Javascript DOM 객체

키워드 : DOM (https://www.w3schools.com/js/js_htmldom.asp)

우리가 테스트 코드를 작성하고 VS Code를 통해 웹 페이지를 실행시키면, 브라우저는 자동으로 우리의 페이지의 DOM(Document Object Model)이라는 것을 생성하게 됩니다. 이는 우리가 보고 있는 모든 웹 페이지에 적용되는 규칙입니다. 이번 장에서는 이 DOM에 대해서 알아보도록 하겠습니다.

사실, Javascript 구조에 대해서 이야기하며 DOM에 대해 잠시 언급한 적이 있습니다. 그때의 기억을 상기하며 다시 한 번 DOM을 한 마디로 정리하면 아래와 같이 나타낼 수 있습니다.

> DOM(Document Object Model)은 웹 문서에 동적으로(Dynamically) 접근하기 위한 표준 방법이다.

그렇다면, DOM 객체를 이용해 어떻게 동적으로 웹 문서에 접근할 수 있는지 알아보도록 하겠습니다.

DOM property

DOM은 하나의 객체입니다. 그렇기 때문에, 다른 내장 객체와 마찬가지로 여러 가지 속성과 메서드를 가지고 있습니다. 이 메서드를 이용해서 동적으로 HTML 문서에 접근하고 이미 만들어진 웹 페이지에 접근하여 사용자 요청에 맞는 처리를 가능하게 할 수 있습니다.

우선 대표적인 DOM 객체의 속성은 다음과 같습니다.

> innerHTML : 엘리먼트 내의 내용(Content)을 가져오고 변경할 수 있는 속성

강의 영상에서도 살펴봤듯이, `innerHTML` 속성을 이용해서 값을 가져오고 변경하는 작업을 자유롭게 할 수 있습니다. 하지만, 우선 HTML 문서 안에 접근할 수 있어야겠죠? 계속해서 아래 DOM 메서드를 통해 엘리먼트(Elements)에 접근하는 방법에 대해 알아보겠습니다.

DOM methods

DOM 메서드를 통해 웹 페이지의 특정 요소에 접근하는 메서드는 굉장히 다양합니다. 하나하나 정리해보도록 하겠습니다.

우선, DOM 객체의 요소 접근 메서드는 다음과 같습니다.

메서드	의미
document.getElementById("id")	id 속성으로 요소 반환
document.getElementsByClassName("class")	class 속성으로 여러 요소 반환
document.getElementsByTagName("tag")	태그명으로 여러 요소 반환
document.getElementsByName("name")	name 속성으로 여러 요소 반환
document.querySelectorAll("css selector")	CSS 선택자 문법으로 찾은 여러 요소 반환

위 메서드와 `innerHTML` 속성을 이용하여 특정 요소의 내용을 변경하거나 가져올 수 있습니다.

아래는 자바스크립트 코드를 이용해 요소에 접근하여 콘텐츠를 가져오는 코드 예시입니다.

- index.html

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Javascript DOM 객체</title>
    </head>
    <body>
       <div id="container">
          <h1 id="head1">DOM 객체!</h1>
          <p class="test">문단 태그</p>
          <p class="test">문단 태그</p>
          <form action="recieve.php" method="GET">
              <input type="text" name="name" value="">
              <input type="text" name="pw" value="">
          </form>
          <div id="head2">
              <p class="test">문단 태그</p>
              <p class="test">문단 태그</p>
          </div>
       </div>
       <script type="text/javascript" src="js/main.js"></script>
    </body>
</html>
```

index.html 코드를 실행하면 위 사진과 같이 나오게 됩니다. 아래 사진은 main.js 코드를 이용해 HTML 엘리먼트를 동적으로 변화시킨 사진입니다. 하나하나 알아보겠습니다.

- main.js

```
//1번 코드
document.getElementById("head1").innerHTML = "아이디 속성으로 접근";
//2번 코드
document.getElementsByTagName("p")[0].innerHTML = "태그로 접근";
//3번 코드
document.getElementsByClassName("test")[1].innerHTML = "클래스 속성으로 접근";
//4번 코드
document.getElementsByName("pw")[0].value = "name 속성으로 접근";
//5번 코드
document.querySelectorAll("div#head2 > p.test")[0].innerHTML = "선택자 문법으로
접근";
```

- 1번 코드는 아이디 속성이 head1인 요소를 찾아서 콘텐츠를 변경하는 코드입니다.
- 2번 코드는 <p>태그를 찾아서 가장 첫 번째로 찾은 요소의 콘텐츠를 변경하는 코드입니다.
- 3번 코드는 클래스 속성이 test인 요소 중 인덱스가 1인 요소의 콘텐츠를 변경하는 코드입니다.

- 4번 코드는 name 속성이 pw인 요소 중 가장 첫 번째로 찾은 요소의 value 속성 값을 변경하는 코드입니다.
- 5번 코드는 선택자 문법을 통해 아이디 속성이 head2인 태그의 자식 요소 중, 클래스 속성이 test인 p태그를 찾아 콘텐츠를 변경하는 코드입니다.

위 코드에서 눈여겨볼 내용은 크게 2가지입니다. 하나는 **"s"** 가 붙은 메서드는 HTMLCollection 혹은, NodeList라는 것으로 요소를 여러 개 반환합니다. 그렇기 때문에, HTML 요소가 하나만 존재하더라도 인덱스로 접근해야 합니다. e.g., document.getElementsByClassName[0]

또 다른 하나는 4번 코드에서 볼 수 있는 value, href, src, etc와 같은 엘리먼트의 속성을 동적으로 변경할 수 있다는 점입니다. e.g., document.getElementById("id").src = "js/main.js"

추가로 DOM 객체의 요소에 접근하는 것이 아닌 엘리먼트 자체의 속성을 변경하거나 엘리먼트를 추가, 삭제하는 메서드에 대해서 알아보겠습니다.

메서드	의미
document.write("Content")	HTML 코드에 자바스크립트 코드로 내용(content) 쓰기
document.setAttribute("Attribute", "Value")	요소의 속성을 변경
document.createElement("Element")	동적으로 요소 생성
document.removeChild(Node)	동적으로 요소 삭제
document.appendChild(Node)	동적으로 자식 요소 추가
document.replaceChild(New node, Old node)	동적으로 자식 요소 대체

아래 코드는 동적으로 요소와 속성을 추가하는 간단한 예시입니다.

- index.html

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Javascript DOM 객체</title>
    </head>
    <body>
       <div id="container">
          <h1 id="head">DOM 객체!</h1>
          <p class="test">문단 태그</p>
          <p class="test">문단 태그</p>
       </div>
       <script type="text/javascript" src="js/main.js"></script>
    </body>
</html>
```

- main.js

```
//1번 코드
var newP = document.createElement("p");
//2번 코드
document.getElementById("container").appendChild(newP);
//3번 코드
newP.setAttribute("class", "newP");
//4번 코드
newP.innerHTML = "문단 태그";
```

자바스크립트 코드로 DOM 객체의 새로운 요소를 추가하거나 삭제하기 위해선 노드(Node)라는 개념을 사용해야합니다. HTML 코드의 엘리먼트들은 DOM 구조 상에서 노드라는 것으로 표현된다고 이해하면 쉽습니다. 위 `main.js`코드를 예로 들어 하나하나 알아보겠습니다.

- 1번 코드는 newP라는 변수를 선언하고, 해당 변수에 <p> 요소를 생성하여 할당하는 코드입니다.
- 2번 코드는 아이디 속성이 `container`인 요소를 찾고, 1번 코드에서 선언한 newP요소를 자식 요소로 추가하는 코드입니다.
- 3번 코드는 1번 코드에서 생성한 요소의 클래스 속성을 newP로 변경하는 코드입니다.
- 4번 코드는 1번 코드에서 생성한 요소의 콘텐츠를 변경하는 코드입니다.

3번과 4번 코드는 어렵지 않게 이해하셨을 거라 생각됩니다. 위 코드에서 주의 깊게 봐야 할 한 가지는 `createElement` 후에 `appendChid()`한다는 점입니다. 만약, `appendChild("요소")`형태로 코드를 작성하면, `parameter is not of type 'Node'`라는 에러가 발생됩니다. 노드를 CreateElement로 먼저 만들고, 후에 해당 노드를 `append`, `replace`, `remove` 등을 할 수 있습니다.

위 코드를 실행하면, 아래와 같이 엘리먼트의 자식 요소로 <p>태그가 추가되고, 클래스 속성 값과 콘텐츠 내용이 추가된 것을 확인할 수 있습니다.

- index.html

```
<div id="container">
    <h1 id="head">DOM 객체!</h1>
    <p class="test">문단 태그</p>
    <p class="test">문단 태그</p>
    <p class="newP">문단 태그</p>
</div>
```

노드와 관련된 더 자세한 내용은 여기 (https://www.w3schools.com/js/js_htmldom_nodes.asp)와 여기 (https://www.w3schools.com/js/js_htmldom_nodelist.asp)를 참고해주세요!

위에서 HTML 엘리먼트에 접근, 값 변경 등을 진행하는 것에 대해 알아보았습니다. 마지막으로, 이번 절에서는 CSS style 코드를 변경하는 방법을 간단히 살펴보겠습니다. 위 내용을 잘 따라오셨다면, css 코드를 변경하는 방법은 굉장히 간단하게 느껴지실 겁니다.

```
document.getElementById("id").style.property = "값";
```

위 기본 형식에서 `property` 부분을 적절한 스타일 코드로 변경하고, 원하는 값을 할당해주면 됩니다. 아래 코드는 요소의 색을 변경하는 간단한 예시입니다.

```
document.getElementById("head").style.color = "red";
```

단, 주의할 점은 `property`의 이름이 css 코드와 다른 내용도 있다는 점에 유의하시면 됩니다.

이번 장에서는 DOM 객체를 이용해서 HTML 엘리먼트에 접근하는 방법에 대해서 알아보았습니다. DOM 객체 부분은 다소 난이도 있는 내용이 많이 등장하는 파트입니다. 하지만, 자바스크립트 코드뿐만 아니라, 웹 개발의 필수적인 내용 중 하나입니다.

클라이언트 측 개발에 관심을 갖고 공부를 진행하다 보면, 자바스크립트를 조금 더 쓰기 쉽게 만든 제이쿼리(jQuery)와 같은 여러 프레임워크와 라이브러리를 마주하게 되실 겁니다. 미리 기본기를 잘 다져놓으면, 나중에 만나는 많은 프레임워크와 라이브러리를 더 잘 이해하고 혼합하여 사용할 수 있을 겁니다! :)