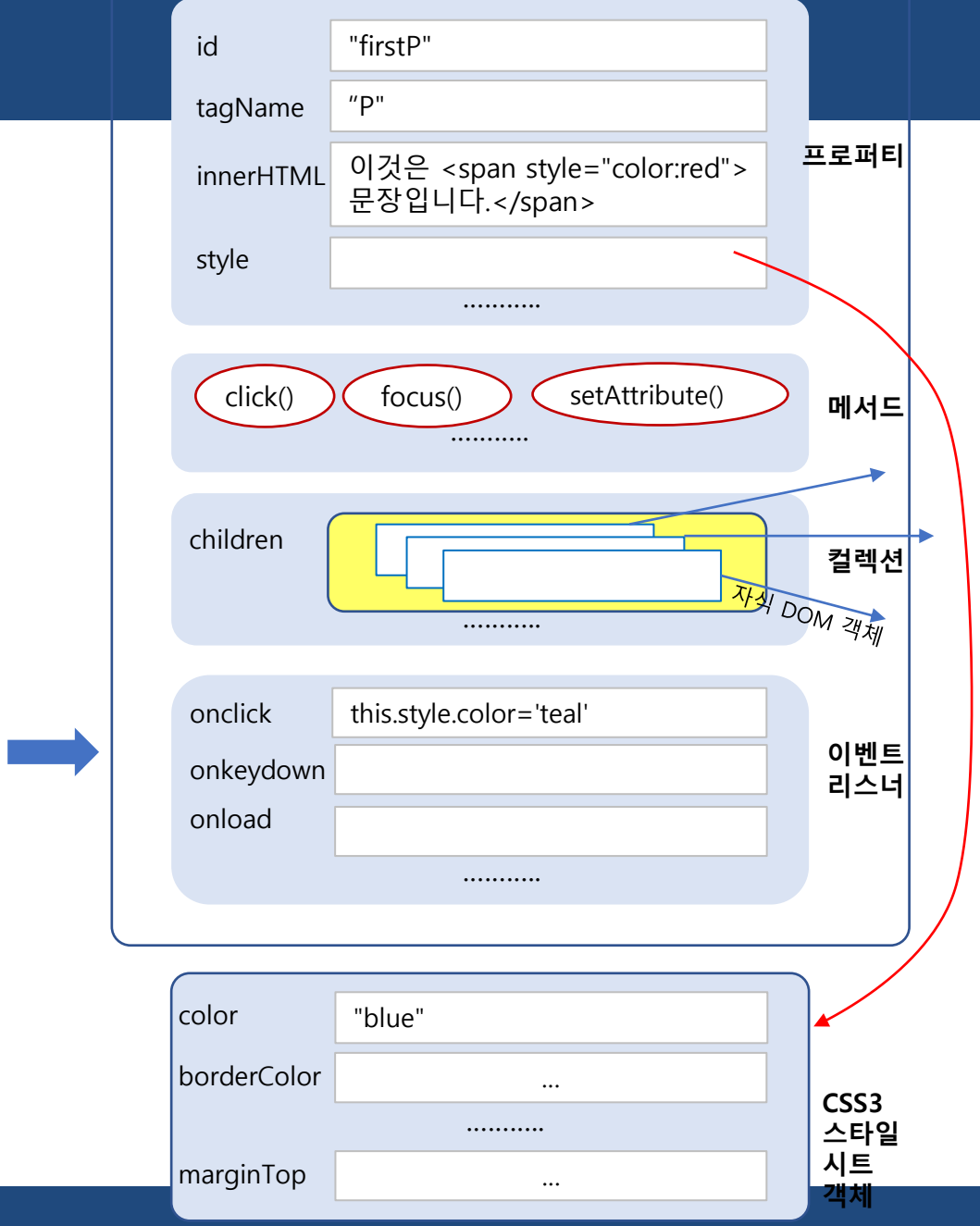
DOM 객체 p

DOM 객체의 구성

- 프로퍼티(property)
- 메서드(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**

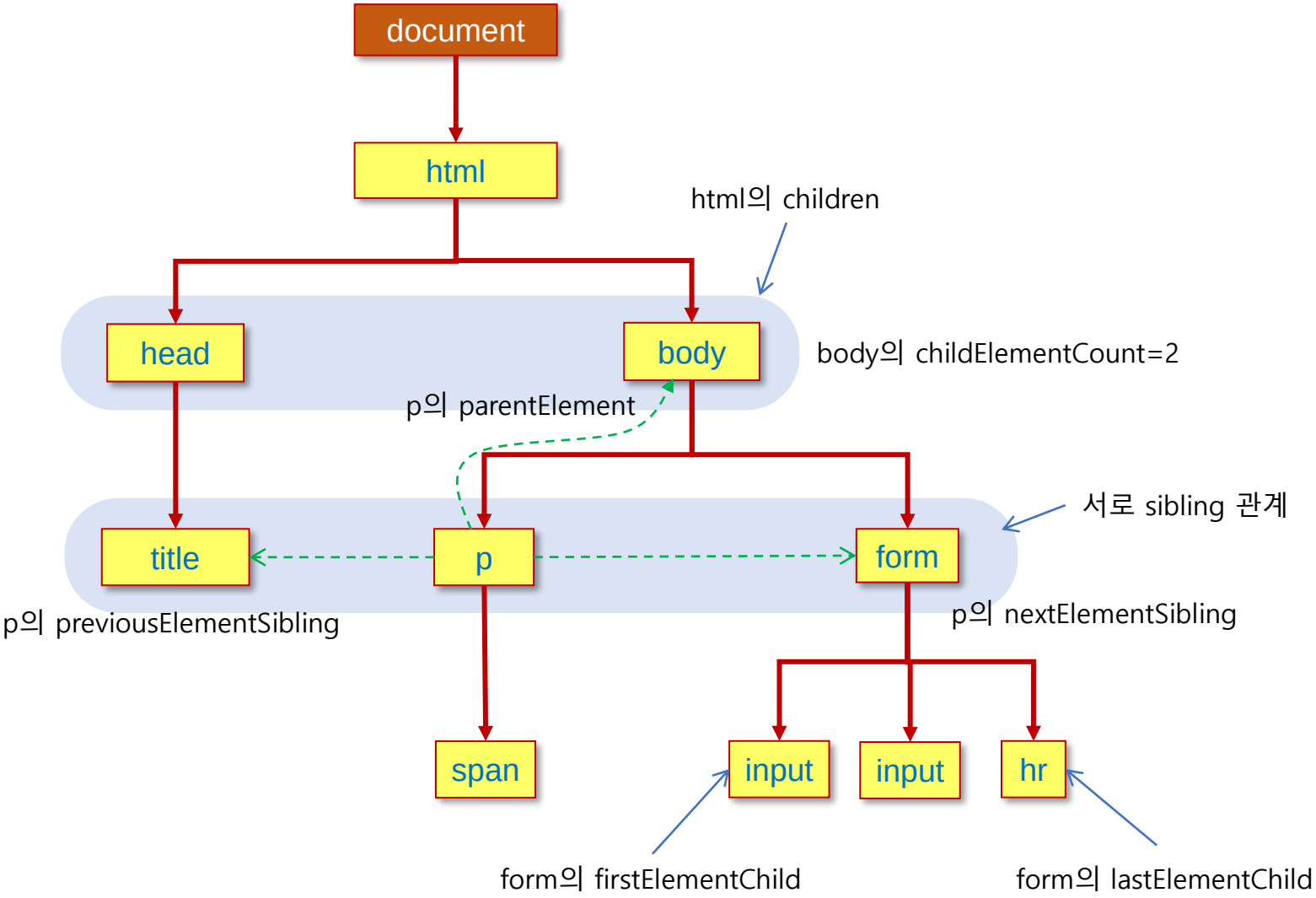


DOM 객체의 프로퍼티와 DOM 객체 사이의 관계

❖ **DOM 객체 사이의 관계**

- DOM 객체들은 DOM 트리에서 부모, 자식, 형제 관계로 연결
- 사용되는 프로퍼티 목록
  - parentElement 프로퍼티
    - 부모 객체
  - children 프로퍼티
    - 직계 자식들의 컬렉션
  - firstElementChild 프로퍼티
    - 첫 번째 직계 자식
  - lastElementChild 프로퍼티
    - 마지막 직계 자식
  - sibling
    - previousElementSibling 프로퍼티 : 왼쪽 sibling 객체
    - nextElementSibling 프로퍼티 : 오른쪽 sibling 객체

DOM 객체의 프로퍼티와 DOM 객체 사이의 관계



DOM 객체의 주요 공통 프로퍼티

	프로퍼티	설명
기본	id	태그의 id 속성 값
	lang	태그의 lang 속성 값
	style	style 객체에 대한 참조
	title	태그의 title 속성 값
	tagName	태그 이름
	innerHTML	시작 태그와 종료 태그 사이의 HTML 텍스트

DOM 객체의 주요 공통 프로퍼티

	프로퍼티	설명
DOM 트리	childElementCount	자식 DOM 객체 개수
	firstElementChild	첫 번째 자식 객체
	lastElementChild	마지막 자식 객체
	nextElementSibling	다음 형제 객체(오른쪽)
	parentElement	부모 객체
	previousElementSibling	이전 형제 객체(왼쪽)

DOM 객체의 주요 공통 프로퍼티

	프로퍼티	설명
크기와 위치	offsetHeight	전체 높이
	offsetWidth	전체 폭
	offsetLeft	객체의 출력 위치, 수평
	offsetTop	객체의 출력 위치, 수직

DOM 객체의 주요 공통 메서드

메서드	설명
addEventlistener()	새로운 이벤트 리스너 등록
appendChild()	마지막 자식 뒤에 새로운 자식 추가
click()	마우스를 클릭한 것과 동일한 작업 수행
focus()	키 입력을 받을 수 있도록 포커스 지정
setAttribute()	속성 추가
insertBefore()	지정된 자식 앞에 새 자식 추가
querySelector()	지정된 셀렉터와 일치하는 첫 번째 자식 리턴
removeChild()	자식 삭제
replaceChild()	자식 대체
removeEventListener()	등록된 이벤트 리스너 제거

document 객체

❖ **document**

- HTML 문서 전체를 대변하는 객체
  - 프로퍼티
    - HTML 문서의 전반적인 속성 내포
  - 메서드
    - DOM 객체 검색, DOM 객체 생성, HTML 문서 전반적 제어

- 모든 DOM 객체를 접근하는 경로의 시작점

- DOM 트리의 최상위 객체
  - 브라우저는 HTML 문서 로드 전, document 객체를 먼저 생성
  - document 객체를 뿌리로 하여 DOM 트리 생성

document 객체

❖ document 객체 접근 방법

window.document 또는 document

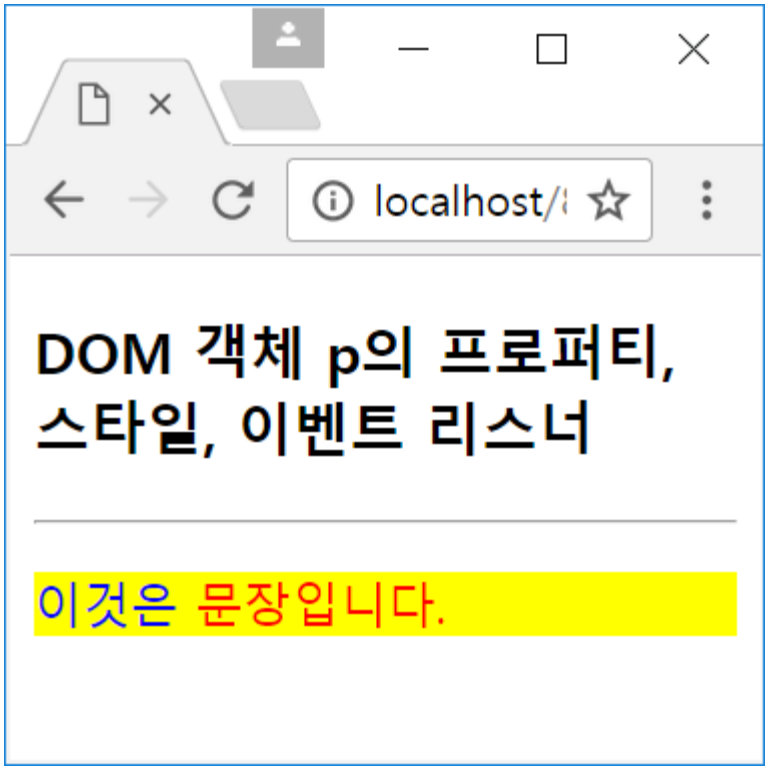
- document 객체는 DOM 객체가 아님
  - 연결된 스타일 시트가 없음

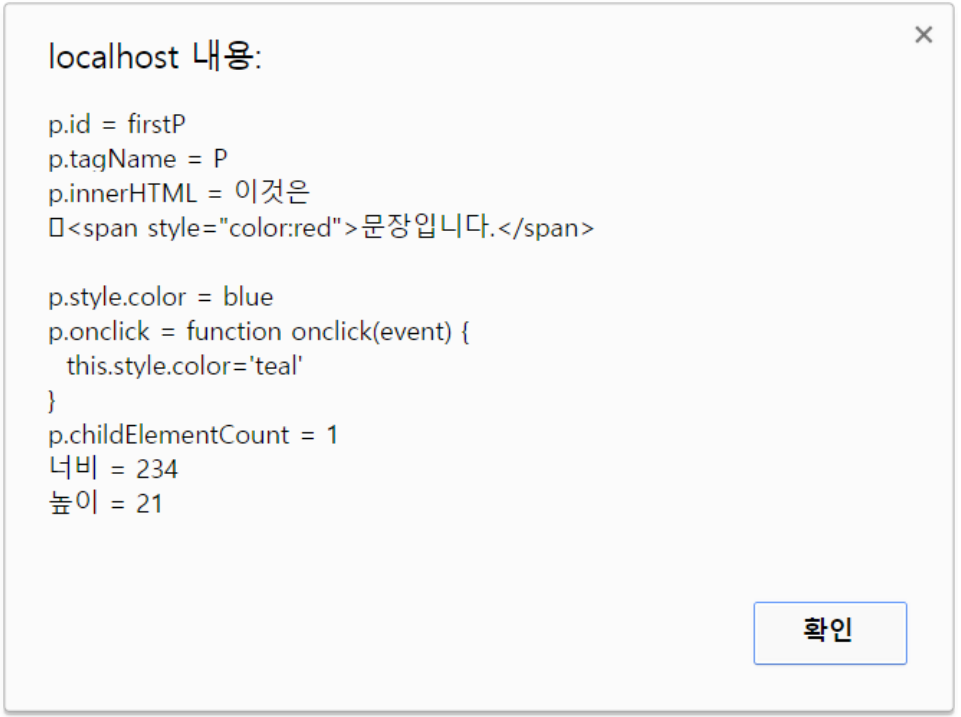
```
// 오류. document에는 CSS3 스타일 시트가 연결되지 않음
document.style.color = "red";
```

[실습] DOM 객체의 구조 출력 : p 객체

```
<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>
```

[실습] DOM 객체의 구조 출력 : p 객체





DOM 객체 다루기

❖ DOM 객체 구분, id 속성

```
<p id="firstP">안녕하세요</p>
```

- 페이지 내에 같은 HTML 태그가 여러 개 있는 경우, id 속성 값으로 구분
- id 속성 값은 HTML 페이지 내에서 유일한 값이어야 함
- 동일한 id 값이 있는 경우
  - 화면 출력에는 문제가 없음
  - id 값을 이용해서 DOM 객체를 찾는 경우, 먼저 나온 태그가 항상 찾아지게 되는 문제가 있음

DOM 객체 다루기

❖ id 값으로 DOM 객체 찾기

- document.getElementById()

```
// id 값이 firstP인 DOM 객체 리턴
var p = document.getElementById("firstP");

// p 객체의 글자 색을 red로 변경
p.style.color = "red";
```

❖ 다음과 같이 한 줄로 작성 가능

```
document.getElementById("firstP").style.color = “red”;
```

DOM 객체 다루기

❖ **DOM 객체의 CSS3 스타일 동적 변경**

- style 객체는 HTML 태그의 CSS3 스타일 시트 정보를 가짐
- style 객체를 이용하면 이미 출력된 HTML 태그의 모양 변경이 가능
- DOM 객체의 style 프로퍼티로 접근

- CSS3 스타일 프로퍼티는 다음과 같이 사용
  - background-color → backgroundColor
  - font-size → fontSize

DOM 객체 다루기

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

```
// id가 mySpan인 객체 찾기
var span = document.getElementById("mySpan");
```

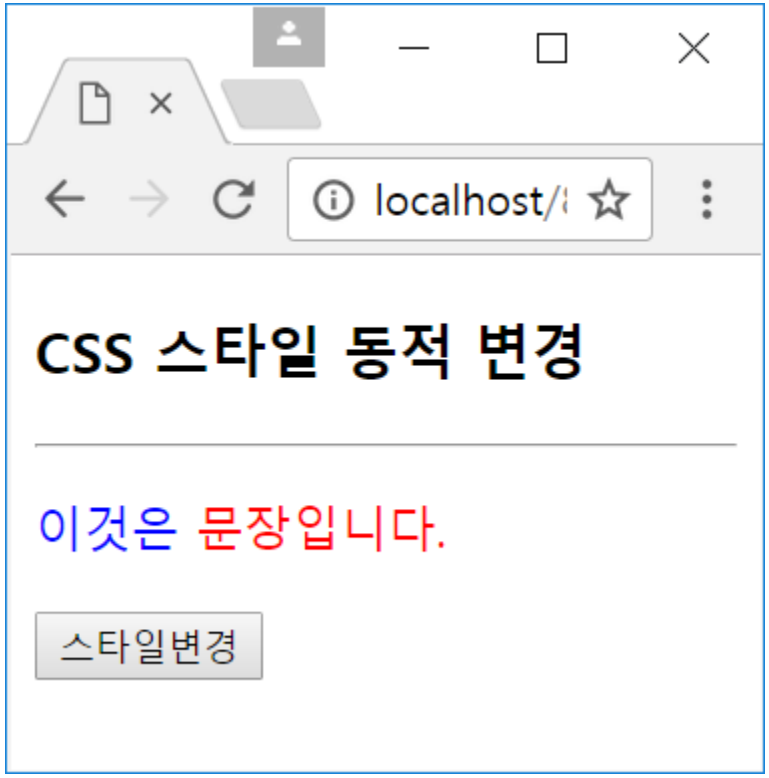
```
// '문장입니다'의 글자 색을 green으로 변경
span.style.color = "green";
```

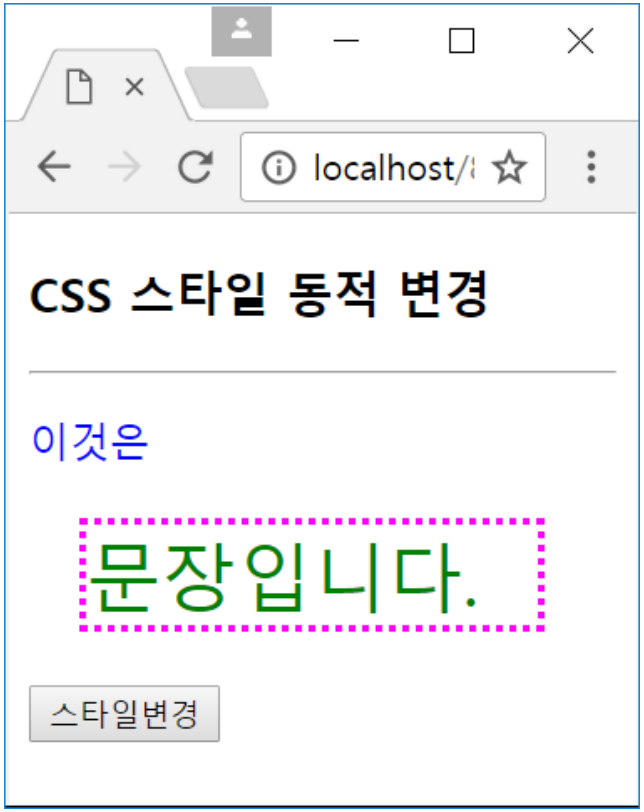
```
// '문장입니다'의 폰트를 30px 크기로 변경
span.style.fontSize = "30px";
```

[실습] 의 CSS3 스타일 동적 변경

```
<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";          // 박스 너비
    span.style.border = "3px dotted magenta"; // 3px점선 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>
```

[실습] 의 CSS3 스타일 동적 변경





innerHTML 프로퍼티

❖ innerHTML 프로퍼티

- 시작 태그와 종료 태그 사이에 들어있는 HTML 컨텐츠

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

innerHTML

- innerHTML 프로퍼티를 이용하여 사이의 HTML 텍스트를 읽을 수 있음

```
var p = document.getElementById("firstP");
var text = p.innerHTML;
```

innerHTML 프로퍼티

❖ innerHTML 프로퍼티

- innerHTML 프로퍼티 수정 → HTML 태그의 컨텐츠 변경

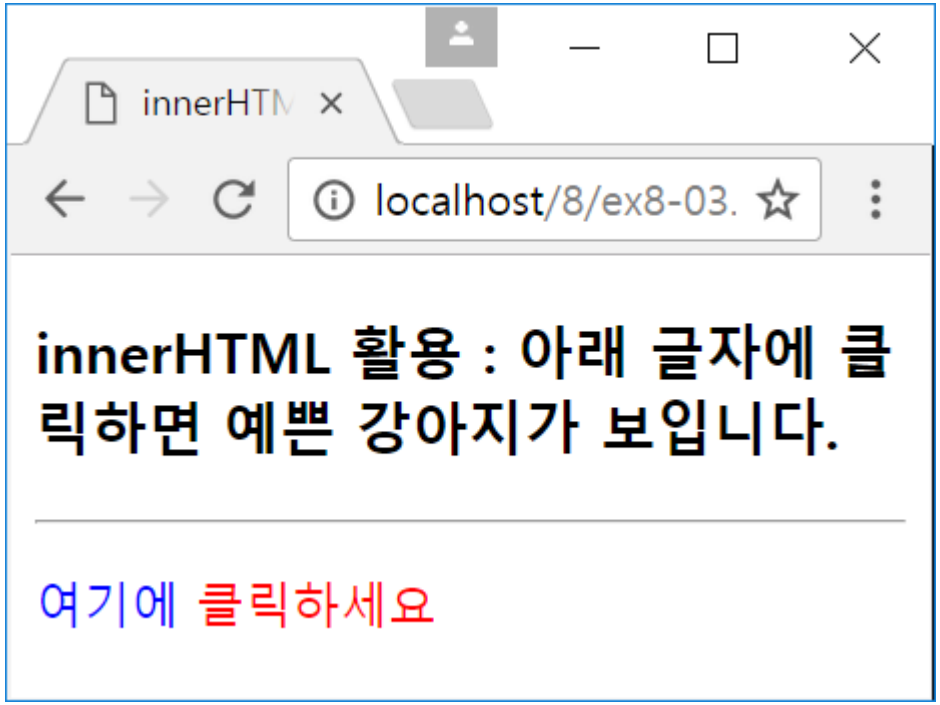
```
var p = document.getElementById("firstP");
p.innerHTML= "나의 <img src='puppy.jpg'>강아지입니다.";
```

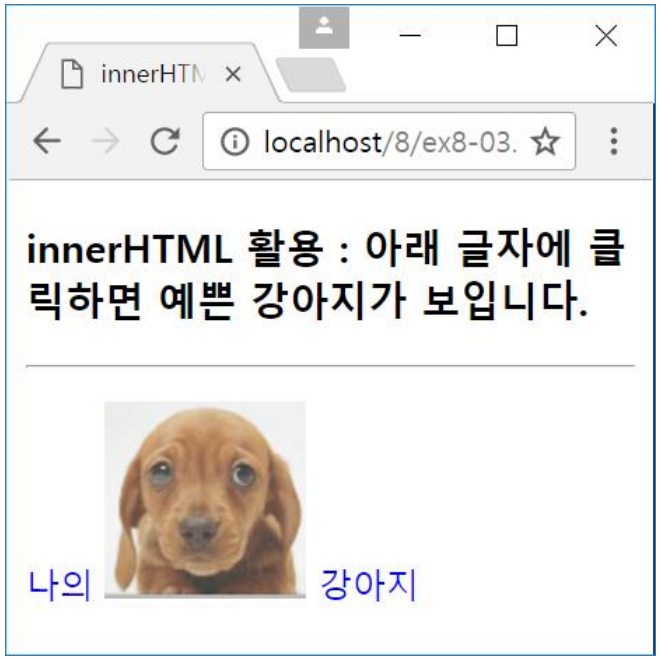
```
<p id="firstP" style="color:blue">
    나의 <img src='puppy.jpg'> 강아지입니다.
</p>
```

[실습] innerHTML을 이용하여 HTML 컨텐츠 변경

```
<body>
  <h3>innerHTML 활용 : 글자를 클릭하면 강아지가 보입니다.</h3>
  <hr>
  <p id="firstP" style="color:blue" onclick="change()">
     여기를  <span style="color:red">클릭하세요</span>
  </p>

  <script>
    function change() {
       var p = document.getElementById("firstP");
       p.innerHTML= "나의 <img src='puppy.png'> 강아지";
    }
  </script>

</body>
```

[실습] innerHTML을 이용하여 HTML 컨텐츠 변경





this

❖ **this 키워드**

- 객체 자신을 가리키는 자바스크립트 키워드
- DOM 객체에서 객체 자신을 가리키는 용도로 사용

- 태그 자신의 배경을 orange 색으로 변경

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

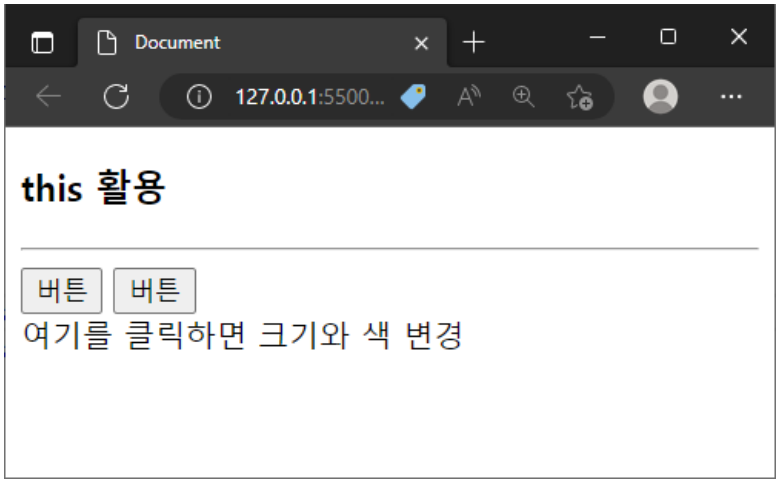
- 버튼이 클릭되면 자신의 배경색을 orange로 변경

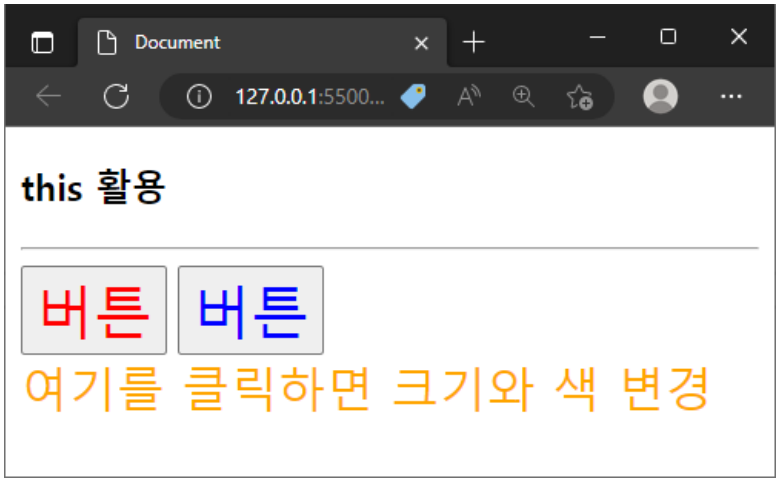
```
<button onclick="this.style.backgroundColor='orange'">
```

[실습] this 활용

```
<body>
  <script>
    function change(obj, size, color) {
      obj.style.color = color;
      obj.style.fontSize = size;
    }
  </script>

  <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')">
     여기를 클릭하면 크기와 색 변경
  </div>
</body>
```

[실습] this 활용


Document
127.0.0.1:5500...
this 활용
버튼
버튼
여기를 클릭하면 크기와 색 변경


Document
127.0.0.1:5500...
this 활용
버튼
버튼
여기를 클릭하면 크기와 색 변경

DOM 트리에서 DOM 객체 찾기

❖ **태그 이름으로 DOM 객체 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
```

DOM 트리에서 DOM 객체 찾기

❖ class 속성으로 DOM 객체 찾기

- document.getElementsByClassName()
  - 동일한 class 이름을 가진 모든 DOM 객체들을 찾아 컬렉션 리턴
  - 예)

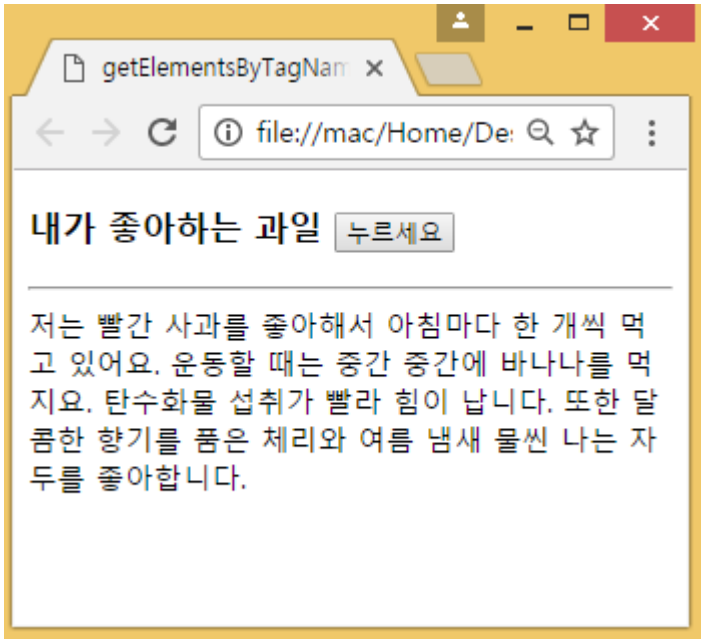
```
<div class="plain">...</div>
<div class="important">...</div>
<div class="plain">...</div>
```

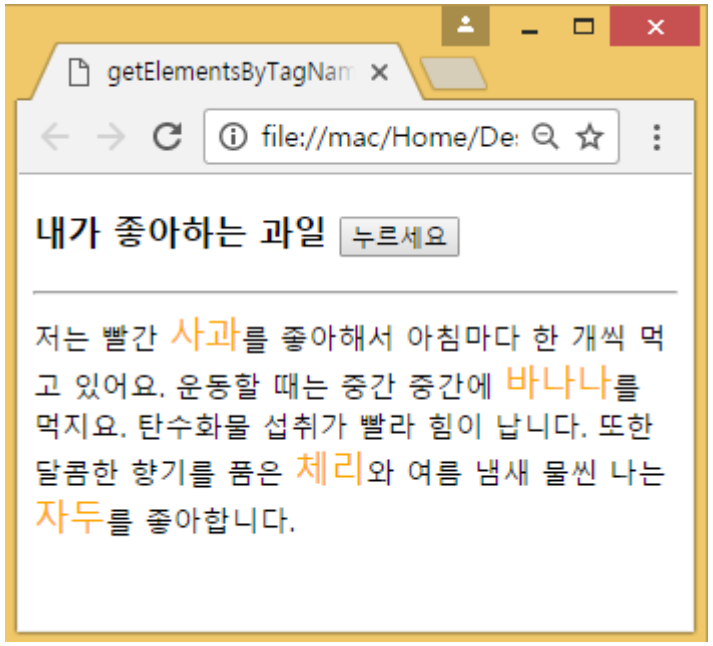
```
var plainClasses = document.getElementsByClassName("plain");

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

[실습] getElementsByTagName()

```
<head>
  <title> 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 = "orange";
        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>
```

[실습] getElementsByTagName()





DOM 트리에서 DOM 객체 찾기

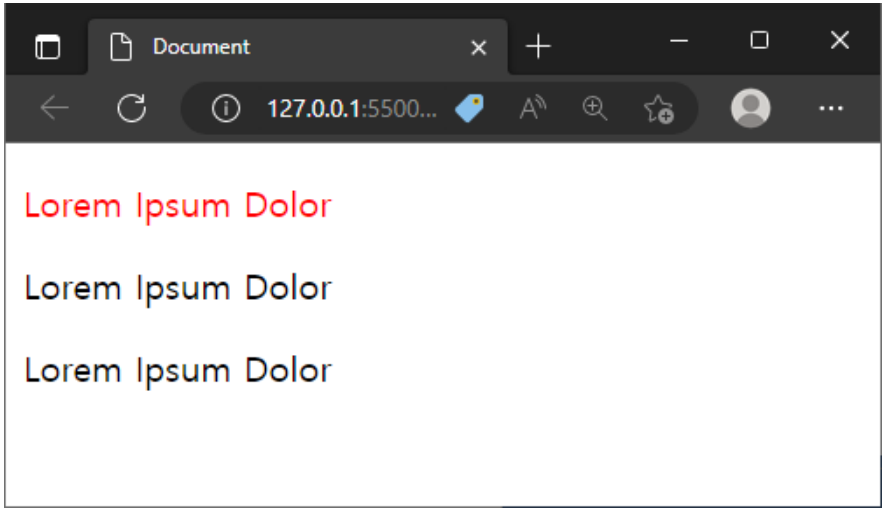
❖ css 선택자로 DOM 객체 찾기

- document.querySelector()
  - 특정 id, class, name으로 제한하지 않고, css 선택자를 사용하여 DOM 객체를 찾음
  - 선택자에 해당하는 첫 번째 요소만 리턴
  - 예)

```
// id 값으로 요소를 찾음
document.querySelector(#id)

// class 값으로 요소를 찾음
document.querySelector(.class)
```

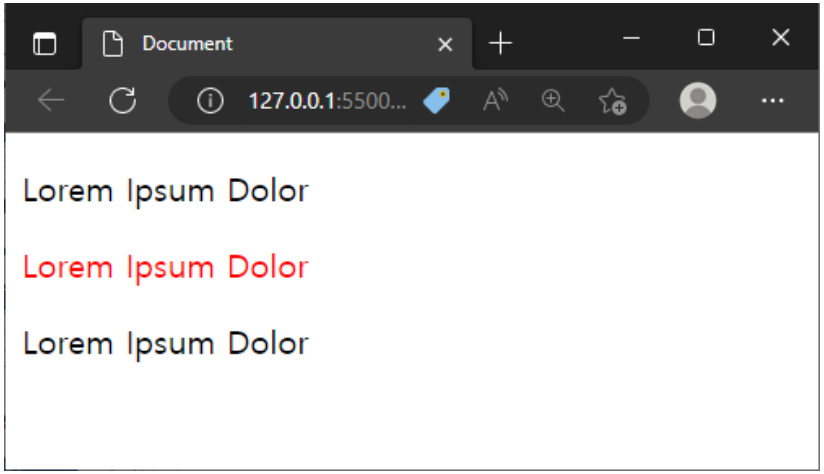
[실습] querySelector() - 1

```
<body>
    <p class="soo">Lorem Ipsum Dolor</p>
    <p class="soo">Lorem Ipsum Dolor</p>
    <p class="soo">Lorem Ipsum Dolor</p>

    <script>
      document.querySelector(".soo").style.color = "red";
    </script>
</body>
```



[실습] querySelector() - 2

```
<body>
    <p class="soo">Lorem Ipsum Dolor</p>
    <div>
      <p class="soo">Lorem Ipsum Dolor</p>
      <p class="soo">Lorem Ipsum Dolor</p>
    </div>

    <script>
      document.querySelector("div .soo").style.color = "red";
    </script>
</body>
```



DOM 트리에서 DOM 객체 찾기

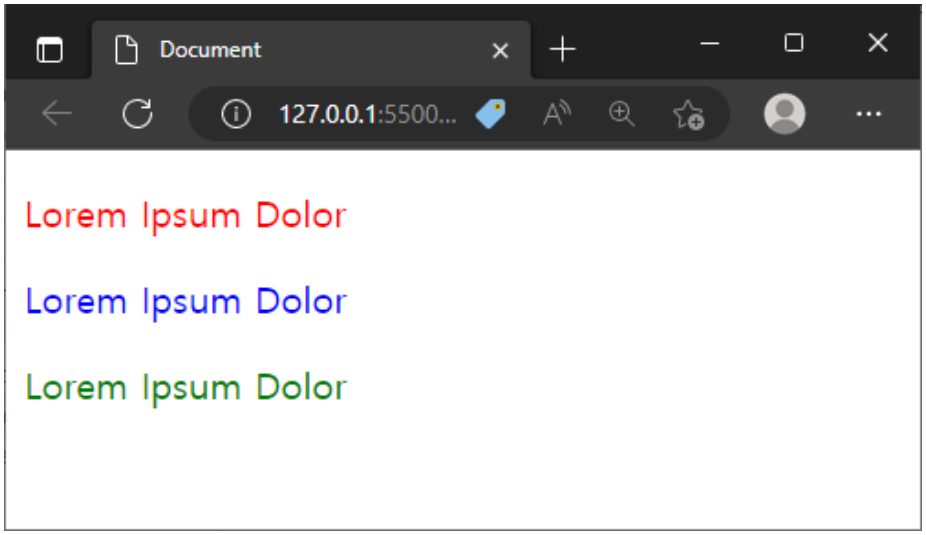
❖ css 선택자로 DOM 객체 찾기

- document.querySelectorAll()
  - 특정 id, class, name으로 제한하지 않고, css 선택자를 사용하여 DOM 객체를 찾음
  - 동일한 css 선택자를 가진 모든 DOM 객체들을 찾아 NodeList로 리턴
  - 콤마(,)를 사용하여 여러 DOM 객체를 한번에 가져올 수 있음

```
querySelectorAll("#id, .class");
```

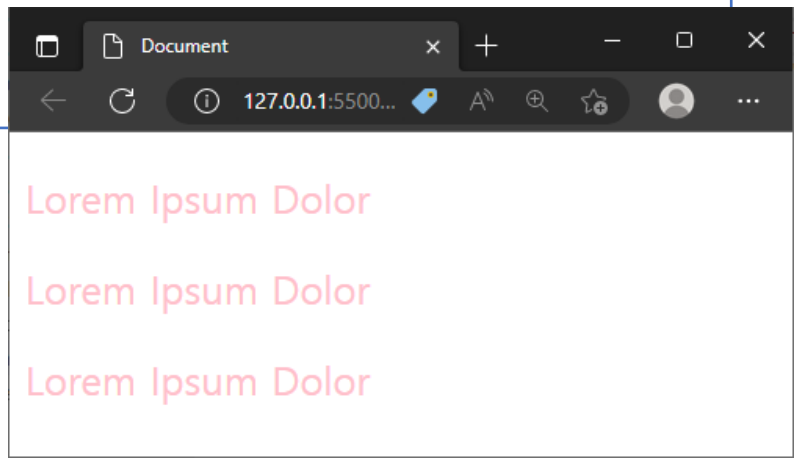
[실습] querySelectorAll() - 1

```
<body>
    <p class="soo">Lorem Ipsum Dolor</p>
    <p class="soo">Lorem Ipsum Dolor</p>
    <p class="soo">Lorem Ipsum Dolor</p>

    <script>
      var pList = document.querySelectorAll(".soo");
      pList[0].style.color = "red";
      pList[1].style.color = "blue";
      pList[2].style.color = "green";
    </script>
</body>
```



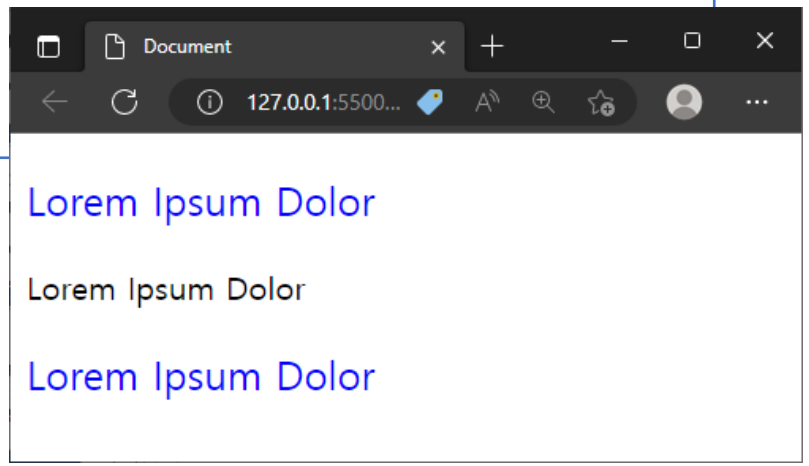
[실습] querySelectorAll() - 2

```
<body>
    <p class="soo">Lorem Ipsum Dolor</p>
    <p class="soo">Lorem Ipsum Dolor</p>
    <p class="soo">Lorem Ipsum Dolor</p>

    <script>
      var pList = document.querySelectorAll(".soo");

      for (var i = 0; i < pList.length; i++) {
        var item = pList[i];
        item.style.color = "pink";
        item.style.fontSize = "20px";
      }
    </script>
</body>
```



[실습] querySelectorAll() - 3

```
<body>
    <p class="soo">Lorem Ipsum Dolor</p>
    <p class="css">Lorem Ipsum Dolor</p>
    <p id="javascript">Lorem Ipsum Dolor</p>

    <script>
      var pList = document.querySelectorAll(".soo, #javascript");

      for (var i = 0; i < pList.length; i++) {
        var item = pList[i];
        item.style.color = "blue";
        item.style.fontSize = "20px";
      }
    </script>
</body>
```



문서의 동적 구성

❖ DOM 객체 동적 생성

- document.createElement(“태그이름”)
- 태그이름의 DOM 객체 생성

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

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

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

문서의 동적 구성

❖ DOM 트리에 삽입

```
// DOM 객체를 부모의 마지막 자식으로 삽입
부모.appendChild(DOM객체);

// DOM 객체를 부모의 자식 객체 중 기준 자식 앞에 삽입
부모.insertBefore(DOM객체, 기준자식);
```

- 태그를 <p id="p"> 태그의 마지막 자식으로 추가

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

문서의 동적 구성

❖ DOM 객체 삭제

```
var remove = 부모.removeChild(삭제하고자_하는_자식객체);
```

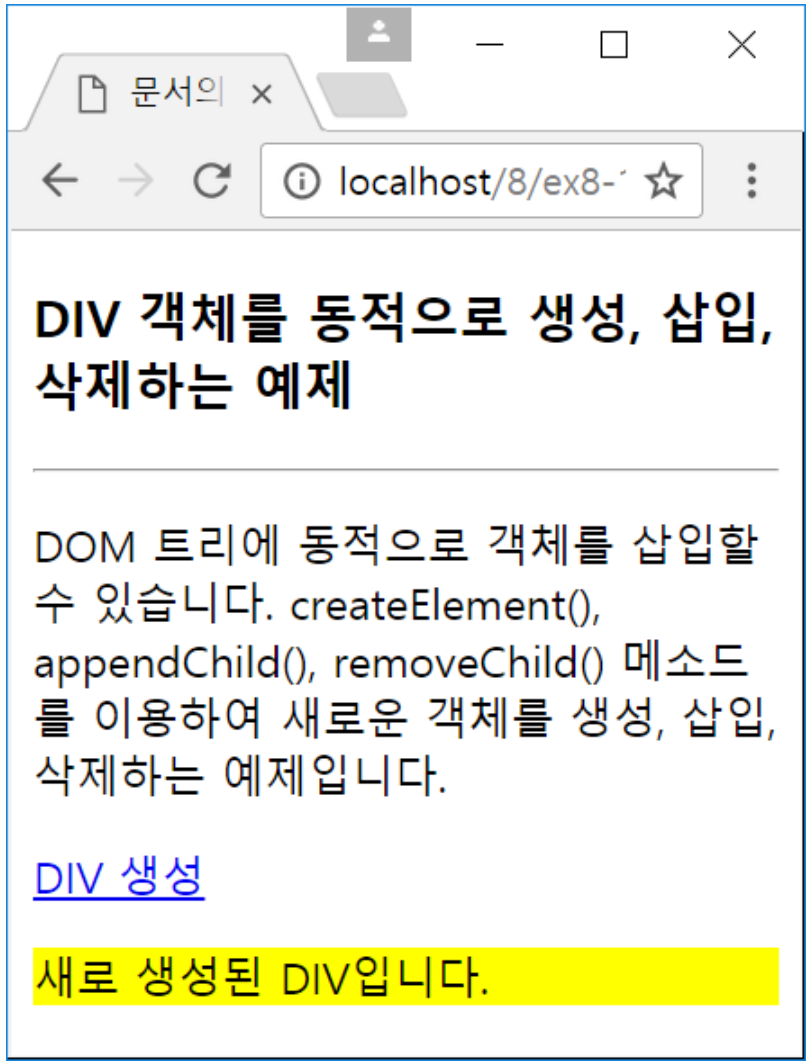
- “id=myDiv”인 DOM 객체를 DOM 트리에서 삭제

```
var myDiv = document.getElementById("myDiv");
var parent = myDiv.parentElement;

// 부모에서 myDiv 객체 삭제
parent.removeChild(myDiv);
```

[실습] HTML 태그의 동적 추가 및 삭제

```
<head>
   <title>문서의 동적 구성</title></head>
   <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 body = this.parentElement;       // 부모 HTML 태그 요소
         body.removeChild(this);              // 자신을 부모로부터 제거
         };
      obj.appendChild(newDIV);
   }
</script>
<body id="parent">
   <h3>DIV 객체를 동적으로 생성, 삽입, 삭제하는 예제</h3>
   <hr>
   <p>DOM 트리에 동적으로 객체를 삽입할 수 있습니다.
      createElement(), appendChild(), removeChild() 메서드를 이용하여 새로운 객체를 생성,
      삽입, 삭제하는 예제입니다.
   </p>
   <a href="javascript:createDIV()">DIV 생성</a>
</body>
```

[실습] HTML 태그의 동적 추가 및 삭제




THANK YOU