
Template
documentFragment
innerHTML,
innerText,
textContent

Template

Template 내부의 태그들은 HTML이 로드된 후에 렌더링되지 않음

Template 내부의 태그들은 JS를 이용해서 렌더링이 가능

Template 태그는 JS에서 틀로 사용하기 위해 사용. **컨텐츠 조각**

```
template.html > html
1 <!DOCTYPE html>
2 <html lang="en">
3   <head> </head>
4   <body>
5     <template>
6       <div class="myClass">I like :</div>
7     </template>
8     <script>
9       const myArr = ['Audi', 'BMW', 'Ford', 'Honda', 'Jaguar', 'Nissan'];
10      function showContent() {
11        let temp, item, a, i;
12        temp = document.getElementsByTagName('template')[0];
13        item = temp.content.querySelector('div');
14        for (i = 0; i < myArr.length; i++) {
15          a = document.importNode(item, true);
16          a.textContent += myArr[i];
17          document.body.appendChild(a);
18        }
19      }
20      showContent();
21    </script>
22  </body>
23 </html>
```

127.0.0.1:5500/template.html

Elements Console Sources Network

```
<!DOCTYPE html>
<html lang="en">
  <head> </head>
  <body>
    <template>...</template>
    <script>...</script>
    <div class="myClass">I like :Audi</div>
    <div class="myClass">I like :BMW</div>
    <div class="myClass">I like :Ford</div>
    <div class="myClass">I like :Honda</div>
    <div class="myClass">I like :Jaguar</div>
    <div class="myClass">I like :Nissan</div>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>
```

Styles Computed Layout Event Listeners DOM

Filter

Console What's New Issues

top Filter Defi

템플릿을 이용해서 만든 태그들

Template

Template 내부 태그에 document.getElementById나 querySelector로 직접 접근이 불가
<head> <body> <frameset> <table> 등 어떤 태그 안에도 위치 가능

template 내부 태그를 사용하는 방법은, template.content로 내용물을 얻은 후, importNode로 deepcopy하는 방법
template.content가 평가되면, 읽기 전용의 documentFragment 반환

```
template.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3    <head> </head>
4    <body>
5      <template id='mytemplate'>
6        <div class="myClass">I like :</div>
7      </template>
8      <script>
9        const t = document.querySelector('#mytemplate');
10       const clone = document.importNode(t.content, true);
11       document.body.appendChild(clone);
12     </script>
13   </body>
14 </html>
15
```

I like :

Template

Template 내부 태그는 .content로 간접 접근 및 수정이 가능

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head> </head>
4   <body>
5     <button onclick='uselt()'>Use me</button>
6     <div id='container'> </div>
7     <script>
8       function uselt() {
9         const content = document.querySelector('template').content;
10        // template 내부 태그 업데이트
11        const span = content.querySelector('span');
12        span.textContent = parseInt(span.textContent) + 1;
13        document.querySelector('#container').appendChild(
14          document.importNode(content, true)
15        )
16      }
17    </script>
18    <template>
19      <div>Template used : <span>0</span> </div>
20      <script>alert('Thanks!')</script>
21    </template>
22  </body>
23 </html>
24
```

Use me

Template used : 1

Template used : 2

DocumentFragment

메모리상에만 존재하는 **빈 문서 템플릿**

메모리상에서 노드 구조를 만들고, 이를 라이브 DOM에 삽입하면 매우 효율적

DocumentFragment를 라이브 DOM에 추가하면, 생성된 **메모리상의 위치에 존재하지 않음**

노드를 포함하기 위해 일시적으로 사용된 후 라이브 DOM으로 이동되는 **element**는 **메모리상의 위치에 존재**

DocumentFragement에서 **innerHTML**은 **사용이 불가능**

접근하기 위해서는 documentFragment 내부를 div로 한번 더 감싸줘야 함. **이러면 div를 쓰는것과 무엇이 다른지?**

```
<body>
<div id='container'></div>
<script>
const $fragment = document.createDocumentFragment();
const $temp = document.createElement('div');
$fragment.appendChild($temp);
$fragment.querySelector('div').innerHTML =
'<div class=seconddiv> <ul> <li>리스트1 </li> <li>리스트2 <div>리스트3 </div> </li> </ul> </div>'

console.log($fragment.innerHTML); // undefined
console.log($fragment.innerText); // undefined
console.log($fragment.textContent); // 리스트1리스트2
console.log($fragment.querySelector('div').innerHTML);
// <div class='second'> <ul> <li>리스트1 </li> <li>리스트2 </li> <div>리스트3 </div> </ul> </div>
</script>
</body>
```

undefined	template.html:13
undefined	template.html:14
리스트1리스트2리스트3	template.html:15
<div class="seconddiv">리스트1리스트2<div>리스트3</div></div>	template.html:16

innerHTML vs innerText

InnerText 출력

- 사용자에게 보여지는 값 표시, 공백 제거 후 표시
- 레이아웃 정보를 제거하고 표시하므로 무거움(performance-heavy) 뒤에서 추가 설명
- HTMLElement 객체에 정의됨
- innerText로 작성된 <div style='text-transform:uppercase;'>Hello World</div>는 HELLO WORLD 출력

textContent 출력

- 사용자에게 보여지지 않는 값, 자손의 모든 텍스트, 공백 유지 후 표시
- 모든 노드에 정의됨
- textContent로 작성된 <div style='text-transform:uppercase;'>Hello World</div>는 Hello World 출력
- IE8에서 동작 불가
- createTextNode로 생성된 텍스트 노드 읽기 가능

개행 관련

```
body>  
<span></span>  
  <script>  
    var span = document.querySelector('span');  
    span.innerHTML = '1<br>2<br>3<br>4Wn5Wn6Wn7Wn8';  
    console.log(span.innerText); // breaks in first half  
    console.log(span.textContent); // breaks in second half  
  </script>  
</body>
```

1
2
3
4 5 6 7 8
1234
5
6
7
8

innerHTML, innerText, textContent

<div> 안녕하세요? 만나서 반가워요 숨겨진 텍스트 </div>

InnerHTML 출력

<div> 안녕하세요? 만나서 반가워요 숨겨진 텍스트 </div>

InnerText 출력

안녕하세요? 만나서 반가워요
(사용자에게 보여지는 값, 공백 제거)

textContent 출력

안녕하세요? 만나서 반가워요 숨겨진 텍스트
(사용자에게 보여지지 않는 값, 자손의 모든 텍스트, 공백 유지)

리플로우는 태그의 최신 스타일을 알아야 하므로 리플로우를 발생시키고 스타일을 다시 계산함.

textContent는 스타일을 인식하지 못하기 때문에 이 과정이 없음

innerHTML, innerText, textContent

appendChild 대신, 템플릿 리터럴을 사용하는 경우,

1. innerHTML을 사용했으면 innerHTML로 읽어들이든지,
2. textContent나 innerText를 사용했으면, textContent나 innerText로 읽어들이자

```
$btn1.addEventListener('click', () => {  
  $div.innerHTML = '<ul><li>리스트1</li><li>리스트2<div>리스트3</div></li></ul>'  
  console.log($div);  
  console.log($div.innerHTML);  
  console.log($div.innerText);  
  console.log($div.textContent);  
})  
  
$btn2.addEventListener('click', () => {  
  $div.innerText = '<ul><li>리스트1</li><li>리스트2<div>리스트3</div></li></ul>'  
  console.log($div);  
  console.log($div.innerHTML);  
  console.log($div.innerText);  
  console.log($div.textContent);  
})  
  
$btn3.addEventListener('click', () => {  
  $div.textContent = '<ul><li>리스트1</li><li>리스트2<div>리스트3</div></li></ul>'  
  console.log($div);  
  console.log($div.innerHTML);  
  console.log($div.innerText);  
  console.log($div.textContent);  
})
```

▶ <div id="container">...</div>

리스트1리스트2<div>리스트3</div>

리스트1

리스트2

리스트3

리스트1리스트2리스트3

▶ <div id="container">...</div>

리스트1리스트2<div>리스트3</div>

리스트1리스트2<div>리스트3</div>

리스트1리스트2<div>리스트3</div>

▶ <div id="container">...</div>

리스트1리스트2<div>리스트3</div>

리스트1리스트2<div>리스트3</div>

리스트1리스트2<div>리스트3</div>

innerHTML 사용시 주의할 점

innerHTML의 값이 어떤 방식으로든 직접 변경되면, 기존 하위 엘리먼트를 모두 삭제하고, 새로운 innerHTML의 내용을 기준으로 다시 HTML을 파싱하고 엘리먼트들을 생성

```
for(var i = 0; i < 1000; i++) {  
    document.body.innerHTML += `<div>~Pen Pineapple Apple Pen~</div>`  
}
```

innerHTML을 이용해서 태그를 생성하면, 기존의 자식 엘리먼트들도 추가된 태그와 함께 다시 파싱되고 만들어짐

루프를 돌고난 후 보여지는 div 태그 개수는 1,000개
파싱해서 만들거나 지우게 된 div 태그 개수는 500,500개

루프를 돌면서 파싱하는 양도 점점 많아져서 수행하는 시간이 길어짐. 해결 방법은 아래와 같다.

```
str = "";  
  
for(var i = 0; i < 1000; i+=1) {  
    str += `<div>~Pen Pineapple Apple Pen~</div>`  
}  
  
document.body.innerHTML = str;
```

textContent와 nodeValue 비교

Run details:

User agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36

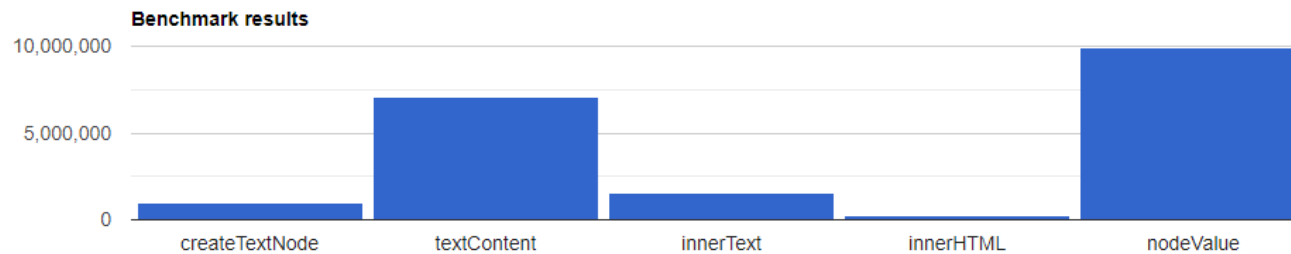
Browser: Chrome 79

Operating system: Mac OS X 10.15.2

Device Platform: Desktop

Date tested: one year ago

Test name	Executions per second
createTextNode	1014412.0 Ops/sec
textContent	7049860.0 Ops/sec
innerText	1516966.0 Ops/sec
innerHTML	223298.4 Ops/sec
nodeValue	9883064.0 Ops/sec



참고

<https://www.measurethat.net/Benchmarks/ShowResult/91680>

Reference

https://www.w3schools.com/tags/tag_template.asp

<https://programmer-seva.tistory.com/60>

<https://www.html5rocks.com/en/tutorials/webcomponents/template/>

<https://stackoverflow.com/questions/35213147/difference-between-textcontent-vs-innertext>

<https://blog.shiren.dev/2016-11-15->

[%ED%81%AC%EB%A1%AC%EA%B0%9C%EB%B0%9C%EC%9E%90%EB%8F%84%EA%B5%AC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%9C-%ED%94%84%EB%A1%A0%ED%8A%B8%EC%97%94%EB%93%9C-%EC%84%B1%EB%8A%A5%EC%B8%A1%EC%A0%95/](https://blog.shiren.dev/2016-11-15-%ED%81%AC%EB%A1%AC%EA%B0%9C%EB%B0%9C%EC%9E%90%EB%8F%84%EA%B5%AC%EB%A5%BC-%EC%9D%B4%EC%9A%A9%ED%95%9C-%ED%94%84%EB%A1%A0%ED%8A%B8%EC%97%94%EB%93%9C-%EC%84%B1%EB%8A%A5%EC%B8%A1%EC%A0%95/)