DOM (Document Object Model)

문서객체모델(DOM)은 웹 브라우저가 웹 문서를 분석하고 접근하기 위해 필요한 가장 기본적인 개념으로
웹문서 안의 텍스트, 태그, 주석문, 속성 등 모든것을 Node 객체로 인식하고 다룬다.

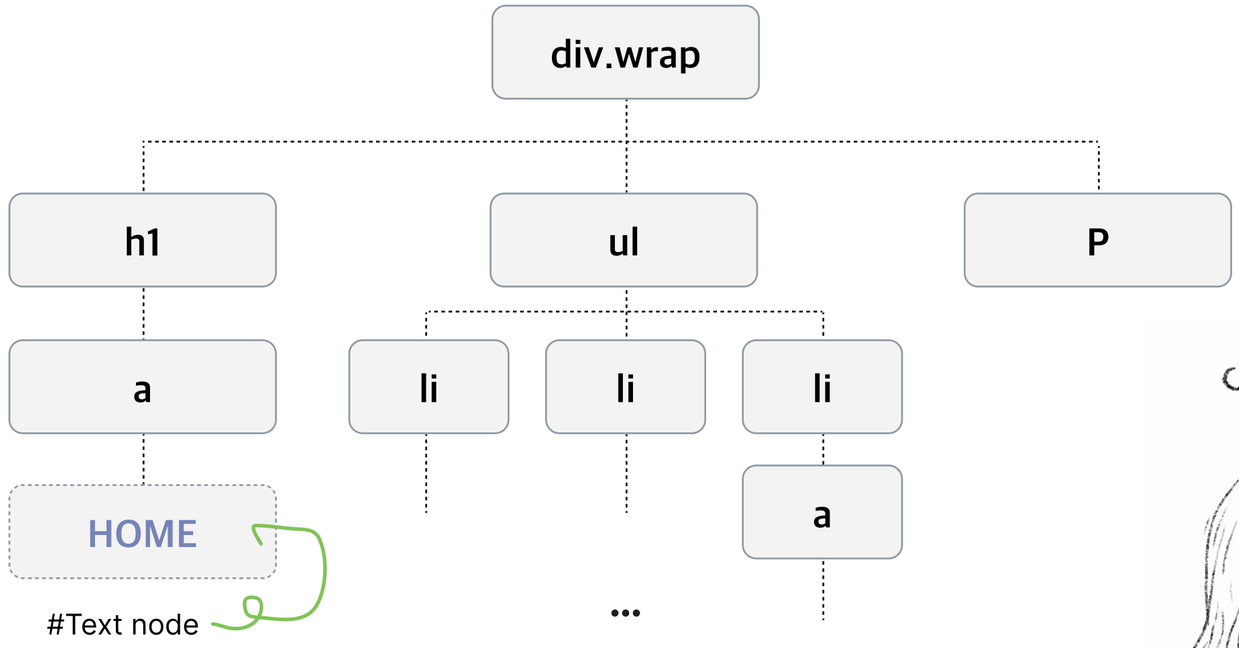
Access › | Css › | Event › | Class › | Form › | Elements Control ›

DOM TREE

HTML 웹 문서는 <html> 태그로 시작하면서 내부에 tag 의 중첩이 일어난다. 태그들이 중첩이 되면서 부모, 자식, 형제, 조상, 자손등의 계층구조가 정해지며 이 구조를 표현 한 것을 DOM TREE 라고 부른다.
아래의 HTML 문서를 웹 브라우저가 해석하여 DOM tree로 표현한 그림을 확인해 보자.

😡 HTML 태그의 중첩이 잘못되거나 웹표준에 어긋날 경우 DOM tree 해석이 잘 못 될 수 있으니, 코드가 올바른지 먼저 꼭 확인해야 한다.

```
<div class="wrap">
  <h1>
    <a href="#">HOME</a>
  </h1>
  <ul>
    <li>GHOST</li>
    <li>DRACULAR</li>
    <li>
      Horror Movie
      <a href="#">view site</a>
    </li>
  </ul>
  <p>Hello</p>
</div>
```



<- GO HOME

DOM with JS - access

Document객체를 통한 DOM 접근 방법

(document.getElementById())

getElement~~

(getElements는 HTML collection을 반환한다.)

- getElementById(id)
- getElementsByTagName(tag)
- getElementsByClassName(class)

querySelector~

(querySelectorAll은 NodeList를 반환한다.)

- querySelector(css selector)
- querySelectorAll(css selector)

특정 태그 종류에 따른 접근방식

- .documentElement
- .body
- .links
- .forms

Element를 통한 HTML 속성 접근방법

(Element는 문서의 모든 요소 객체(즉, 요소를 나타내는 객체)가 상속되는 가장 일반적인 기본 클래스입니다.)

MDN Reference

EX) Element.dataset

Element.attributes

(element가 가지고 있는 속성을 통한 접근방식 일부 소개)

- .dataset
- .src
- .id
- .className
- .기타 다른속성이나 name값으로 접근가능

<- GO HOME

DOM with JS - css

Blog summary

style property를 이용한 접근

Javascript로 css 선언문을 추가, 수정, 삭제 하거나, 리딩할 수 있다.

단일속성 지정하기

```
Element.style.css속성명 = value
```

- css속성명은 camel표기법을 사용한다.

```
const h1 = document.querySelector('h1');
h1.style.color = 'red';

console.log(h1.style)
console.log(h1.style.cssText)
```

다중속성 지정하기

```
Element.style.cssText = 'font-size:20px; color:red';
```

- string 으로 css 선언문을 나열한다.

<- GO HOME

DOM with JS - event

Event 종류 정리 | MDN | JS.info

이벤트 적용하기 및 처리기(ON과 addEventListener)

웹 페이지에서 어떠한 이벤트가 발생했을때에, 이벤트 발생이후에 처리할 동작을 지정해야하는데, 이때에 이벤트 처리기(핸들러)가 필요하다.

---

on이벤트명

```
target.onevent = function
```

- event명이 on뒤에 바로 붙는다.
- function은 이벤트 발생시 수행할 동작이다.

addEventListener

```
target.addEventListener(event, function, useCapture)
```

- event는 String 형식으로 지정해야 한다.
- function은 이벤트 발생시 수행할 동작이다.
- useCapture의 기본값은 false 라서 버블링이 일어나고, true로 지정시 버블링대신 캡쳐링이 일어나게 된다.

```
window.onload = ( ) => { }
window.addEventListener(‘load’, ( ) => { } );

<button onClick=”console.log(‘hello’)”>CLICK</button> // HTML 요소에 inline으로 이벤트를 주는 경우
```

NEXT →

<- GO HOME

DOM with JS - event

Event 종류 정리 | MDN | JS.info

1. 문서로딩 이벤트

Event	이벤트가 발생하는 순간
DOMContentLoaded	웹브라우저가 HTML을 리딩하여 DOM 트리구조를 만든 직후에 발생하며, CSS 및 Image, Video, Audio등의 자원이 로딩될때까지 기다리지는 않는다. Document객체에서 발생한다. 그래서 on 이벤트는 적용되지 않고, addEventListener를 사용해야 한다. document.addEventListener('DOMContentLoaded', () => { ... });
load	DOM 트리구조 완성 및 이미지,CSS등의 자원들의 로딩도 끝났을 때 발생
beforeunload	웹페이지를 종료하려고 할때
unload	웹페이지가 종료될 때
abort	웹페이지 로딩중 완료되지 못하고 중지되었을 때
error	웹페이지가 온전히 로딩되지 못했을 때
resize	웹페이지화면의 크기에 변동이 생겼을 때
scroll	웹페이지에 스크롤 동작이 발생했을 때

← BACK NEXT →

<- GO HOME

DOM with JS - event

Event 종류 정리 | MDN | JS.info

2. 마우스 이벤트

Event	이벤트가 발생하는 순간
click	클릭 동작이 발생했을 때
dblclick	더블클릭 동작이 발생했을 때
mouseenter	HTML 요소에 마우스를 올렸을 때,
mouseleave	mouseenter 이벤트발생후 해당 HTML요소에서 벗어날 때
mouseover / mouseout	child element에도 계속해서 이벤트가 발생하므로 주의
mousemove	마우스가 해당요소안에서 움직일 때

3. 키보드 이벤트

Event	이벤트가 발생하는 순간
keydown	키를 누르는 동안
keyup	키보드에서 손을 떼는 순간(눌렀던 키가 올라왔을 때)
keypress	키를 누르는 순간

← BACK NEXT →

<- GO HOME

DOM with JS - event

Event 종류 정리 | MDN | JS.info

4. 폼 이벤트

Event	이벤트가 발생하는 순간
change	폼요소의 상태가 변경되는 순간
focus	폼요소가 포커스 받았을 때
focusin	폼요소의 자식요소가 포커스 받았을 때
focusout	폼요소의 자식요소가 포커스를 잃었을 때
blur	폼요소가 포커스를 잃었을 때
reset	폼태그에 Reset 되었을 때
submit	폼태그에서 submit 버튼이 클릭 되었을 때

BACK

<- GO HOME

DOM with JS - classList

Example →

MDN Reference

classList속성으로 HTML 요소의 class 속성 제어하기

(클래스명은 String으로 표기하여 넣는다)

- `.add(class)` - 클래스 추가

  클래스가 이미 존재한다면 추가로 붙이지 않는다. 여러개 한번에 지정 가능

- `.remove(class)` - 클래스 삭제

  지정한 클래스가 없어도 에러를 발생시키지 않는다. 여러개 한번에 지정 가능

- `.toggle(class)` - 클래스 추가/삭제 토글(인수가 한개일때는 계속 반복된다)
- `.toggle(class, condition)` - 인수가 2개일 때 condition 이 true일때는 add, false 일때는 remove

- `.contains(class)` - 클래스 확인(Boolean 값 리턴)
- `.replace(oldClass, newClass)` - oldClass명 삭제후, newClass를 추가함

<- GO HOME

DOM with JS - form

Example →

form요소에 접근하기

id, class, name 값등을 통하여 form요소에 접근할 수 있다.

- `.value` - 선택한 요소의 값을 지정하거나 가져온다
- `.options` - 옵션목록들을 가져온다.
- `document.forms[0].elements` - 폼태그내의 폼관련 elements 들을 가져온다.

<- GO HOME

DOM with JS - Element control

Summary

HTML 요소를 추가/수정/삭제 등 제어해 보는 방법을 알아보자

(Document 또는 다른 요소에 HTML element를 지정할 수 있는 방법)

- `document.createElement(element)` - element 생성

  HTML 요소노드를 새로 생성한다.

- `element.remove()` - element 삭제

  지정한 HTML 요소노드를 삭제한다.

- `element.removeChild(target)` - element의 target자식요소 삭제

- `element.appendChild(target)` - target요소를 element의 child로 추가
- `element.append(target)` - target요소를 element의 child로 추가(여러개의 노드, 텍스트 및 자식노드 포함)

NEXT →

<- GO HOME

DOM with JS - Element control

Summary

HTML 요소의 컨텐츠에 접근하여 제어

(아래의 속성으로 Elements의 컨텐츠의 내용을 지정하거나 삭제할 수 있다.)

- `.innerHTML` - html 요소를 포함하여 수정가능
- `.innerText` - 텍스트만 지정가능
- `.textContent` - 텍스트만 지정가능

“

여기서 잠깐~!

textContent와 innerText는 둘다 텍스트를 가져오는데 무슨차이가 있나요?

> innerText는 브라우저가 텍스트를 렌더링 한 후의 모습을 인식하여서 css에서 숨기거나 한 글들은 가져오지 않습니다.
> 하지만 textContent는 css를 인식하지 않기때문에 해당 Node의 텍스트를 모두 출력합니다.

← BACK