**DOM이라는 용어 개념정리 & load 이벤트**

개념설명 시간이라 글로 진행합니다.

**1. DOM 이라는 말이 자주 등장합니다.**

책보고 검색하고 그러면 맨날 DOM이라는 용어를 남용하는데 그게 뭔지 알아봅시다.

Document Object Model 이라는 뜻인데 그냥 설명하면 이해가 안되니 배경설명부터 갑니다.

자바스크립트는 HTML 조작에 특화된 언어라고 설명드렸습니다.

그런데 자바스크립트가 어떻게 HTML을 조작할 수 있는지 원리를 생각해본 적이 있습니까.

생각해보면 HTML과 자바스크립트는 다른 언어입니다.

그래서 자바스크립트에선 <p></p> 이런 html을 직접 해석하고 조작할 수 없습니다.

<script>

<p></p>.innerHTML = '안녕' (당연히 에러날듯)

</script>

자바스크립트는 <p> 이런건 못알아듣습니다.

그런데 어떻게 HTML 태그들을 알아보고 조작할 수 있는 것일까요?

자바스크립트가 HTML 조작을 하기 위해선 **HTML을 자바스크립트가 해석할 수 있는 문법으로 변환해놓면**됩니다.

HTML을 자바스크립트가 좋아하는 array 혹은 object 자료형에 담아버리는게 좋지 않을까요?

그래서 실제로 브라우저는 HTML 페이지를 열어줄 때

HTML을 자바스크립트로 쉽게 찾고 바꾸기 위해 object와 비슷한 자료형에 담아줍니다.

예를 들어서

<div style="color : red">안녕하세요</div>

브라우저는 이런 HTML을 발견하면 object 자료로 바꿔서 보관해둡니다.

구체적으로는 var document = { } 이런 변수를 하나 만들어서 거기 넣어줍니다.

var document = {

div1 : {

style : {color : 'red'}

innerHTML : '안녕하세요'

}

}

▲ 대충 이렇게 object 자료에 정리를 해놓는다는 소리입니다.

왜냐고요? 이래야 저렇게 점찍어서 원하는 자료를 출력하고 조작하고 할 수 있지 않겠습니까.

이제 document.div1.innerHTML = '안녕' 이렇게 자바스크립트를 짜면 HTML 조작이 가능할듯요

그래서 저렇게 object에 담아두는 것입니다.

(물론 실제 DOM과 생김새는 좀 다르겠지만)

위 변수를 **document object**라고 부릅니다. 간지나게 + model이라고 붙여서 Document Object Model이라고 합니다.

요약하자면 자바스크립트가 HTML에 대한 정보들 (id, class, name, style, innerHTML 등)을

object 자료로 정리한걸 DOM이라고 부릅니다.

**2. 브라우저는 HTML 문서를 위에서 부터 읽으며 DOM을 생성합니다.**

브라우저는 HTML문서를 위에서 부터 차례로 읽어내려갑니다.

읽을 때 마다 HTML을 발견하면 DOM에 추가해줍니다.

그래서 이걸 알면 왜 하단 코드가 에러가 나는지도 알 수 있겠군요.

(html 파일)

<script>

document.getElementById('test').innerHTML = '안녕'

</script>

<p id="test">임시글자</p>

이렇게 코드짜면 에러가 납니다.

왜겠습니까

왜냐면 브라우저는 HTML을 위에서부터 한줄한줄 읽습니다.

그런데 갑자기 자바스크립트로 <p id="test">인 요소를 DOM에서 찾고 바꾸라고하는겁니다.

그래서 에러가 날 뿐입니다.

왜냐면 아직 <p id="test">를 읽기 전이라 p태그에 대한 DOM이 아직 생성되지 않았으니까요.

이렇듯 자바스크립트는 DOM이 생성된 경우에만 HTML을 변경할 수 있습니다.

**3. 혹은 자바스크립트 실행을 약간 나중으로 미루는 방법도 있음**

"이 코드는 HTML 전부 다 읽고 실행해주세요" 라고 코드짜놓을 수 있습니다.

그러면 아까 예제같은 문제가 발생하지 않습니다.

$(document).ready(function(){ 실행할 코드 })

document.addEventListener('DOMContentLoaded', function() { 실행할 코드 })

둘 중 마음에 드는거 쓰면 됩니다.

이 이벤트리스너들은 "HTML을 다 읽어들였는지"를 알려주는 이벤트리스너입니다.

(html 파일)

<script>

document.addEventListener('DOMContentLoaded', function() {

document.getElementById('test').innerHTML = '안녕'

})

</script>

<p id="test">임시글자</p>

그래서 거기 안에 "저 밑에있는 <p id="test"> 변경해주세요~" 라고 코드 짜도 잘 동작함

그래서 예전스타일 강의들 보시면 ready 아니면 DOMContentLoaded 이벤트리스너를 꼭 쓰라고 가르칩니다.

안정적이니까요.

게시판에 "왜 html 변경안되냐"는 질문도 없고 좋을듯

요즘은 그냥 자바스크립트를 <body>태그 끝나기 전에 전부 작성하기 때문에 그런 걱정은 안하셔도됩니다.

자바스크립트 위치를 내가 정할 수 없을 경우에만 유용한 방법이라고 보시면 되겠습니다.

**4. load 이벤트리스너**

배운 김에 이것도 배워보도록 합시다.

load 라는 이벤트리스너를 사용하면 DOM 생성뿐만 아니라 이미지, css, js파일이 로드가 됐는지도 체크가능합니다.

이미지 같은게 로드되면 load라는 이벤트가 발생하기 때문입니다.

셀렉터로찾은이미지.addEventListener('load', function(){

*//이미지 로드되면 실행할 코드*

})

이런 식으로 쓰면 됩니다.

근데 외부 자바스크립트 파일에 저걸 적어놓으면

그 js 파일보다 이미지가 더 먼저 로드되는 경우도 있으니 이벤트 발생체크를 못할 수도 있겠군요

$(window).on('load', function(){

*//document 안의 이미지, js 파일 포함 전부 로드가 되었을 경우 실행할 코드*

});

window.addEventListener('load', function(){

*//document 안의 이미지, js 파일 포함 전부 로드가 되었을 경우 실행할 코드*

})

 window에 붙이셔도 됩니다. document에 포함된 이미지, CSS파일 등 모든것이 로드가 되었는지 체크해줍니다.

ready 이런거랑 차이는 앞선 .ready()는 DOM 생성만 체크하는 함수인데,

이것보다 약간 더 나아가서 모든 파일과 이미지의 로드상태를 체크한다고 보시면 되겠습니다.

그래서 이거 써서 "이미지가 다 로드되면 사이트 보여주세요~" 이런 코드짜는 사람들이 있는데

한국인이라면 오히려 속터짐