



03



03-3. document



*Implicit Object*

# DOM Node

## Document Object Model

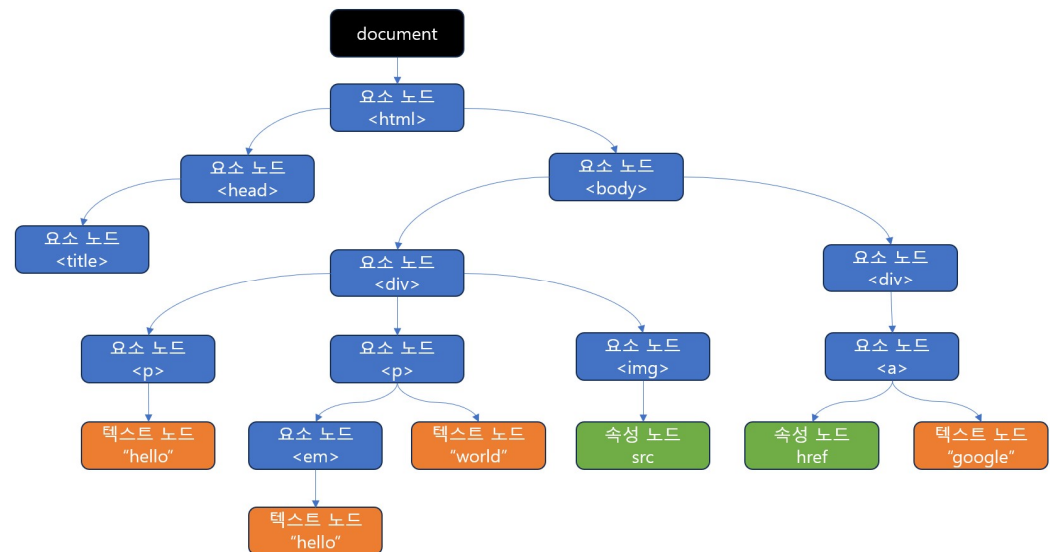
모든 태그가 트리구조로 객체로 묶이고,  
가장 위에 있는 것이 document다.

- document 는 브라우저에서 실행되는 프론트 웹 애플리케이션에서 이용할 수 있는 내장 객체로 브라우저에 출력되는 HTML 문서 자체를 지칭하는 객체입니다.
- document 에 HTML 문서가 파싱되어 만들어진 다양한 DOM 노드 객체가 만들어지며 코드에서는 document 객체를 이용해 이 노드에 접근해 다양한 작업을 하게 됩니다.
- 요소 노드 : 태그를 지칭하는 노드
- 속성 노드 : 태그의 속성을 지칭하는 노드
- 텍스트 노드 : 태그의 바디에 작성된 글을 지칭하는 노드
- 주석 노드 : 문서의 주석을 지칭하는 노드

# DOM Node

HTML 문서

```
1 <html>
2 <head>
3   <title>Document</title>
4 </head>
5 <body>
6   <div>
7     <p>hello</p>
8     <p>
9       <em>hello</em>
10      world
11    </p>
12    
13  </div>
14  <div>
15    <a href="http://www.google.com">google</a>
16  </div>
17 </body>
18 </html>
```



# Node Selector

---

## getElementById()

- HTML 문서의 태그를 지칭하는 DOM 노드를 선택하여 그 노드에 있는 값을 획득하거나 그 노드에 값을 출력해 주어야 합니다.
  - 자바스크립트에서 원하는 노드를 선택하는 기능이 있어야 하는데 이 부분을 노드 선택자(Selector) 라고 합니다.
  - id 속성값으로 식별해 획득을 하거나 노드에 추가된 CSS 클래스 명으로 식별하거나 아니면 태그명으로 식별해 획득해야 합니다.
- 
- 노드를 획득하는 방법은 여러가지가 있지만 가장 기본이 되는 방법이 id 값을 이용해 획득하는 것입니다.

```
let oneNode = document.getElementById('one')
```

document에게 명령을 내린다. 'document 하위에 있는~'

그런데 특정 노드를 얻은 다음에 '특정 노드.getEle...' 도 가능하다.

# Node Selector

---

## getElement~~s~~ByTagName()

- getElementByTagName() 함수는 태그명을 매개변수에 설정하고 그 태그명에 해당되는 노드를 획득하는 함수입니다.
- getElementByTagName() 함수의 반환 값은 획득된 노드 객체 여러 개가 담겨있는 HTMLCollection 객체입니다.

```
let divNodes = document.getElementsByTagName('div')
```

# Node Selector

---

## **getElementsByClassName()**

- 태그와 상관없이 CSS 에 의해 동일한 화면 효과가 적용된 노드를 획득하는 경우인데 이를 위해 `getElementsByClassName()` 이라는 함수를 제공합니다.
- `getElementsByClassName()` 함수의 매개변수에 CSS 클래스명을 지정하면 이 클래스명이 선언된 모든 노드 객체가 `HTMLCollection` 타입으로 반환되게 됩니다.

```
let enableNodes = document.getElementsByClassName('enable')
```

# Node Selector

`<input name="id" />`

## **getElementsByTagName()**

- 태그에 name 속성을 설정하고 그 name 속성값으로 노드를 획득하기 위한 함수가 `getElementsByTagName()` 입니다.
- 이 함수의 매개변수에 name 속성값을 지정하면 반환 값은 `HTMLCollection` 입니다.

`getElementsByTagName()` 으로 노드 획득

```
1 let idInput = document.getElementsByTagName('id')
```

# Node Selector

---

## querySelector() 와 querySelectorAll()

- querySelector(), querySelectorAll() 함수는 document 객체 혹은 element 객체의 함수입니다.
- document 는 HTML 문서 자체를 지칭하며 document.querySelector() 를 사용하면 전체 HTML 문서내에서 원하는 노드를 획득하겠다는 것입니다.
- element 는 HTML 문서내의 특정 노드를 지칭하는 객체이며 element.querySelector() 를 이용하면 특정 노드 하위에 있는 노드들 중 조건에 맞는 노드를 획득하겠다는 것입니다.
- querySelector() : 조건에 맞는 첫 노드를 반환
- querySelectorAll() : 조건에 맞는 모든 노드를 배열로 반환
- CSS Selector 로 조건 설정



# Node 이용

---

## innerHTML vs innerText

- 노드 객체를 다양하게 이용해 주어야 하는데 가장 대표적인 것이 노드 객체 하위에 선언된 자식 노드를 얻거나 변경하는 것입니다.
- 어떤 노드의 자식 노드를 이용할 때 innerHTML 과 innerText 를 이용할 수 있는데 이 둘은 차이가 있습니다.
- innerHTML 은 어떤 노드의 하위에 선언된 모든 구성요소를 포함시키지만 innerText 는 텍스트 노드만 포함시키기 때문에 차이가 발생하게 됩니다.

테스트 HTML	
1	<code>&lt;div id="two"&gt;&lt;a href="#"&gt;google&lt;/a&gt;&lt;/div&gt;</code>

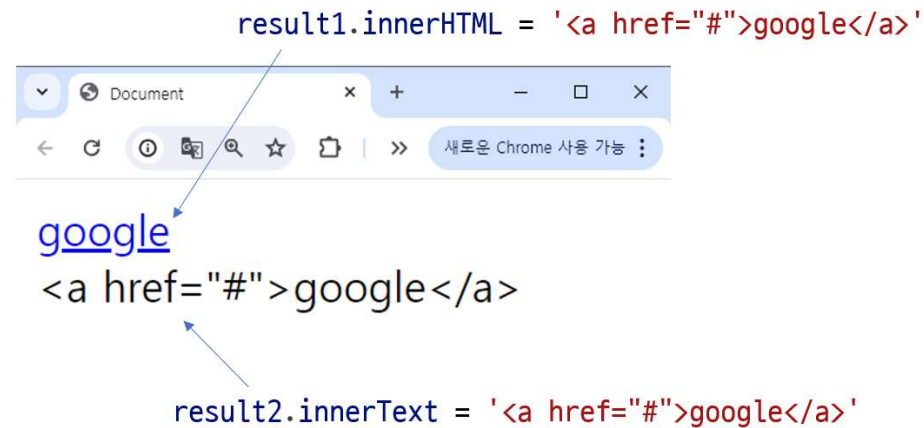
innerHTML 과 innerText	
1	<code>let twoNode = document.getElementById('two')</code>
2	<code>console.log(twoNode.innerHTML)//&lt;a href="#"&gt;google&lt;/a&gt;</code>
3	<code>console.log(twoNode.innerText)//google</code>

# Node 이용

---

## innerHTML vs innerText

- innerHTML, innerText 에 값을 대입해 어떤 노드의 하위 부분을 동적으로 변경시킬 수도 있습니다.



# Node 이용

---

## 속성 이용

- 노드의 속성 값을 획득하거나 변경해야 하는 경우가 있습니다. 그리고 경우에 따라 노드의 속성을 제거해야 하는 경우가 있는데 이를 위해 아래의 함수를 제공합니다.

- `getAttribute()` : 노드의 속성 값 획득
  - `setAttribute()`: 노드의 속성 값 변경
  - `removeAttribute()` : 노드의 속성 삭제
  - `hasAttribute()` : 노드에 속성이 있는지 판단
- 썸네일을 클릭하면 메인 썸네일이 바뀌는 것 (src 값 변경)

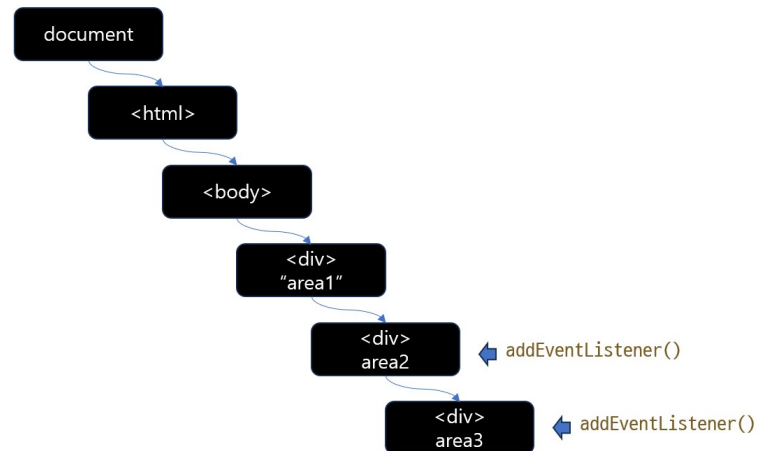
```
console.log(link1.href)//http://www.google.com  
console.log(link1.getAttribute('href'))//http://www.google.com
```

# Node 이용

---

## 타겟팅과 버블링

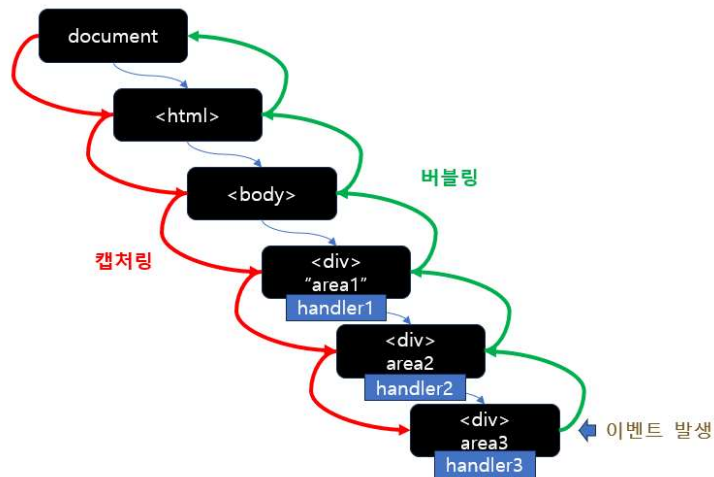
- 중첩구조에서 여러 노드에 이벤트 핸들러를 등록하게 되는 경우가 있습니다.



# Node 이용

## 타겟팅과 버블링

- 노드는 트리구조로 구성되고 트리구조의 여러 노드에 이벤트 핸들러가 등록될 수 있습니다. 이때 실행순서를 신경써야 하거나 혹은 이벤트 실행을 취소 시켜야 하는 경우가 있습니다.
- 캡처링은 이벤트가 발생했을 때 상위 노드부터 하위노드로 이벤트가 전파되는 단계입니다.
- 반대로 버블링은 이벤트가 발생한 노드부터 상위 노드로 이벤트가 전파되는 단계입니다.



# Node 이용

---

## 타겟팅과 버블링

- 하나의 이벤트가 발생하게 되면 캡처링과 버블링 단계를 거치면서 여러 노드에 등록된 이벤트 핸들러가 실행되게 되는데 이때 캡처링 단계에서 실행될 것인지 버블링 단계에서 실행될 것인지를 제어할 수 있습니다.
- 이벤트 처리를 캡처링 단계에서 할 것인지 버블링 단계에서 할 것인지 제어는 이벤트를 등록시키는 함수인 `addEventListener()` 함수의 매개변수로 제어할 수 있습니다.

```
addEventListener(type, listener, useCapture)
```

- `addEventListener()` 함수의 3번째 매개변수는 `true/false` 값을 가지는데 이 값이 `false` 이면 버블링 단계에서 이벤트가 처리됨을 의미하며 `true` 이면 캡처링 단계에서 이벤트가 처리됨을 의미합니다.
- 기본값이 `false` 입니다.

# Node 이용

---

## preventDefault()

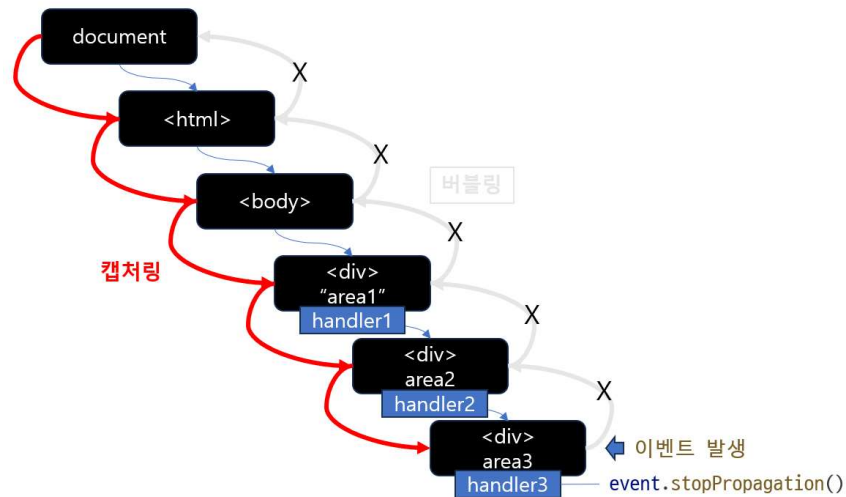
- 이벤트를 중단시키는 방법은 preventDefault() 함수를 이용하는 방법과 stopPropagation() 함수를 이용하는 방법으로 나누어 지는데 둘에 차이가 있습니다
- preventDefault() 는 노드에 기본으로 등록된 이벤트가 처리되지 않게 하기 위해서 사용됩니다.

```
link1.addEventListener('click', function(e) {  
  console.log('link click..')  
  e.preventDefault()  
})
```

# Node 이용

## stopPropagation()

- 노드에 발생하는 이벤트는 캡처링과 버블링 단계를 거쳐 여러 이벤트 핸들러가 실행될 수 있습니다. stopPropagation() 은 이 캡처링과 버블링을 중단시키는 함수입니다.





# Node 이용

---

## 스타일 이용

- 자바스크립트에서 노드에 적용된 스타일 값을 획득하거나 변경해야 하는 경우가 있습니다.
- 이때 이용되는 것이 노드의 style 프로퍼티입니다.

노드의 스타일 확인

```
1 let area1 = document.getElementById('area1')
2
3 console.log(area1.style.width)//200px
4 console.log(area1.style.height)//200px
5 console.log(area1.style.backgroundColor)//green
```

# Node 이용

---

## 스타일 이용

- 노드의 style 프로퍼티로 노드에 적용된 스타일을 이용할 때 스타일 명은 CSS 와 차이가 있습니다.
- CSS 는 대소문자를 구분하지 않습니다.
- 그럼으로 두 단어가 연결되어 스타일 이름이 지정되는 경우 - 로 두 단어를 연결하는 snake case 명명규칙을 따릅니다.
- 하지만 자바스크립트는 대소문자를 구분합니다.
- 그럼으로 두 단어가 연결되는 경우 뒷 단어의 첫 글자를 대문자로 하는 camel case 명명 규칙을 따릅니다.
- CSS 에서 border-color, font-size 등 두 단어가 - 으로 연결된 것은 모두 자바스크립트에서는 borderColor, fontSize 등으로 이용해야 합니다.

# Node 이용

---

## 스타일 이용 - `getComputedStyle()` 함수로 스타일 값 획득

- `style` 프로퍼티는 태그내에서 `style` 속성으로 지정한 스타일만 이용할 수 있습니다.
- 외부에서 정의해서 적용된 스타일을 획득하고자 한다면 노드의 `style` 속성을 이용할 수 없으며 `getComputedStyle()` 함수를 이용해 주어야 합니다.

```
console.log(getComputedStyle(area2).width)//200px
```

# Node 추가

---

- 자바스크립트에서 HTML 문서에 새로운 노드를 추가하는 방법은 이미 살펴본 innerHTML 을 이용할 수 있습니다.
- innerHTML 방식은 자바스크립트 코드에서 문자열을 추가한 것이고 추가된 문자열을 브라우저에서 노드로 만드는 방식입니다.
- 복잡한 구조의 노드를 추가하거나 추가하는 노드의 속성, 바디 문자열을 동적으로 변경하면서 추가하기에는 불편합니다.
- document 객체에는 자바스크립트 코드에서 노드 객체를 직접 만들고 추가할 수 있는 함수를 제공합니다.

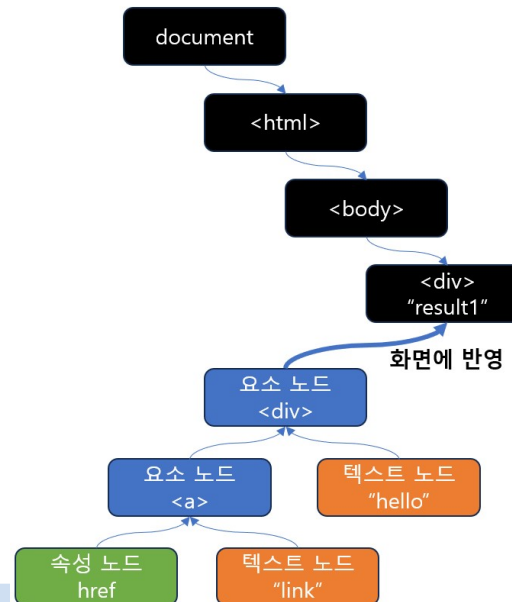
## 노드 객체 생성

- createElement() : 요소 노드 생성
- createAttribute() : 속성 노드 생성
- createTextNode() : 텍스트 노드 생성

# Node 추가

## appendChild() 로 노드 객체 추가

- 만들어진 노드 객체를 추가 해 주어야 합니다.
- 노드 객체는 노드 트리로 구성됨으로 어떤 노드를 다른 노드에 추가해서 화면을 구성하는데 이때 사용되는 함수가 appendChild() 입니다.

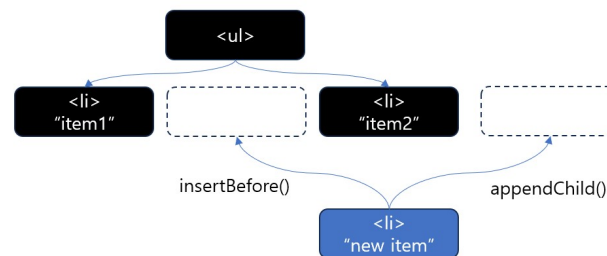


# Node 추가

## insertBefore() 함수로 노드 추가

- appendChild() 함수를 이용하면 새로 추가하는 노드는 마지막 부분에 추가가 됩니다.
- 그런데 경우에 따라 새로 추가하는 노드를 특정 위치에 추가하고 싶다면 insertBefore() 함수를 이용할 수 있습니다.
- insertBefore() 함수의 첫번째 매개변수는 새로 추가하고자 하는 노드이며 두번째 매개변수는 추가되는 위치입니다.

```
list.insertBefore(newItem, list.childNodes[0])
```



# Node 추가

---

## 노드 객체 삭제

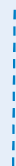
- 특정 노드를 삭제하고 싶다면 `removeChild()` 함수를 이용하면 됩니다.
- `removeChild()` 는 부모 노드에서 자식 노드를 삭제하기 위한 함수로 매개변수에 삭제하고자 하는 노드를 지정해 주면 됩니다.

```
removeChild()
1  newItem.addEventListener('click', function(){
2      console.log('11')
3      list.removeChild(this)
```



# 감사합니다

단단히 마음먹고 떠난 사람은  
산꼭대기에 도착할 수 있다.  
산은 올라가는 사람에게만 정복된다.



윌리엄 셰익스피어  
William Shakespeare