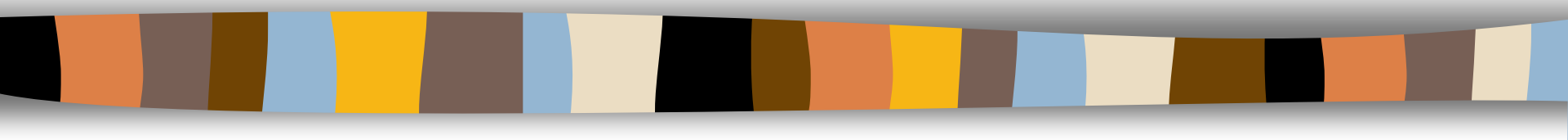
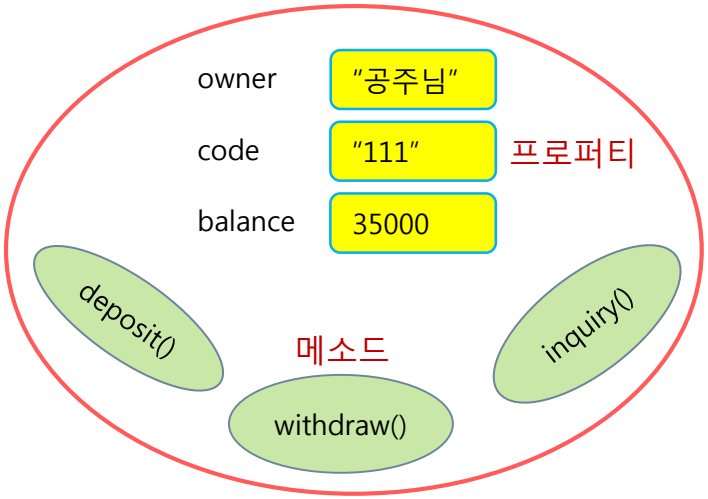
Javascript 객체



1. 코어, 사용자 객체
2. DOM 객체
3. BOM 객체

객체

- 정의
  - 실세계는 객체들과 그들간의 상호작용으로 운영
  - 객체는 프로퍼티(property)와 메소드로 구성
  - 프로퍼티 : 객체의 고유한 속성
  - 메소드(method) : 함수



자바스크립트 객체 account

```
var account = {
   owner     :  " 공주님",
   code      : "111",
   balance   : 35000,
   deposit   : function() { ... },
   withdraw  : function() { ... },
   inquiry   : function() { ... }
};
```

account 객체를 만드는 자바스크립트 코드

자바스크립트 객체 종류

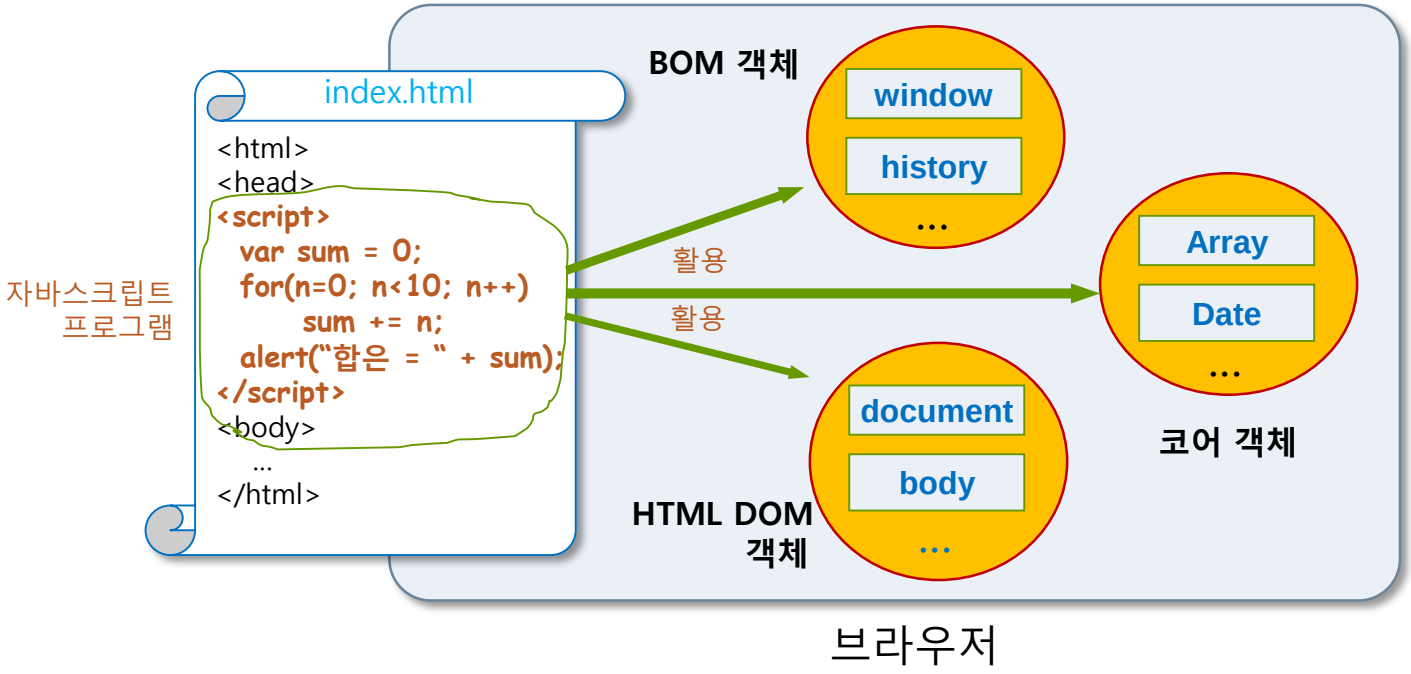
- 자바스크립트는 객체 기반 언어

- 자바스크립트 객체의 유형

1. 코어 객체
   - 자바스크립트 언어가 실행되는 어디서나 사용 가능한 기본 객체
   - 기본 객체로 표준 객체
   - Array, Date, String, Math 타입 등
   - 웹 페이지 자바스크립트 코드에서 혹은 서버에서 사용 가능
2. HTML DOM 객체
   - HTML 문서에 작성된 각 HTML 태그들을 객체화한 것들
   - HTML 문서의 내용과 모양을 제어하기 위한 목적
   - W3C의 표준 객체
3. 브라우저 객체
   - 자바스크립트로 브라우저를 제어하기 위해 제공되는 객체
   - BOM(Brower Object Model)에 따르는 객체들
   - 비표준 객체
4. 사용자 정의 객체

HTML 페이지와 자바스크립트 객체

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



Javascript 코어객체

코어 객체

- 코어 객체 종류
  - Array, Date, String, Math 등
- 코어 객체 생성
  - new 키워드 이용

```
var today = new Date();          // 시간 정보를 다루는 Date 타입의 객체 생성
var msg = new String("Hello"); // "Hello" 문자열을 담은 String 타입의 객체 생성
```

  - 객체가 생성되면 객체 내부에 프로퍼티와 메소드들 존재

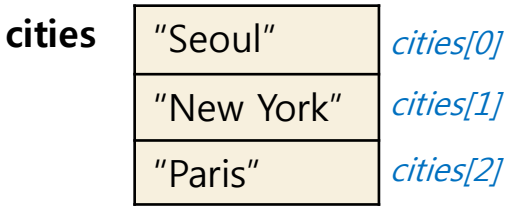
- 객체 접근
  - 객체와 멤버 사이에 점(.) 연산자 이용

```
obj.프로퍼티 = 값;              // 객체 obj의 프로퍼티 값 변경
변수 = obj.프로퍼티;            // 객체 obj의 프로퍼티 값 알아내기
obj.메소드(매개변수 값들);      // 객체 obj의 메소드 호출
```

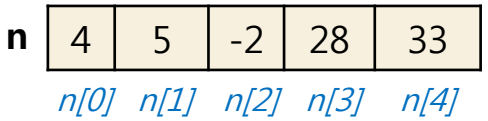
배열

- 배열
  - 여러 개의 원소들을 연속적으로 저장
  - 전체를 하나의 단위로 다루는 데이터 구조
- 배열 생성 사례

var **cities** = [“Seoul”, “New York”, “Paris”];



var **n** = [4, 5, -2, 28, 33];



  - 0에서 시작하는 인덱스를 이용하여 배열의 각 원소 접근

```
var name = cities[0];          // name은 “Seoul”
cities[1] = “Gainesville”;     // “New York” 자리에 “Gainesville” 저장
```

배열

- 배열 생성 방법
  - []로 배열 만들기

```
var week = [“월”, “화”, “수”, “목”, “금”, “토”, “일”];
var plots = [-20, -5, 0, 15, 20];
```

  - Array 객체로 배열 만들기

```
var week = new Array(“월”, “화”, “수”, “목”, “금”, “토”, “일”);
var week = new Array(7);
```

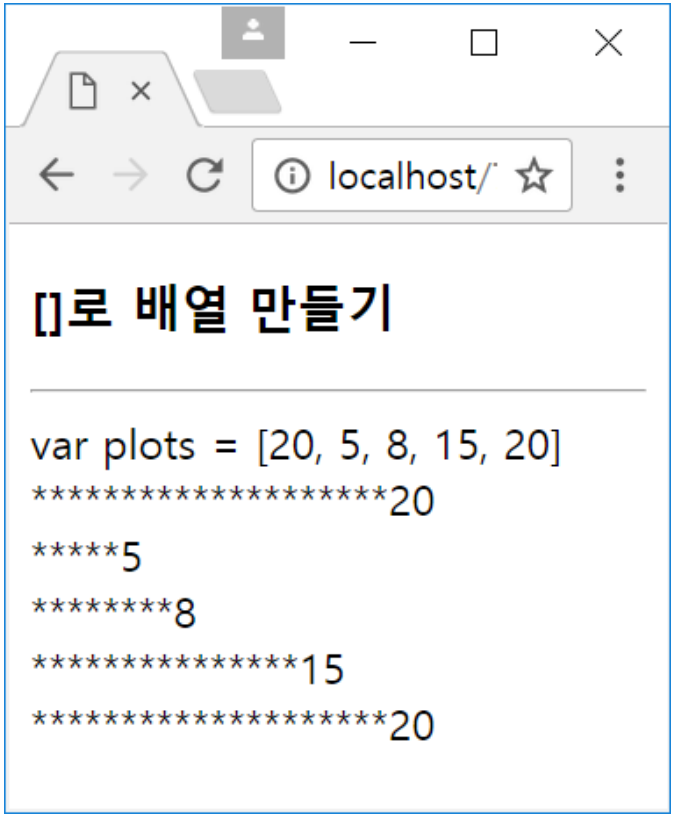
- 배열 사용
  - 배열의 크기는 고정되지 않고, 마지막 원소 추가 시 늘어남

```
plots[5] = 33; // plots 배열에 6번째 원소 추가. 배열 크기는 6이 됨
plots[6] = 22; // plots 배열에 7번째 원소 추가. 배열 크기는 7이 됨
```

  - length 프로퍼티 사용
    - var n = **week.length**;
  - 서로 다른 타입의 자료도 가능
  - 첨자는 0부터

예제1


```
<!DOCTYPE html>
<html><head><title>[]로 배열 만들기</title></head>
<body>
<h3>[]로 배열 만들기</h3>
<hr>
<script>
   var plots = [20, 5, 8, 15, 20]; // 원소 5개의 배열 생성
   document.write("var plots = [20, 5, 8, 15, 20]<br>");

   for(i=0; i<5; i++) {
      var size = plots[i]; // plots 배열의 i번째 원소
      while(size>0) {
         document.write("*");
         size--;
      }
      document.write(plots[i] + "<br>");
   }
</script>
</body>
</html>
```

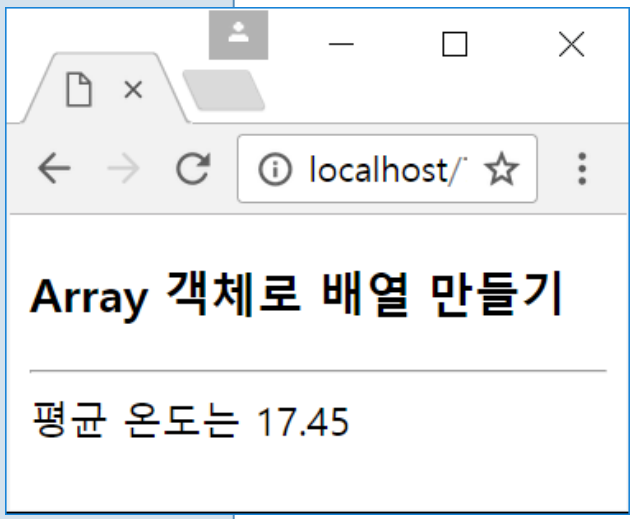


예제2


```
<!DOCTYPE html>
<html><head><title>Array 객체로 배열 만들기</title></head>
<body>
<h3>Array 객체로 배열 만들기</h3>
<hr>
<script>
   var degrees = new Array(); // 빈 배열 생성
   degrees[0] = 15.1;
   degrees[1] = 15.4;
   degrees[2] = 16.1;
   degrees[3] = 17.5;
   degrees[4] = 19.2;
   degrees[5] = 21.4;

   var sum = 0;
   for(i=0; i<degrees.length; i++)
      sum += degrees[i];

   document.write("평균 온도는 " + sum/degrees.length + "<br>");
</script>
</body>
</html>
```

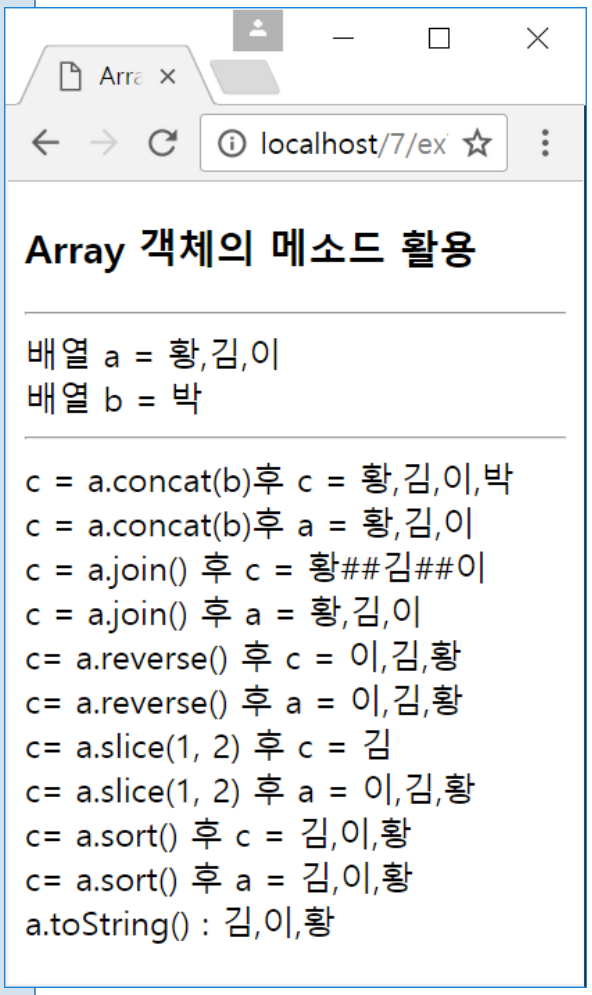
배열 크기만큼 루프

배열 degrees의 크기, 6



예제3 Array 객체의 메소드 활용


```
<!DOCTYPE html>
<html><head><title>Array 객체의 메소드 활용</title>
<script>
    function pr(msg, arr) { document.write(msg + arr.toString() + "<br>"); }
</script>
</head>
<body>
<h3>Array 객체의 메소드 활용</h3>
<hr>
<script>
    var a = new Array("황", "김", "이");
    var b = new Array("박");
    var c;

    pr("배열 a = ", a);
    pr("배열 b = ", b);
    document.write("<hr>");

    c = a.concat(b); // c는 a와 b를 연결한 새 배열
    pr("c = a.concat(b)후 c = ", c);
    pr("c = a.concat(b)후 a = ", a);

    c = a.join("##"); // c는 배열 a를 연결한 문자열
    pr("c = a.join() 후 c = ", c);
    pr("c = a.join() 후 a = ", a);

    c = a.reverse(); // a.reverse()로 a 자체 변경. c는 배열
    pr("c= a.reverse() 후 c = ", c);
    pr("c= a.reverse() 후 a = ", a);

    c = a.slice(1, 2); // c는 새 배열
    pr("c= a.slice(1, 2) 후 c = ", c);
    pr("c= a.slice(1, 2) 후 a = ", a);

    c = a.sort(); // a.sort()는 a 자체 변경. c는 배열
    pr("c= a.sort() 후 c = ", c);
    pr("c= a.sort() 후 a = ", a);

    c = a.toString(); // toString()은 원소 사이에 ","를 넣어  문자열 생성
    document.write("a.toString() : " + c); // c 는 문자열
</script></body></html>
```



Date 객체

- Date 객체
  - 시간 정보를 담는 객체
  - 현재 시간 정보

```
var now = new Date(); // 현재 날짜와 시간(시, 분, 초) 값으로 초기화된 객체 생성
```

  - 학기 시작일 2017년 3월 1일의 날짜 기억

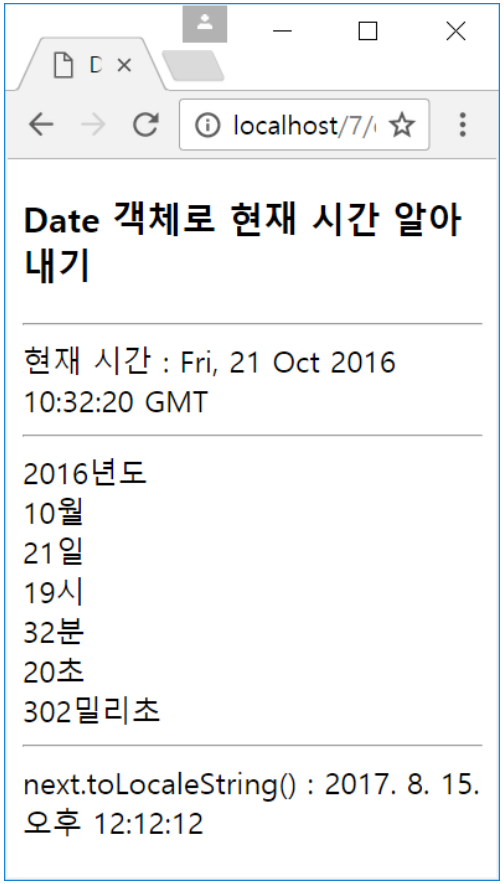
```
var startDay = new Date(2017, 2, 1);      // 2017년 3월 1일(2는 3월을 뜻함)
```

- Date 객체 활용

```
var now = new Date();          // 현재 2017년 5월 15일 저녁 8시 48분이라면
var date = now.getDate();      // 오늘 날짜. date = 15
var hour = now.getHours();     // 지금 시간. hour = 20
```

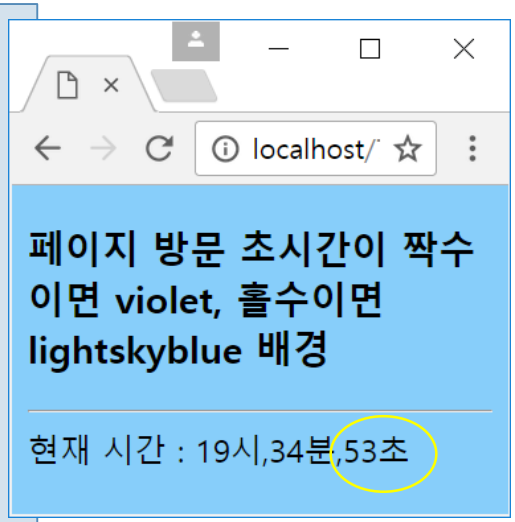
예제4

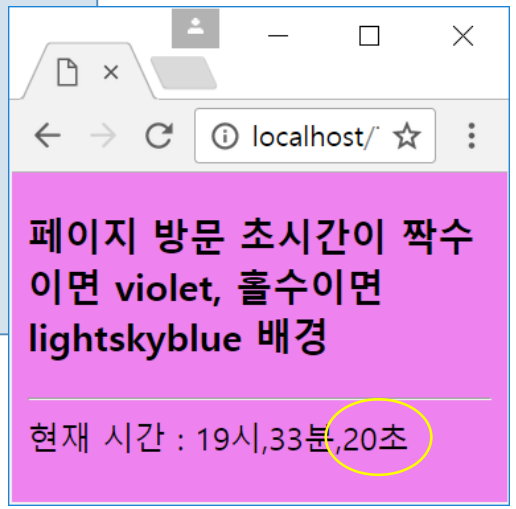

```
<!DOCTYPE html>
<html>
<head>
<title>Date 객체로 현재 시간 알아내기</title>
</head>
<body>
<h3>Date 객체로 현재 시간 알아내기</h3>
<hr>
<script>
var now = new Date(); // 현재 시간 값을 가진 Date 객체 생성
document.write("현재 시간 : " + now.toUTCString()
            + "<br><hr>");
document.write(now.getFullYear() + "년도<br>");
document.write(now.getMonth() + 1 + "월<br>");
document.write(now.getDate() + "일<br>");
document.write(now.getHours() + "시<br>");
document.write(now.getMinutes() + "분<br>");
document.write(now.getSeconds() + "초<br>");
document.write(now.getMilliseconds() + "밀리초<br><hr>");

var next = new Date(2017, 7, 15, 12, 12, 12); // 7은 8월
document.write("next.toLocaleString() : "
            + next.toLocaleString() + "<br>");
</script>
</body>
</html>
```



예제5


```
<!DOCTYPE html>
<html>
<head>
<title>방문 시간에 따라 변하는 배경색</title>
</head>
<body>
<h3>페이지 방문 초시간이  짝수이면 violet, 홀수이면 lightskyblue 배경</h3>
<hr>
<script>
   var current = new Date(); // 현재 시간을 가진 Date 객체 생성
   if(current.getSeconds() % 2 == 0)
      document.body.style.backgroundColor = "violet";
   else
      document.body.style.backgroundColor = "lightskyblue";

   document.write("현재 시간 : ");
   document.write(current.getHours(), "시,");
   document.write(current.getMinutes(), "분,");
   document.write(current.getSeconds(), "초<br>");
</script>
</body>
</html>
```

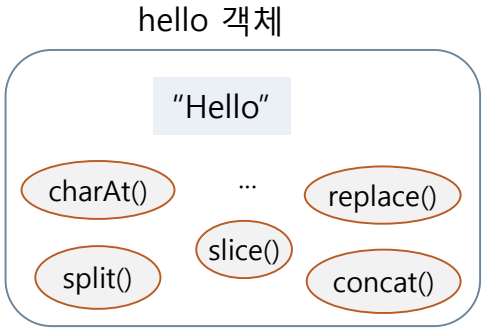




String 객체

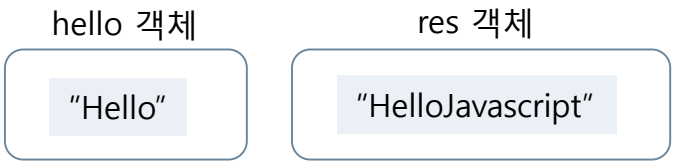
- String
  - 문자열을 담기 위한 객체

```
// 2 경우 모두 오른쪽 String 객체 생성

var hello = new String("Hello");
var hello = "Hello";
```



  - String 객체는 일단 생성되면 수정 불가능

```
var hello = new String("Hello");
var res = hello.concat("Javascript");

// concat() 후 hello의 문자열 변화 없음
```



String 객체의 특징

- 문자열 길이
  - String 객체의 length 프로퍼티 : 읽기 전용

```
var hello = new String("Hello");
var every = "Boy and Girl";
var m = hello.length;              // m은 5
var n = every.length;              // n은 12
```

```
var n = "Thank you".length;    // n은 9
```

- 문자열을 배열처럼 사용
  - [] 연산자를 사용하여 각 문자 접근

```
var hello = new String("Hello");
var c = hello[0];      // c = "H". 문자 H가 아니라 문자열 "H"
```

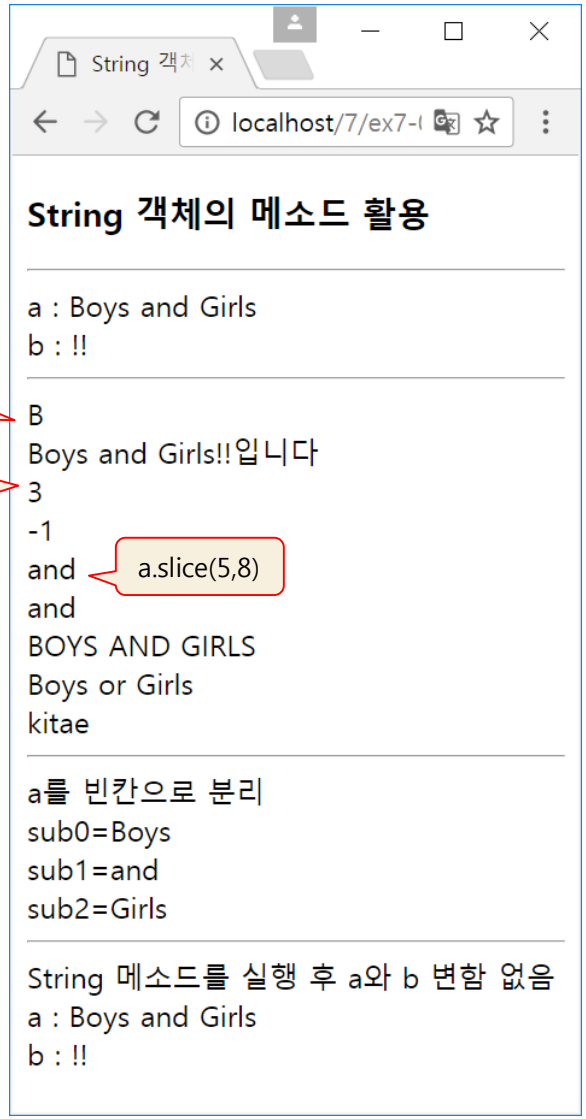
예제: String 객체의 메소드 활용


```
<!DOCTYPE html>
<html><head><title>String 객체의 메소드 활용</title></head>
<body>
<h3>String 객체의 메소드 활용</h3>
<hr>
<script>
var a = new String("Boys and Girls");
var b = "!!";
document.write("a : " + a + "<br>");
document.write("b : " + b + "<br><hr>");

document.write(a.charAt(0) + "<br>");
document.write(a.concat(b, "입니다") + "<br>");
document.write(a.indexOf("s") + "<br>");
document.write(a.indexOf("And") + "<br>");
document.write(a.slice(5, 8) + "<br>");
document.write(a.substr(5, 3) + "<br>");
document.write(a.toUpperCase() + "<br>");
document.write(a.replace("and", "or") + "<br>");
document.write("    kitae    ".trim() + "<br><hr>");

var sub = a.split(" ");
document.write("a를 빈칸으로 분리<br>");
for(var i=0; i<sub.length; i++)
    document.write("sub" + i + "=" + sub[i] + "<br>");

document.write("<hr>String 메소드를 실행 후 a와 b 변함 없음<br>");
document.write("a : " + a + "<br>");
document.write("b : " + b + "<br>");
</script>
</body>
</html>
```

a.charAt(0)

a.indexOf("s")

a.slice(5,8)



Math 객체

- Math
  - 수학 계산을 위한 프로퍼티와 메소드 제공
  - new Math()로 객체 생성하지 않고 사용

```
var sq = Math.sqrt(4);          // 4의 제곱근을 구하면 2
var area = Math.PI*2*2;         // 반지름이 2인 원의 면적
```

  - 난수 발생
    - Math.random() : 0~1보다 작은 랜덤한 실수 리턴
    - Math.floor(m)은 m의 소수점 이하를 제거한 정수 리턴

```
// 0~99까지 랜덤한 정수를 10개 만드는 코드
for(i=0; i<10; i++) {
   var m = Math.random()*100;  // m은 0~99.999... 보다 작은 실수 난수
   var n = Math.floor(m);          // m에서 소수점 이하를 제거한 정수(0~99사이)
   document.write(n + " ");
}
```

실습

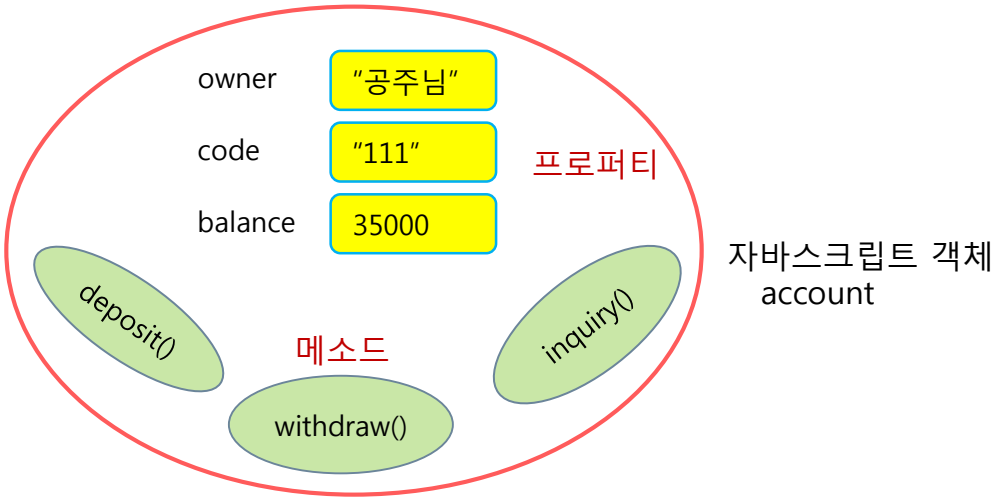
- 다음을 해결하시오
  - 난수를 발생하여 100보다 작은 정수를 얻는다
  - 그 수가 소수인지를 가려라

Javascript 사용자객체

사용자 객체 만들기

- 사용자가 새로운 타입의 객체 작성 가능
- 새로운 타입의 객체 만드는 2 가지 방법만 소개
  - new Object() 이용
  - 리터럴 표기법 이용

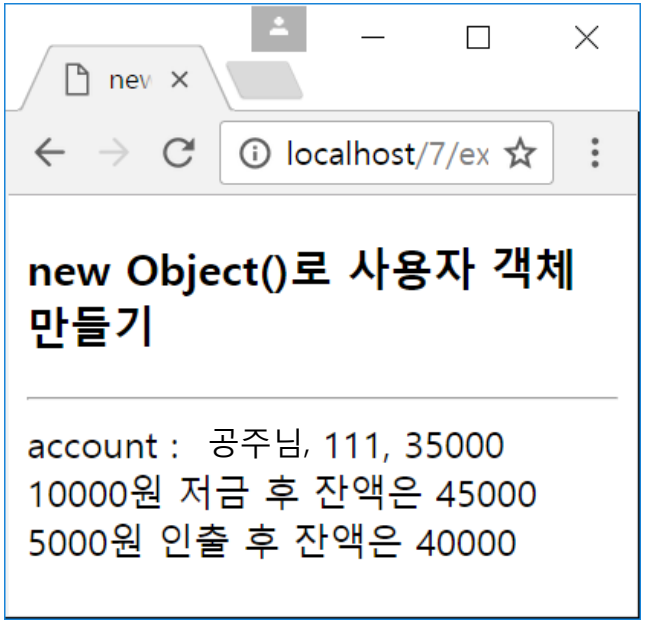
- 은행 계좌를 표현하는 account 객체



예제: new Object()로 객체 만들기

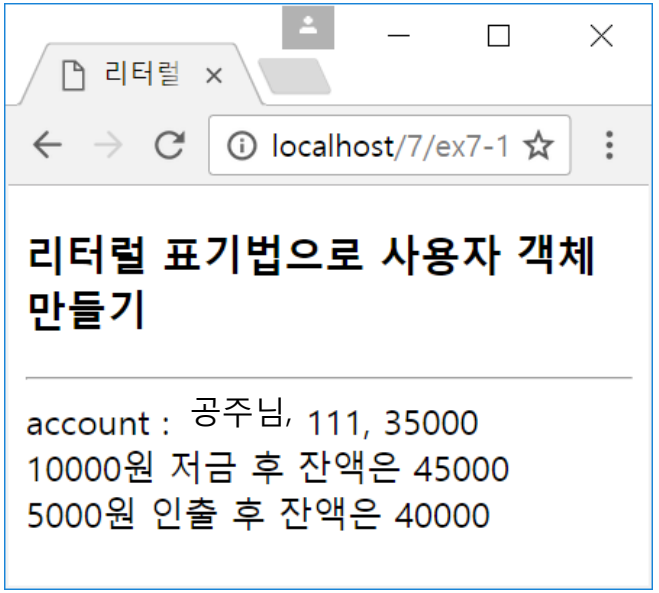

```
<!DOCTYPE html>
<html><head><title>new Object()로 사용자 객체 만들기</title>
<script>
    //메소드로 사용할 3 개의 함수 작성
    function inquiry() { return this.balance; } // 잔금 조회
    function deposit(money) { this.balance += money; } // money 만큼 저금
    function withdraw(money) { // 예금 인출, money는 인출하고자 하는 액수
                                    // money가 balance보다 작다고 가정
        this.balance -= money;
        return money;
    }

    // 사용자 객체 만들기
    var account  = new Object();
    account.owner = " 공주님"; // 계좌 주인 프로퍼티 생성 및 초기화
    account.code = "111"; // 코드 프로퍼티 생성 및 초기화
    account.balance = 35000; // 잔액 프로퍼티 생성 및 초기화
    account.inquiry = inquiry; // 메소드 작성
    account.deposit = deposit; // 메소드 작성
    account.withdraw = withdraw; // 메소드 작성
</script></head>
<body>
<h3>new Object()로 사용자 객체 만들기</h3>
<hr>
<script>
    // 객체 활용
    document.write("account : ");
    document.write(account.owner + ", ");
    document.write(account.code + ", ");
    document.write(account.balance + "<br>");

    account.deposit(10000); // 10000원 저금
    document.write("10000원 저금 후 잔액은 " + account.inquiry() + "<br>");
    account.withdraw(5000); // 5000원 인출
    document.write("5000원 인출 후 잔액은 " + account.inquiry() + "<br>");
</script>
</body></html>
```

this.balance는 객체의 balance 프로퍼티

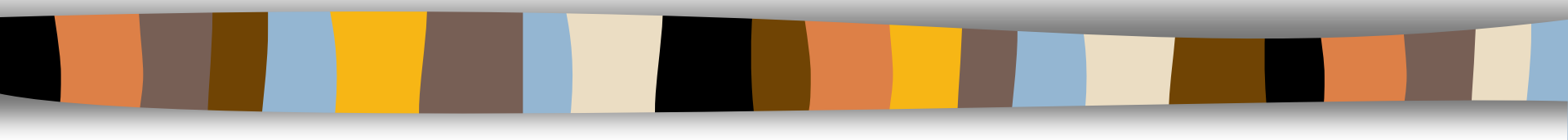


예제: 리터럴 표기법으로 객체 만들기


```
<!DOCTYPE html>
<html>
<head><title>리터럴 표기법으로 사용자 객체 만들기</title>
<script>
//사용자 객체 만들기
var account = {
    // 프로퍼티 생성 및 초기화
    owner : " 공주님 ", // 계좌 주인
    code : "111", // 계좌 코드
    balance : 35000, // 잔액 프로퍼티

    // 메소드 작성
    inquiry : function () { return this.balance; }, // 잔금 조회
    deposit : function(money) { this.balance += money; }, // 저금. money 만큼 저금
    withdraw : function (money) { // 예금 인출, money는 인출하고자 하는 액수
                                  // money가 balance보다 작다고 가정
        this.balance -= money;
        return money;
    }
};
</script></head>
<body>
<h3>리터럴 표기법으로 사용자 객체 만들기</h3>
<hr>
<script>
    document.write("account : ");
    document.write(account.owner + ", ");
    document.write(account.code + ", ");
    document.write(account.balance + "<br>");

    account.deposit(10000); // 10000원 저금
    document.write("10000원 저금 후 잔액은 " + account.inquiry() + "<br>");
    account.withdraw(5000); // 5000원 인출
    document.write("5000원 인출 후 잔액은 " + account.inquiry() + "<br>");
</script>
</body></html>
```

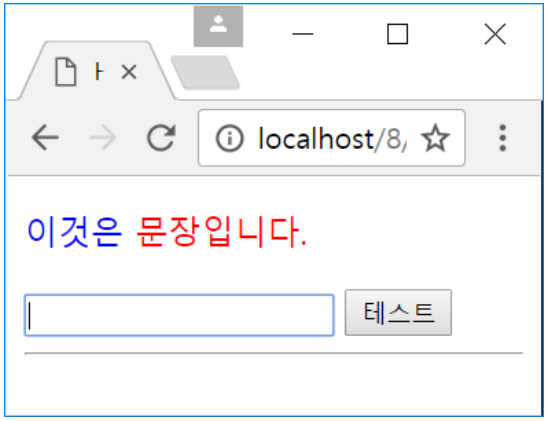


Javascript DOM객체

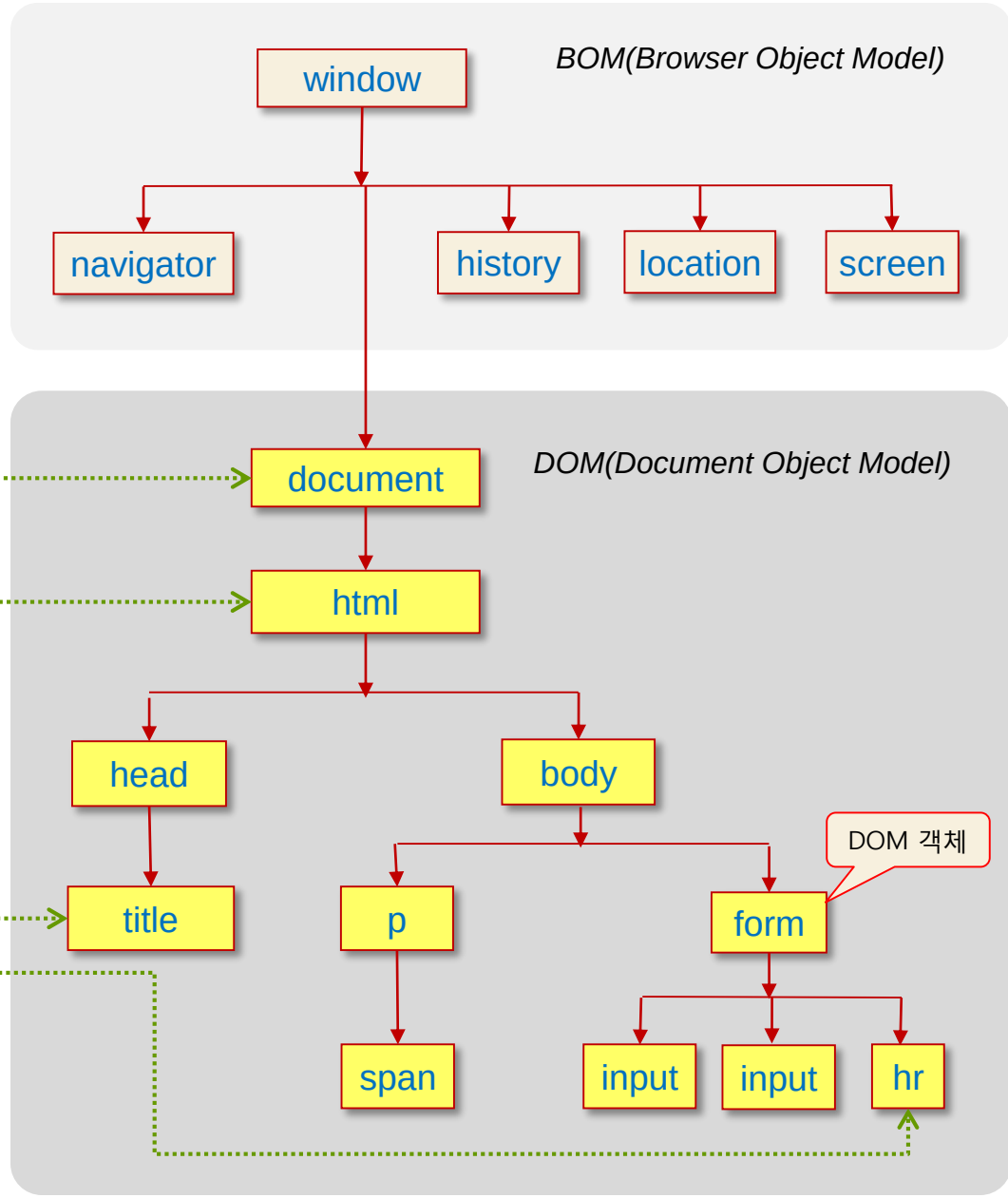


HTML DOM(Document Object Model)

- 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
  - 부모 자식 관계 포함
- DOM 객체
  - DOM 트리의 한 노드
  - HTML 태그 당 하나의 DOM 객체 생성
    - DOM 노드(Node), DOM 엘리먼트(Element) 라고도 불림




```
<!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>
```



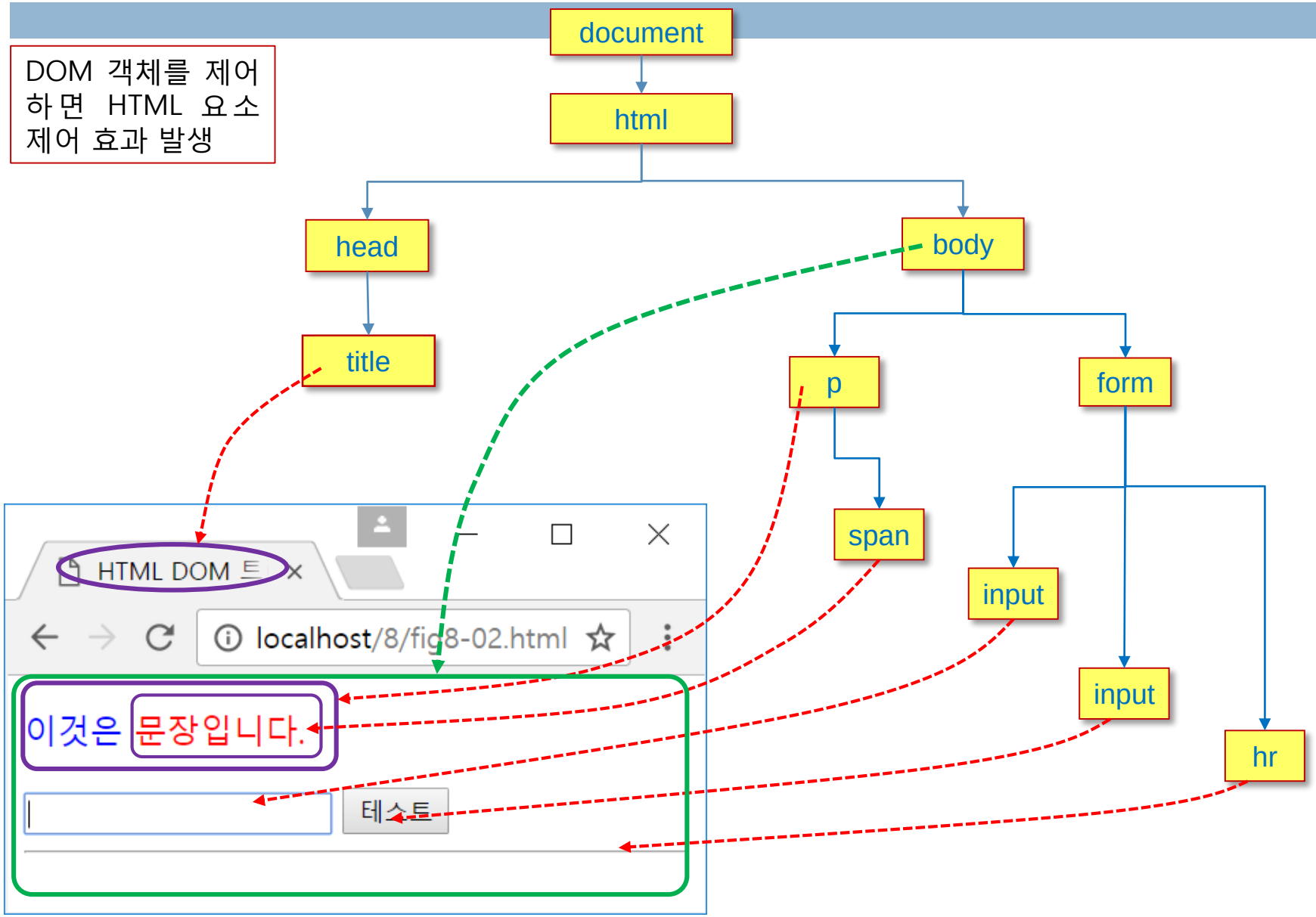
DOM 트리의 특징

- DOM 트리의 특징
  - DOM 트리의 루트는 document 객체
  - HTML 엘레먼트 당 DOM 객체가 하나씩 생성
  - HTML 태그의 포함관계에 따라 부모 자식 관계 형성

- 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 출력 바로 갱신

DOM 객체와 HTML 페이지의 화면 출력

DOM 객체를 제어하면 HTML 요소 제어 효과 발생



HTML 엘리먼트의 구성요소

- HTML 엘리먼트(element)
  - 구성요소
    - 엘리먼트 이름
    - 속성
    - CSS3 스타일
    - 이벤트 리스너
    - 콘텐츠(innerHTML)

DOM 객체의 구성 요소

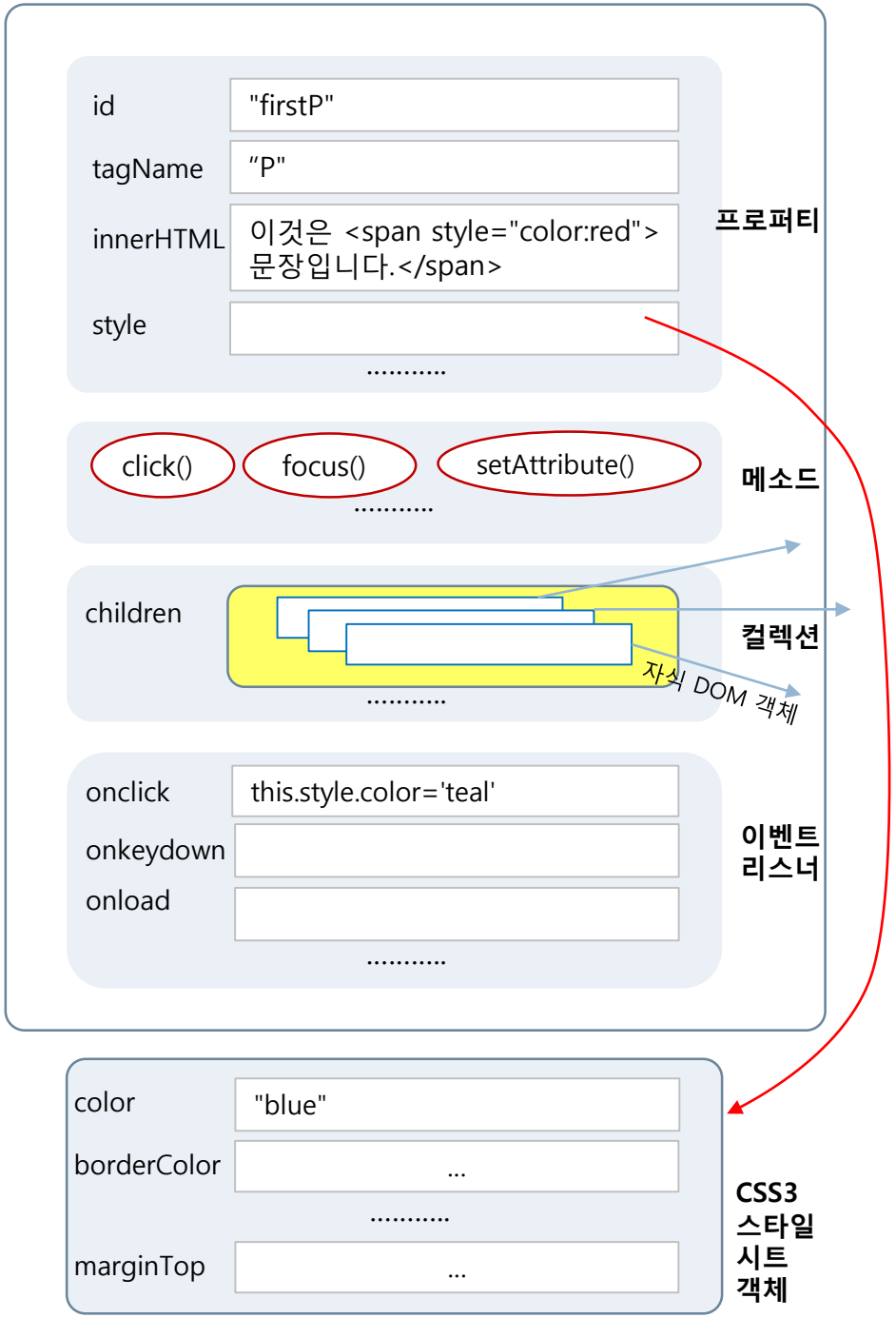
- DOM 객체 구성요소
  - 프로퍼티(property)
    - HTML 태그의 속성(attribute) 반영
  - 메소드(method)
    - DOM 객체의 멤버 함수로서, HTML 태그 제어 가능
  - 컬렉션(collection)
    - 자식 DOM 객체들의 주소를 가지는 등 배열과 비슷한 집합적 정보
  - 이벤트 리스너(event listener)
    - HTML 태그에 작성된 이벤트 리스너 반영
    - 약 70여개의 이벤트 리스너를 가질 수 있음
  - CSS3 스타일
    - HTML 태그에 설정된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

DOM 객체의 구성

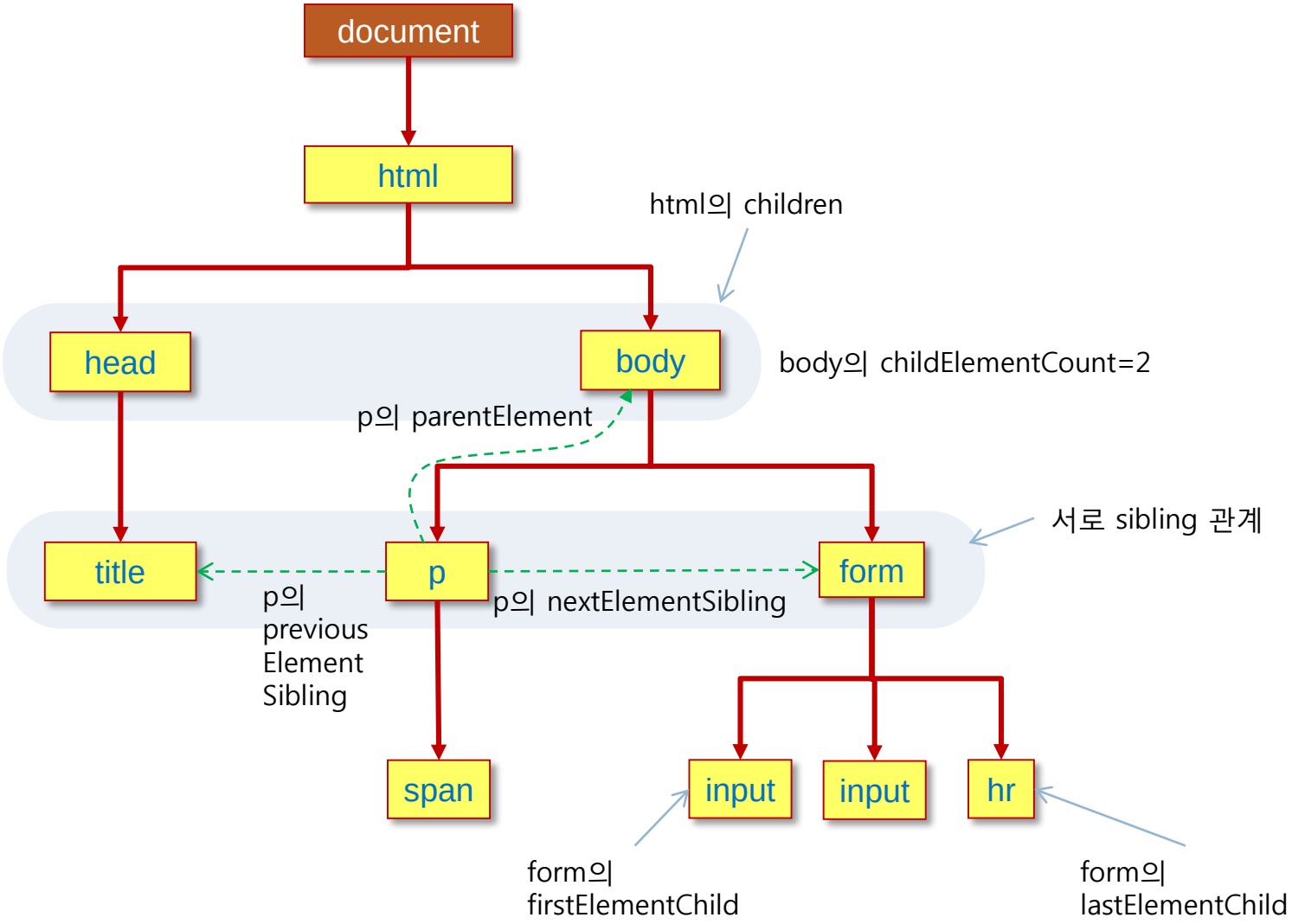
- 프로퍼티(property)
- 메소드(method)
- 컬렉션(collection)
- 이벤트 리스너(event listener)
- CSS3 스타일

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

** ... 태그**



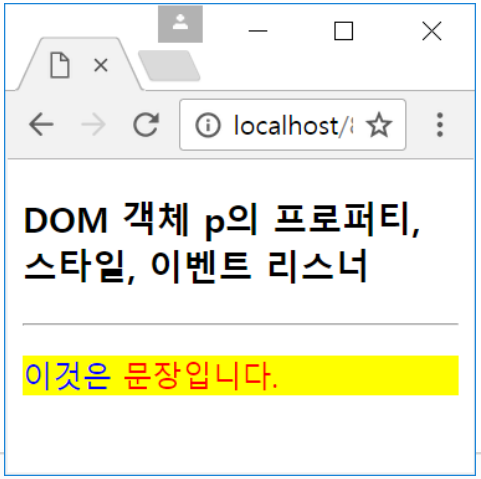
DOM 객체의 프로퍼티와 객체와의 관계

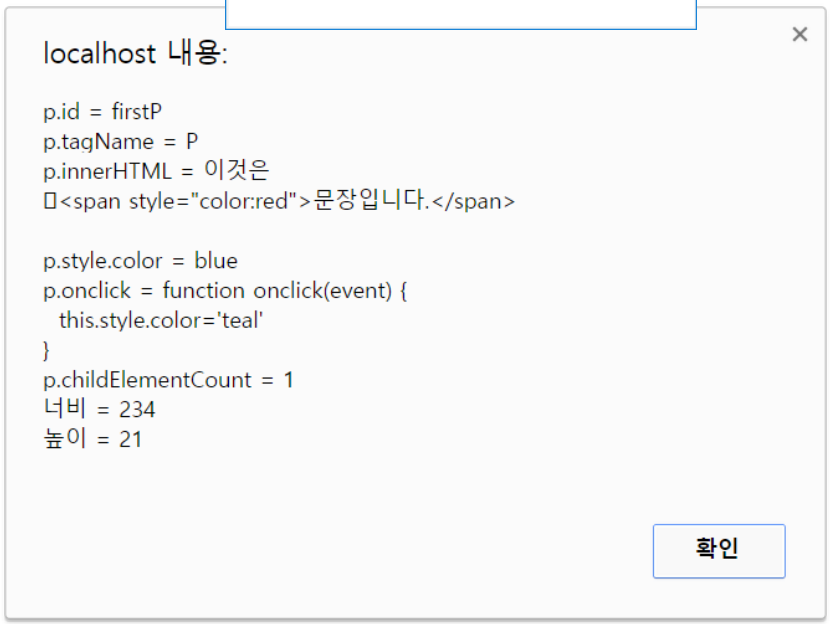


예제: DOM 객체의 구조 출력


```
<!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
```

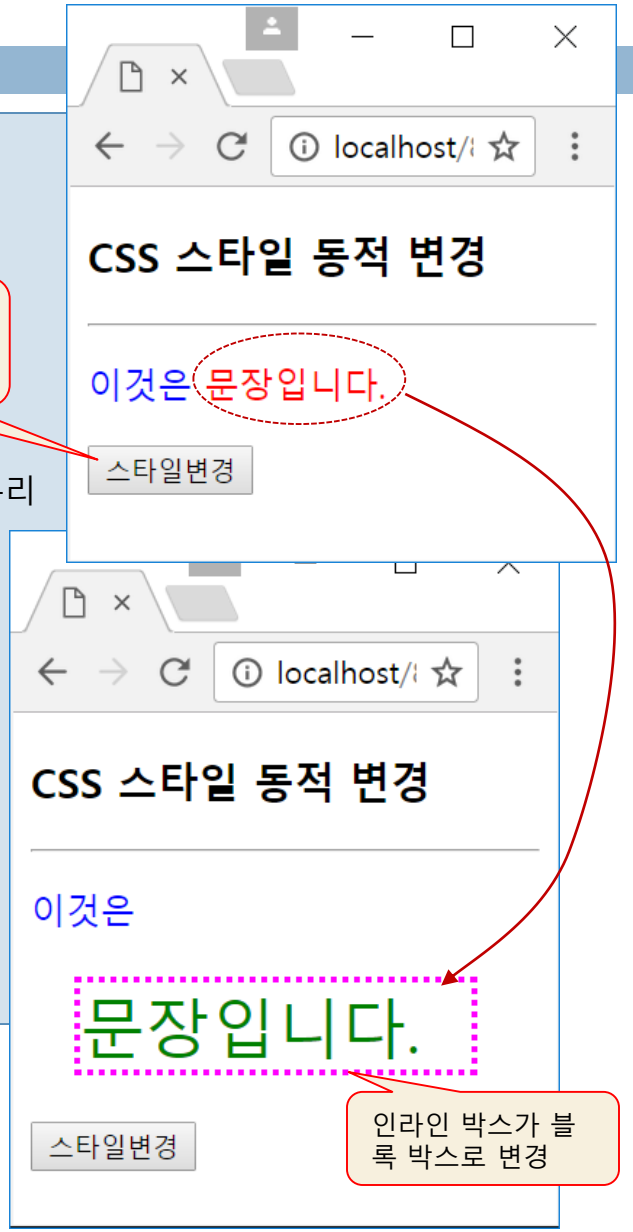
```
span.style.fontSize = "30px";        // '문장입니다'의 폰트를 30px 크기로 변경
```

```
span.style.border = "3px dotted magenta"; // 3픽셀의 magenta 점선 테두리
```

예제: 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

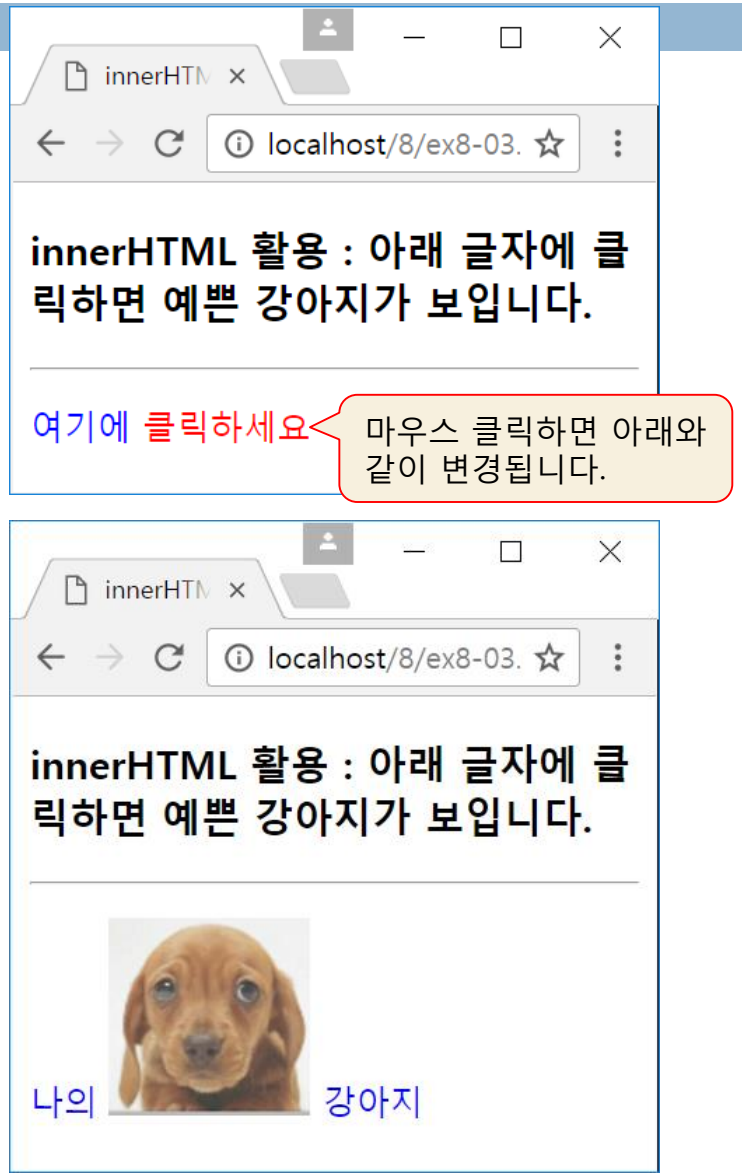
  - innerHTML 프로퍼티 수정 -> HTML 태그의 콘텐트 변경

```
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: innerHTML을 이용하여 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) <p> 태그 자신의 배경을 orange 색으로 변경

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

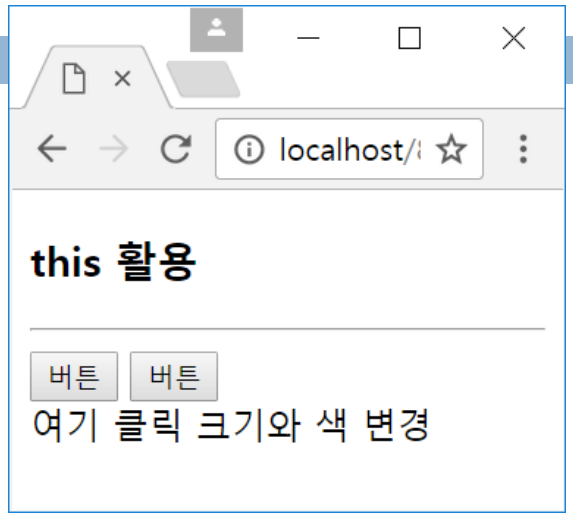
    - 예) 버튼이 클릭되면 자신의 배경색을 orange로 변경

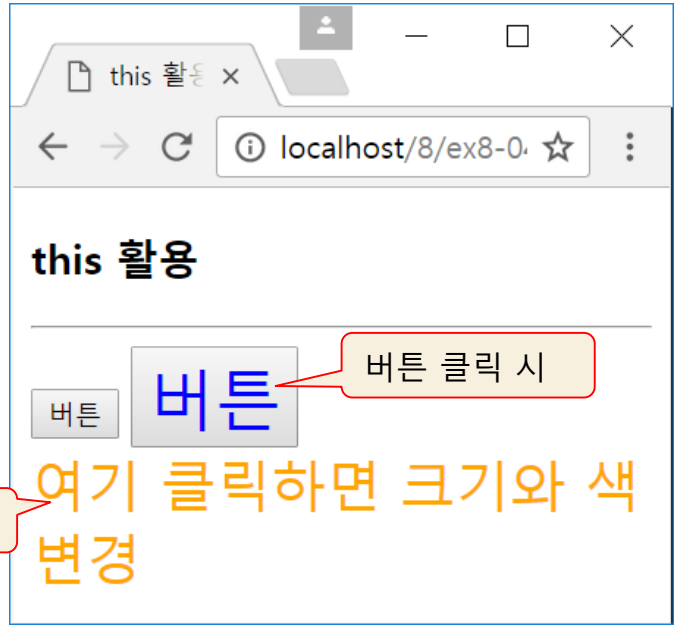
```
<button onclick="this.style.backgroundColor='orange'">
```

예제: 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>
<p onclick="change(this, '25px', 'orange')">
   여기 클릭하면 크기와 색 변경
</p>
</body>
</html>
```

this는 이 <button> 객체의 주소





document 객체

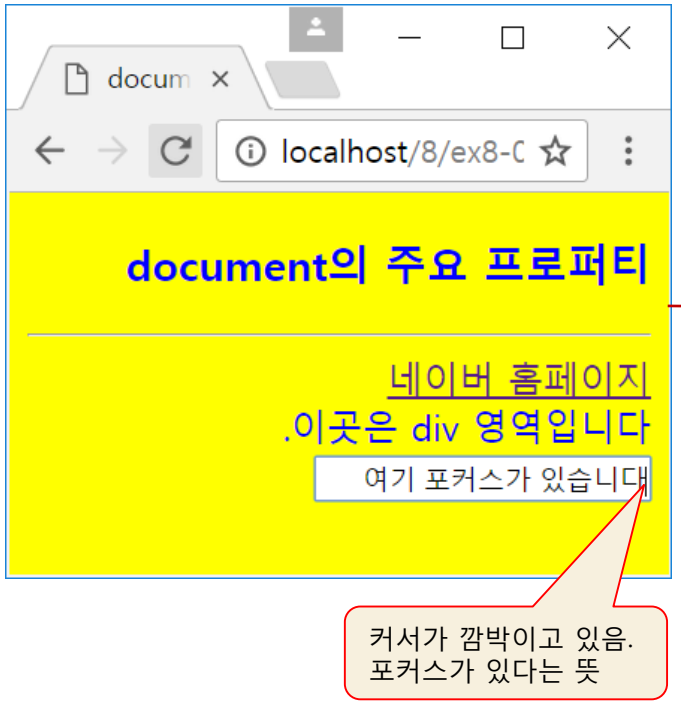
- document
  - HTML 문서 전체를 대변하는 객체
    - 프로퍼티 – HTML 문서의 전반적인 속성 내포
    - 메소드 – DOM 객체 검색, DOM 객체 생성, HTML 문서 전반적 제어
  - DOM 객체를 접근하는 경로의 시작점
  - DOM 트리의 최상위 객체
    - 브라우저는 HTML 문서 로드 전, document 객체를 먼저 생성
    - document 객체를 뿌리로 하여 DOM 트리 생성
- document 객체 접근
  - window.document 또는 document 이름으로 접근
 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

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

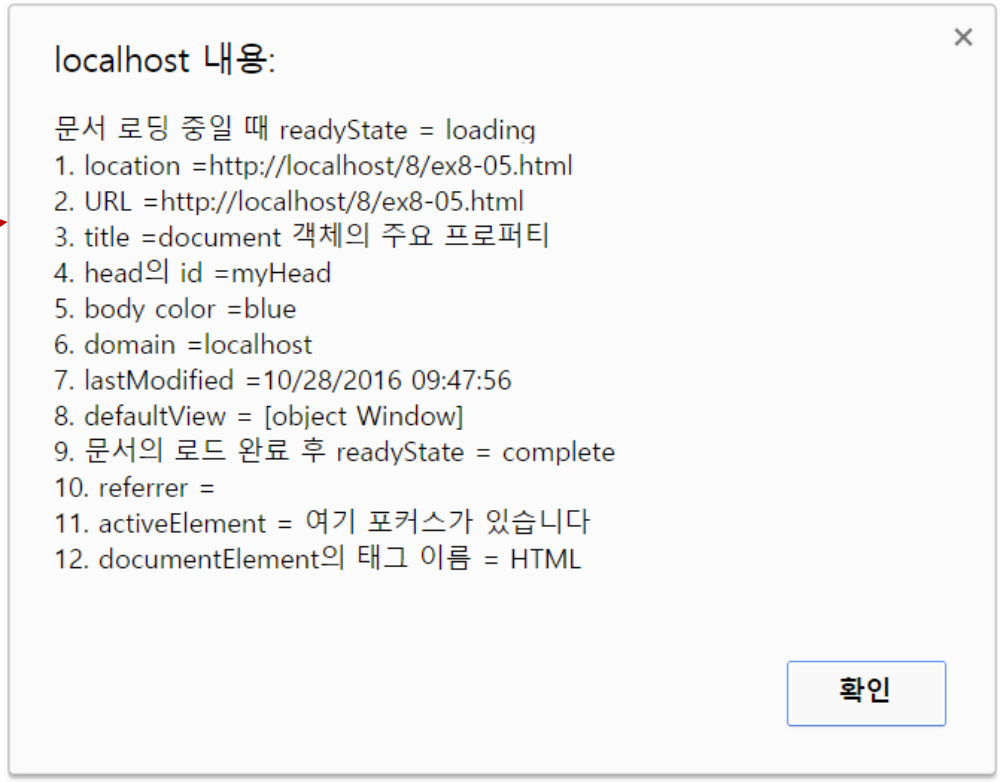
예제: document 객체의 프로퍼티


```
<!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>
```

예제: 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
```

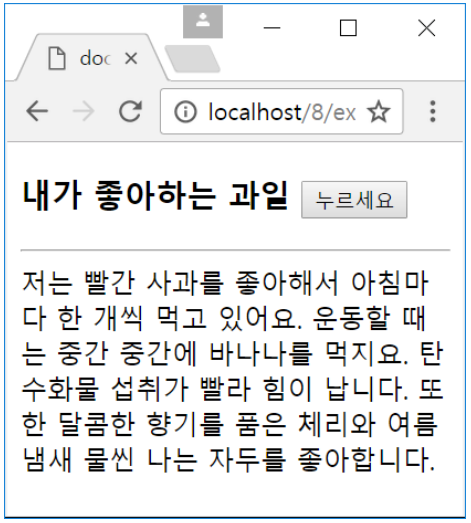
- class 속성으로 찾기
  - document.getElementsByClassName()
    - class 속성이 같은 모든 DOM 객체들을 찾아 컬렉션 리턴
    - 예)

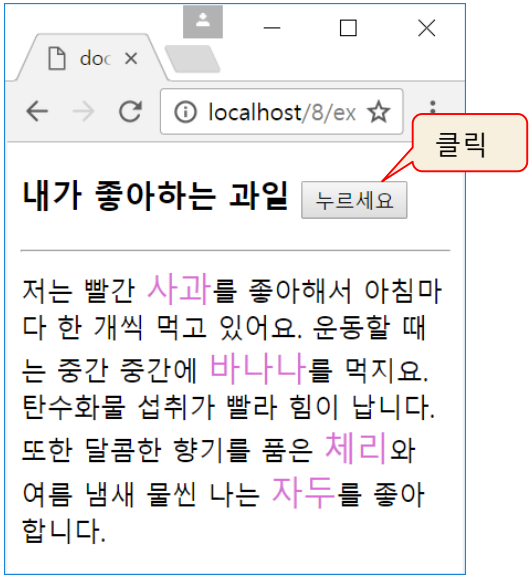
```
<div class="plain">...</div>
<div class="important">...</div>
<div class="plain">...</div>
```

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

예제: 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)

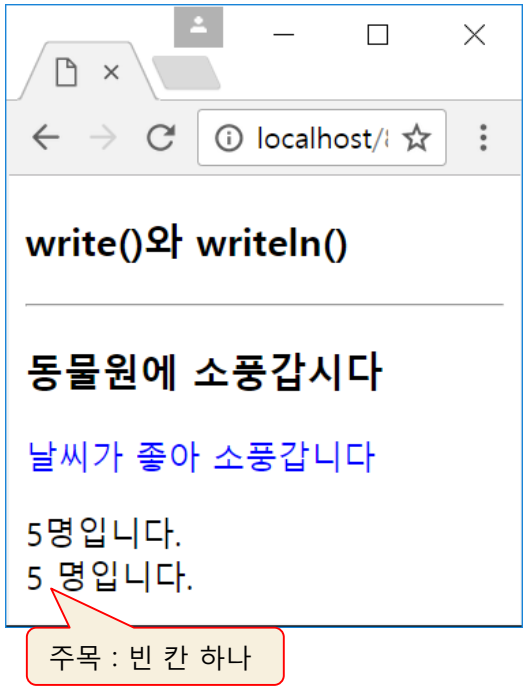
```
document.write("<h3>Welcome to my home</h3>");
document.write(2+3); // 합한 결과 5 출력
document.write("<p>오늘은 " + "sunny day 입니다</p>");
```

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

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

예제


```
<!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>
```



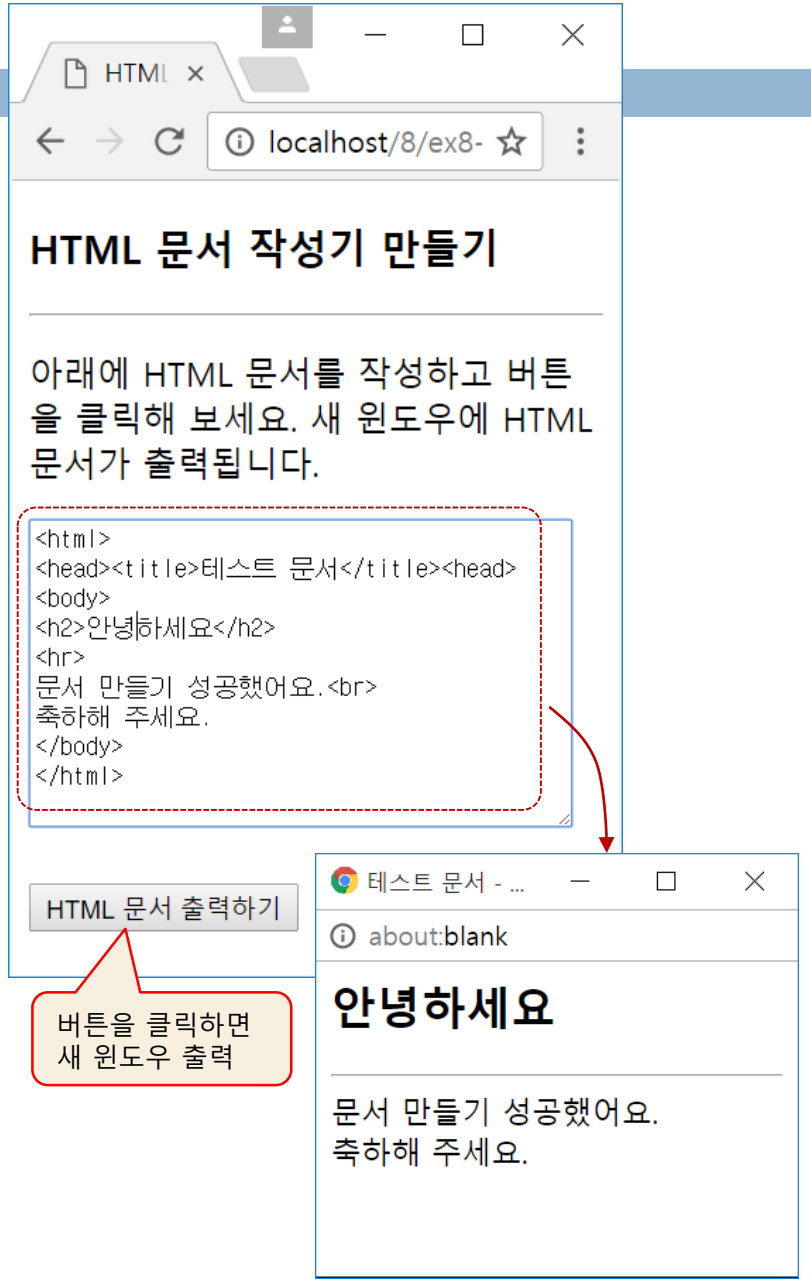
document의 열기와 닫기, open()과 close()

- 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();
```

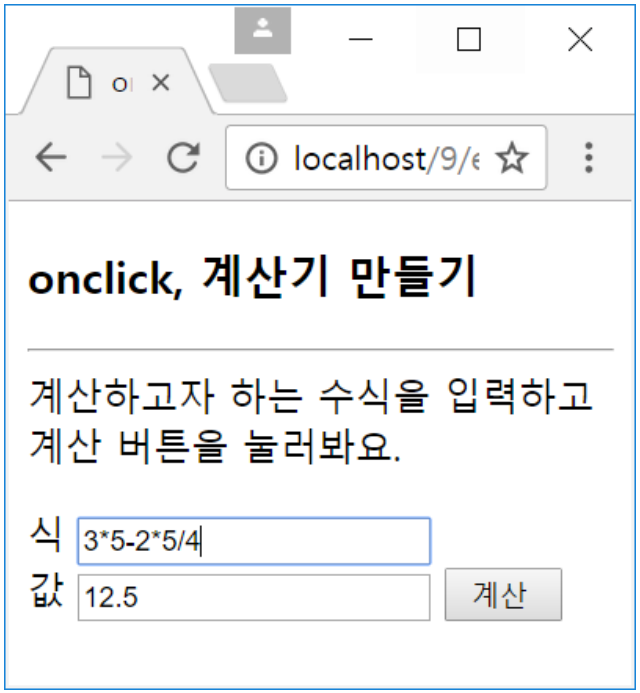
예제


```
<!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>
```



예제


```
<!DOCTYPE html>
<html><head><title>onclick</title>
<script>
function calculate() {
   var exp = document.getElementById("exp");
   var result = document.getElementById("result");
   result.value = eval(exp.value);
}
</script>
</head>
<body >
<h3> onclick, 계산기 만들기</h3>
<hr>
계산하고자 하는 수식을
입력하고 계산 버튼을 눌러봐요.'
<p>
<form>
식 <input type="text" id="exp" value=""><br>
값 <input type="text" id ="result">
<input type="button" value=" 계산  "
      onclick="calculate()">
</form>
</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";
```

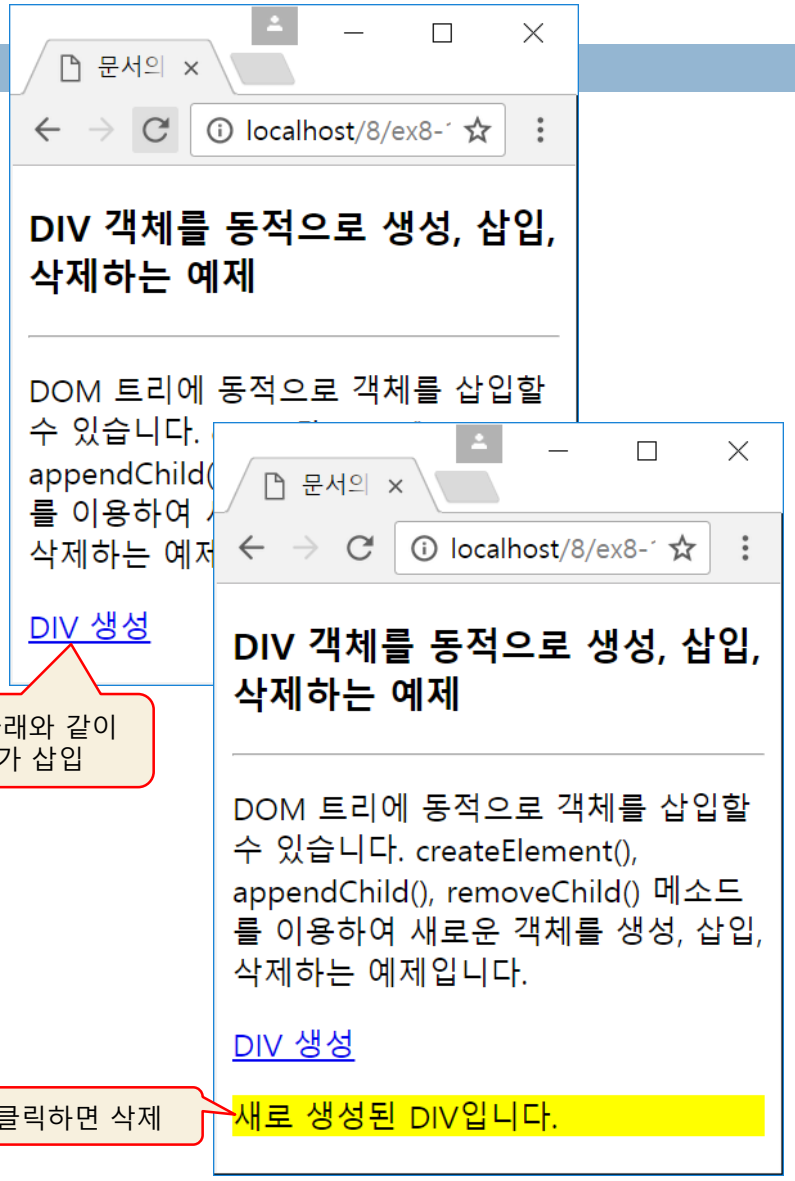


```
<div id="myDiv"
      style="background-color:yellow">
      새로 생성된 DIV입니다.
</div>
```

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

예제: 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>
```



실습과제

- 즐겨듣는 노래 앱에서
  - 새로운 노래를 입력하고 추가 버튼을 눌러 목록에 노래 추가하기
  - 기존 노래 목록에서 삭제 버튼을 눌러 해당 노래 삭제하기