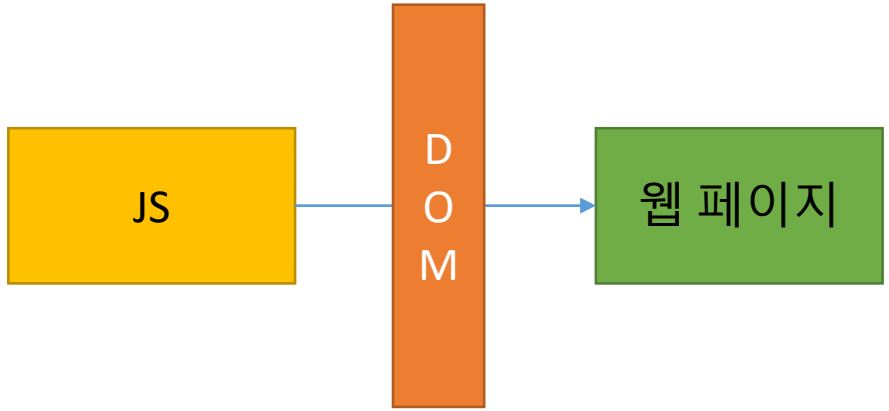
# Document Object Model
# 문서객체모델

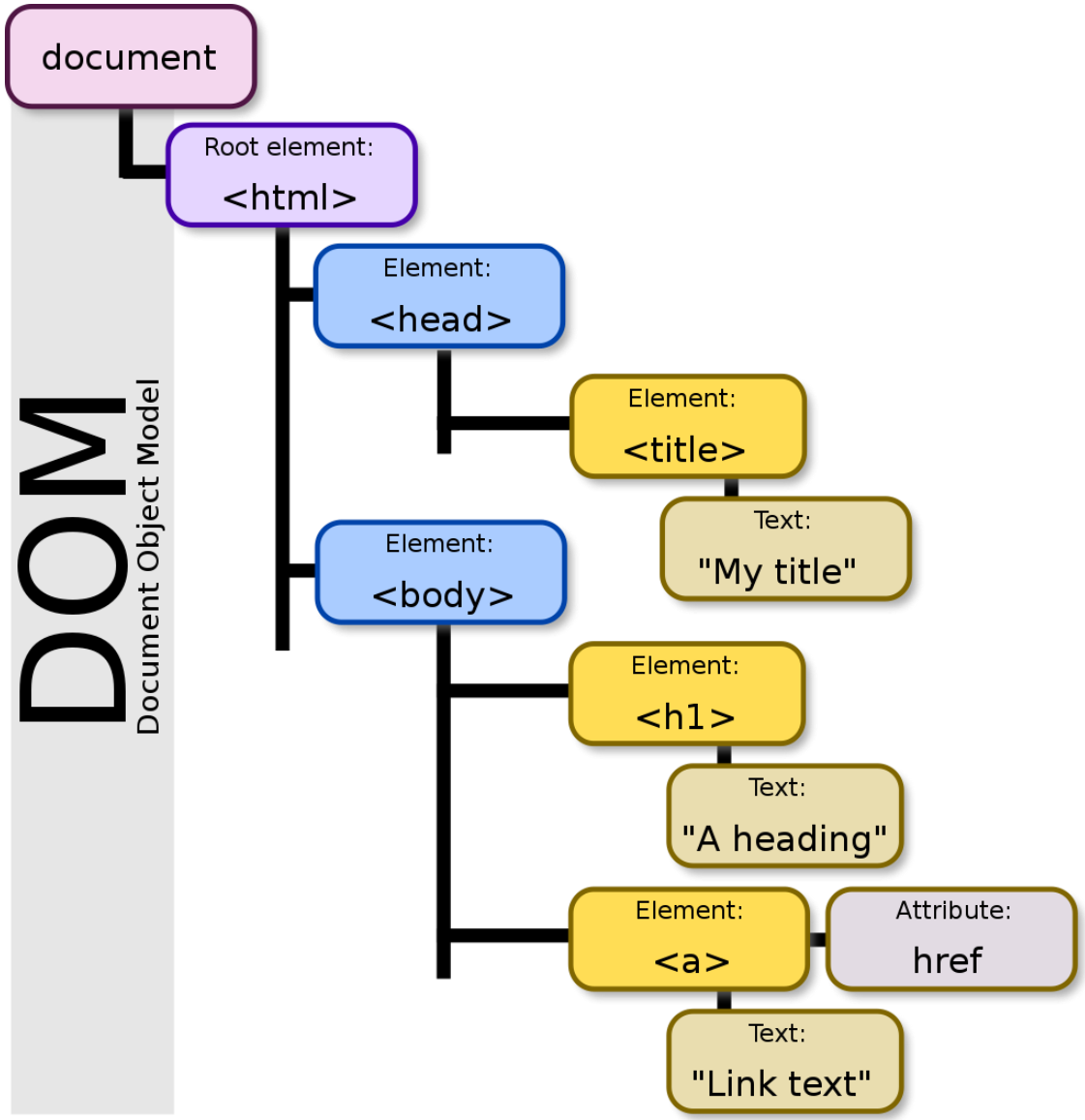
- 요소 가져오기
- 요소 조작하기

# 문서 객체 모델

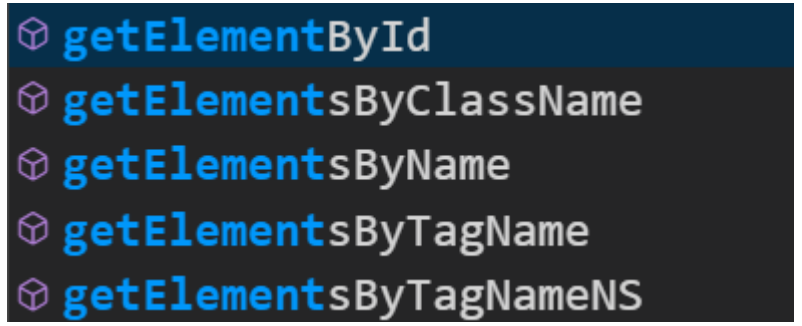
- **DOM**
  - language-independent interface
  - XML이나 HTML과 같은 문서(Document)를 트리구조로 표현하며 각 노드는 객체(Object)
  - DOM 메서드를 이용하여 문서의 구조, 스타일, 내용 등을 변경할 수 있음
  - 웹 페이지가 load되면 브라우저는 웹 페이지의 DOM을 생성
  - 웹 문서를 브라우저가 이해할 수 있는 구조
  - DOM→웹페이지를 자바스크립트로 제어하기 위한 객체 모델




인터페이스



# 문서 객체 모델

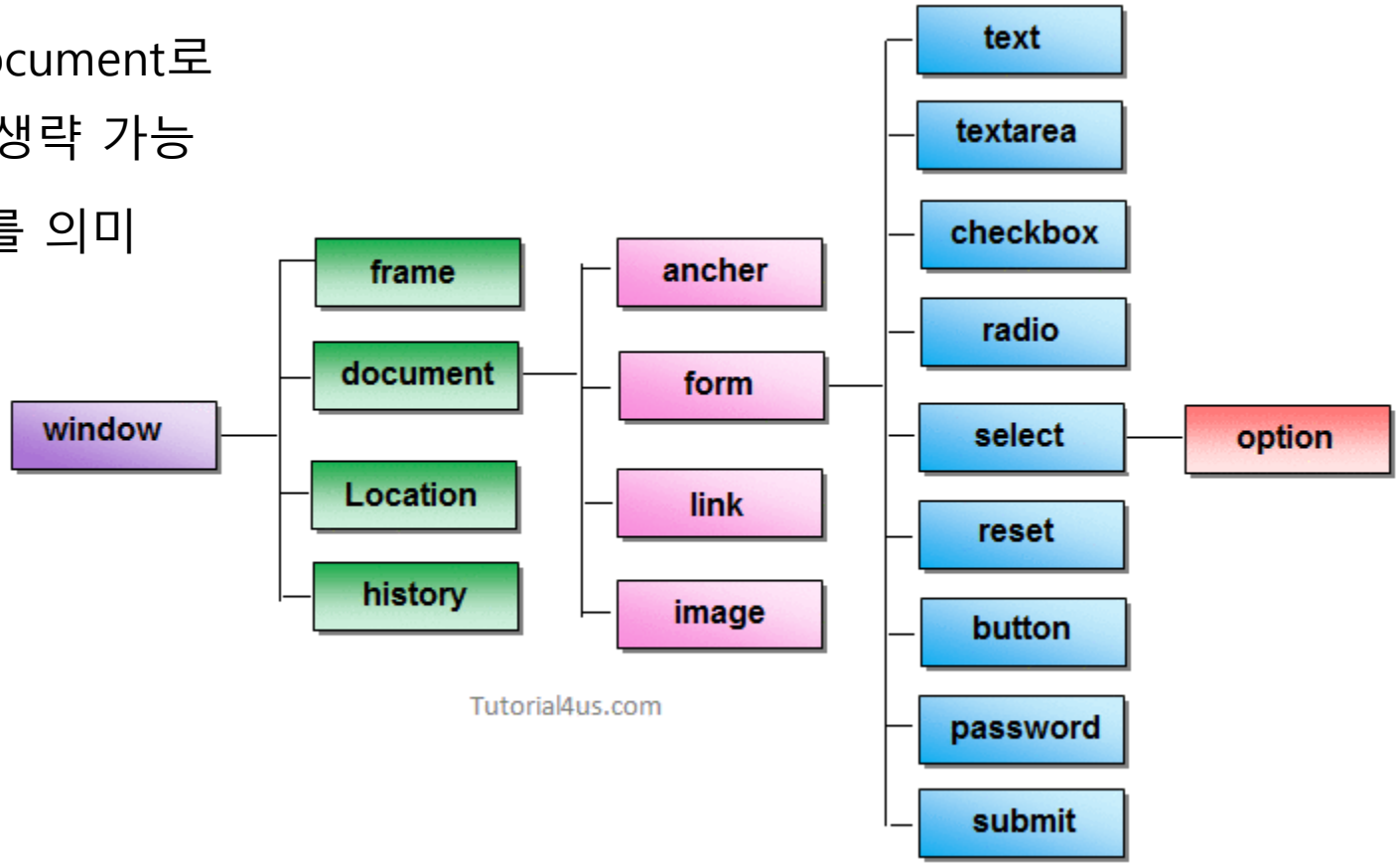



- **제어할 대상 선택하기(Get 메서드)**
  - 제어할 대상을 선택하기 위해 document 객체의 메서드를 이용함
    - 마크업을 이용하여
      - **document**.getElementsByTagName: 태그명을 통해 모든 요소 선택
      - **document**.getElementsByClassName: 클래스명을 통해 모든 요소 선택
      - **document**.getElementById: 아이디명을 통해 요소 선택
    - CSS를 이용하여
      - **document**.querySelector: CSS선택자를 이용해 가장 처음 나오는 요소 하나 선택
      - **document**.querySelectorAll: CSS선택자를 이용해 모든 요소 선택

ID를 사용할 때는 주의 깊게, 남용하지 않도록 해야 함
브라우저는 ID가 중복되어도 경고하지 않으므로 HTML 유효성 검사기 등을 통해 ID의 유일성을 검사해야 함
- html검사 https://validator.w3.org/
- CSS검사 http://www.css-validator.org/

# 문서 객체 모델

- **Document 객체**
  - HTML 태그들 중 최상위 노드
  - 다른 모든 노드(태그)들의 소유자
  - window객체 아래에 있으므로 window.document로 접근해야 하지만 window.는 암묵적으로 생략 가능
  - (window)객체는 전역 변수로 창 그 자체를 의미



# 요소 가져오기

복사 붙여넣기를 하는 과정에
코드가 잘못 인식될 수 있으므로
타이핑으로 테스트하세요

- **getElementsByTagName()** w3schools.com 참조

```
<p>An unordered list:</p>
    <ul>
        <li>Coffee</li>
        <li>Tea</li>
        <li>Milk</li>
    </ul>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>

<script>
    function myFunction() {
        var x = document.getElementsByTagName("LI");
        document.getElementById("demo").innerHTML = x[1].innerHTML;
        for (var i in [...x]) {
            x[i].style.color = 'green';
        }
    }
</script>
```

- getElementsBy…의 반환값은 *유사배열*이므로 for in 혹은 for of를 사용할 수 없음
- 따라서 for(var i; i<lis.length; i++)형태로 사용하거나 확산 연산자(…)를 이용해 배열화 할 수 있음

# 요소 가져오기

- **getElementsByName()**

```
First Name: <input name="fname" type="text" value="Michael"><br>
First Name: <input name="fname" type="text" value="Doug">

<p>Click the button …    value "fname".</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
    function myFunction() {
        var x = document.getElementsByName("fname")[0].tagName;
        document.getElementById("demo").innerHTML = x;
    }
</script>
```

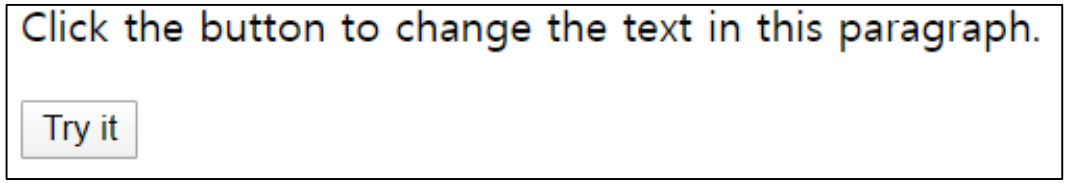
# 요소 가져오기

- **getElementById()**

```
<p id="demo">Click the button to change the text in this paragraph.</p>

<button onclick="myFunction()">Try it</button>

<script>
    function myFunction() {
        document.getElementById("demo").innerHTML = "Hello World";
    }
</script>
```



# 요소 가져오기

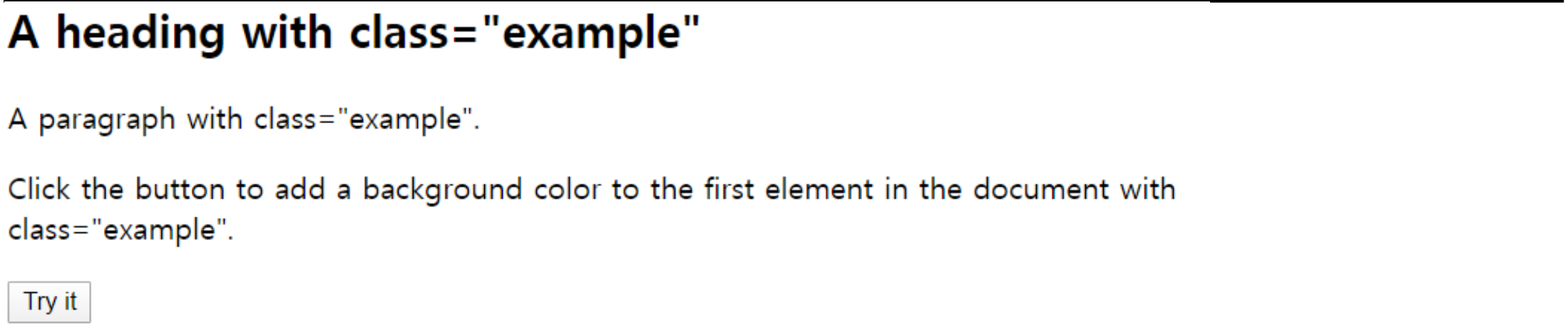
- **querySelector()**

```
<h2 class="example">A heading with class="example"</h2>
<p class="example">A paragraph with class="example".</p>

<p>Click the button to add . . .  in the document with class="example".</p>

<button onclick="myFunction()">Try it</button>

<script>
    function myFunction() {
        document.querySelector(".example").style.backgroundColor = "red";
    }
</script>
```

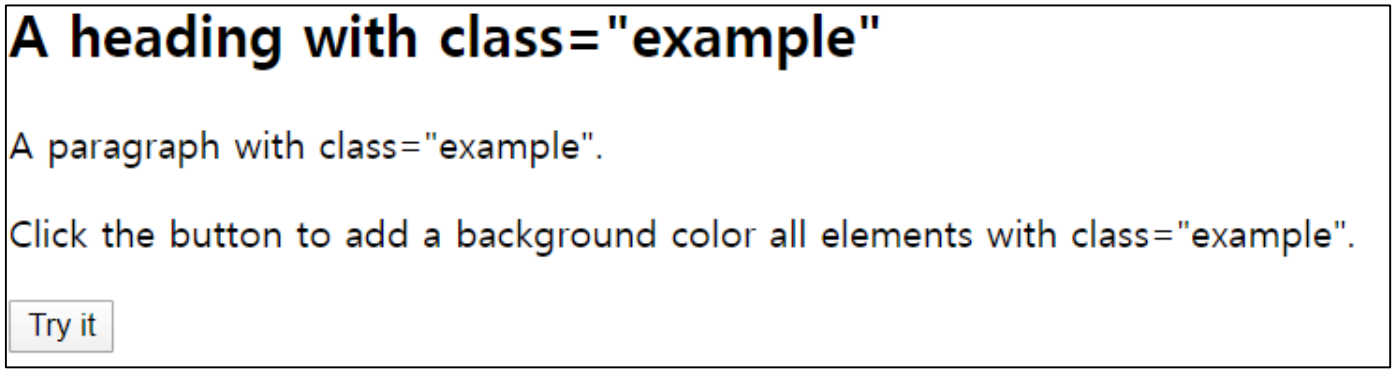


# 요소 가져오기

- **querySelectorAll()**

```
<h2 class="example">A heading with class="example"</h2>
    <p class="example">A paragraph with class="example".</p>
    <p>Click the button to add a background color all elements with class="example".</p>
    <button onclick="myFunction()">Try it</button>

    <script>
        function myFunction() {
            var x, i;
            x = document.querySelectorAll(".example");
            for (i = 0; i < x.length; i++) {
                x[i].style.backgroundColor = "red";
            }
        }
    </script>
```



# 요소 가져오기

- **querySelector**
  - querySelector를 사용할 경우 CSS선택자를 사용할 순 있지만, 가상 클래스를( ex) nth-child(n)) 사용하는 것이 번거로움
  - 대신 Node객체가 제공하는 메서드를 사용하면 선택자의 활용폭이 더 넓어짐
  - 즉, child, parent, sibling과 같은 관계성을 이용하여 요소를 선택할 수 있음
    - firstElement
    - lastElement
    - nextElementSibling

# 요소 가져오기

- **firstElementChild, lastElementChild**

```
<p>Example list:</p>
<ul id="myList">
  <li>Coffee</li>
  <li>Tea</li>
</ul>
<p>Click the button to get the HTML content of the list's first child element.</p>
<button onclick="myFunction()">Try it</button>
<p><strong>Note:</strong> The firstElementChild property . . .versions.</p>

<p id="demo"></p>

<script>
  function myFunction() {
    var list = document.getElementById("myList").firstElementChild.innerHTML;
    document.getElementById("demo").innerHTML = list;
    var list = document.getElementById("myList").lastElementChild.innerHTML;
    document.getElementById("demo").innerHTML = list;

  }
</script>
```

# 요소 가져오기

- **nextElementSibling**

```
<ul>
    <li id="item1">Coffee (first li)</li>
    <li id="item2">Tea (second li)</li>
</ul>


<button onclick="myFunction()">Try it</button>


<p id="demo"></p>

<script>
    function myFunction() {
        var x = document.getElementById("item1").nextElementSibling.innerHTML;
        document.getElementById("demo").innerHTML = x;
    }
</script>
```

# 요소 조작하기

- **요소를 선택한 다음, 요소를 조작**
  - 속성 가져오기
  - 속성 설정하기(CSS조작하기)
  - Text노드 값 가져오기
  - Text노드 값 조작하기
  - 요소 추가/삭제/교체

# 요소 조작하기

- **속성 가져오기**
  - 각 요소는 속성을 가질 수 있으며 속성에는 일반적으로 해당 요소나 콘텐츠에 대한 정보가 들어 있음
  - 접근하는 방법 1) 프로퍼티로 접근 2) 메서드로 접근

```
<div id="myDiv" class="bd" title="Body text" lang="en" dir="ltr"></div>
```

```
var div = document.getElementById('myDiv');
console.log(1, div.id);
console.log(2, div.className);
console.log(3, div.title);
console.log(4, div.lang);
console.log(5, div.dir);
```

class가 아님

프로퍼티로 속성 접근

```
console.log(1, div.getAttribute('id'));
console.log(2, div.getAttribute('class'));
console.log(3, div.getAttribute('title'));
console.log(4, div.getAttribute('lang'));
console.log(5, div.getAttribute('dir'));
div.removeAttribute('dir')
```

메서드로 속성 접근

# 요소 조작하기

- **속성 가져오기**
  - 각 요소마다 접근할 수 있는 속성이 서로 다름

```
<input id="button" type="button" onclick="alert('clicked')" value="go!">
<a id="anchor" href="http://cs.kumoh.ac.kr/cs/index.do"></a>
```

```
var button = document.getElementById('button');
var anchor = document.getElementById('anchor');

console.log(button.href);
console.log(anchor.href);
console.log(button.onclick);
console.log(anchor.onclick);
```

```
undefined
http://cs.kumoh.ac.kr/cs/index.do
ƒ onclick(event) {
alert('clicked')
}
null
```

undefined ➔ 그런 거 없다
null ➔ 그런 거 있긴 한데 값은 없다

# 요소 조작하기

- **속성 설정하기**
  - 속성을 설정하는 방법 1) 프로퍼티를 직접 수정 2) setter이용

```
var button = document.getElementById('button');
var anchor = document.getElementById('anchor');
var para = document.getElementById('para');

button.title = 'inputBtn';
button.onclick = "alert('modified click')" //javascript 적용 안됨
anchor.href = 'http://se.kumoh.ac.kr/';
para.style.fontSize = '32px';
para.style.textAlign = 'right';

anchor.setAttribute('href', 'http://cs.kumoh.ac.kr/cs/index.do');
para.setAttribute('style','font-size:48px;text-align:center; border: 1px solid blue');
```

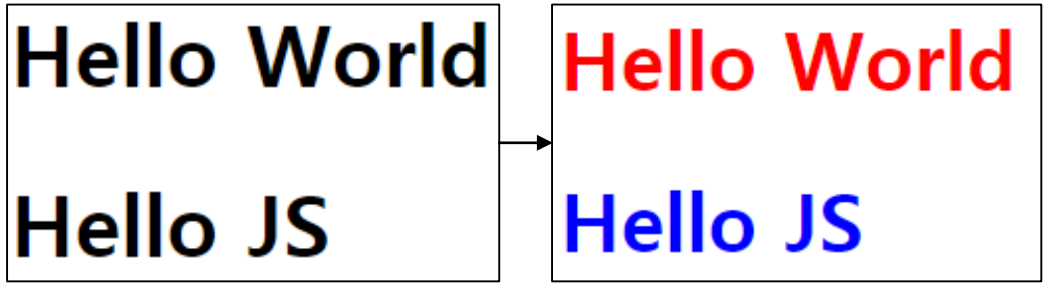
프로퍼티를 이용하는 것과 setAttribute를 이용하여 CSS를 조작할 때 이름 주의!

- 프로퍼티에서 style관련 속성 네이밍: 카멜표기법
- setAttribute style관련 속성 네이밍: 모두 소문자이며 하이픈(-)으로 구분(기존 css와 동일)

# 요소 조작하기

- **속성 설정하기**
  - 직접 값을 변경하기보다는 클래스를 적용(js는 동적 페이지 담당, css는 꾸미기 담당)

```css
.democlass {
  color: red;
}
.democlass1 {
  color: blue;
}
```




```html
<h1>Hello World</h1>
<h1>Hello JS</h1>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  document.getElementsByTagName("H1")[0].setAttribute("class", "democlass");
  document.getElementsByTagName("H1")[1].setAttribute("class", "democlass1");
}
</script>
```

# 요소 조작하기

- **속성 설정하기**
  - 클래스 속성 다루기

```
<div id="classModi" class="first"></div>
<script>
    var classModi = document.getElementById('classModi');
    classModi.className = classModi.className+' second';
    classModi.className = classModi.className+' third';
    // second를 제거할려면?
    var classes = classModi.className.split(' ');
    var classElements = '';
    for(i of classes){
        if(i==='second')
            continue;
        classElements+=i;
        classElements+=' ';
    }
    classModi.className = classElements;    //상당히 번거로움!!
</script>
```

```
<div id="classModi" class="first second third"></div>
```

# 요소 조작하기

토글예제
https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_toggle_class

- **속성 설정하기**
  - 클래스 속성 다루기
    - classList.add(class): 클래스 추가
    - classList.contains(class): 클래스가 있는지 검사
    - classList.item(index): 클래스 반환
    - classList.remove(class): 클래스 제거
    - classList.toggle(class): 클래스가 없으면 추가, 있으면 제거

```
classModi.classList.add('first');      <div id="classModi" class="first"></div>
classModi.classList.add('second'); <div id="classModi" class="first second"></div>
classModi.classList.add('third');     <div id="classModi" class="first second third"></div>
classModi.classList.item(1);           "second"
classModi.classList.remove('fourth');   <div id="classModi" class="first second third"></div>
classModi.classList.remove('second');  <div id="classModi" class="first third"></div>
classModi.classList.toggle('second');    <div id="classModi" class="first third second"></div>
```

# 요소 조작하기

- **요소 추가/삭제/교체**
  - 요소 추가

```
<h2>축제하면 가장 먼저 생각나는 것</h2>
<ol>
    <li class="festival">가수의 무대</li>
    <li class="festival">주점</li>
    <li class="festival">이벤트</li>
</ol>
```

```
<li class="festival">게임</li>
```

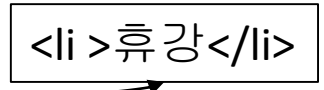
```
var root = document.getElementsByTagName('ol')[0]; //root는 살아있는 객체
var li = document.createElement('li');                    // <li><li>
var content = document.createTextNode('게임');     //  '게임'
li.id = 'newLi';
li.className = 'festival';                                  //<li class="festival"><li>
li.appendChild(content);                                  //<li class="festival">게임<li>
root.appendChild(li);
```

요소를 추가할 경우 부모 요소를 먼저 찾아야 함

# 요소 조작하기

- **요소 추가/삭제/교체**
  - 요소 추가
  - appendChild는 마지막에 추가하지만 insertBefore는 원하는 위치에 추가

```
<h2>축제하면 가장 먼저 생각나는 것</h2>
<ol>
    <li class="festival">가수의 무대</li>
    <li class="festival">주점</li>
    <li class="festival">이벤트</li>
    <li class="festival">게임</li>
</ol>
```




```
var li = document.createElement('li');
var content = document.createTextNode('휴강');
li.appendChild(content);
root.insertBefore(li, root.firstElementChild.nextElementSibling);
```

추가할 요소　　기준 요소

# 요소 조작하기

- **요소 추가/삭제/교체**
  - 요소 삭제
  - remove()

```
<h2>축제하면 가장 먼저 생각나는 것</h2>
<ol>
      <li class="festival">가수의 무대</li>
      <li >휴강</li>
      <li class="festival">주점</li>
      <li class="festival">이벤트</li>
      <li class="festival">게임</li>
</ol>
```

```
var jujum = document.getElementsByTagName('li')[2];
jujum.remove();
```

# 요소 조작하기

- **요소 추가/삭제/교체**
  - 요소 삭제
  - replaceChild()

```
<h2>축제하면 가장 먼저 생각나는 것</h2>
<ol>
      <li class="festival">가수의 무대</li>
      <li >휴강</li>
      <li class="festival">이벤트</li>
      <li class="festival">게임</li>
</ol>
```

```
var li = document.createElement('li');
li.textContent = '체육대회';
root.replaceChild(li, root.lastElementChild);
```

교체할 요소 교체될 요소

# 요소 조작하기

- W3Schools How To(https://www.w3schools.com/howto/default.asp)
- 체크박스: https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_display_checkbox_text
- 탭: https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_tabs
- 사이드바 → https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_sidenav
- 푸쉬 사이드바 → https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_sidenav_push
- 푸쉬 사이드바(+투명) → https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_sidenav_push_opacity
- Clickable Dropdown → https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_js_dropdown
- Carousel → https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_slideshow
- TODO리스트 → https://www.w3schools.com/howto/tryit.asp?filename=tryhow_js_todo