JavaScript 101 - (7) DOM

DOM이란 Document Object Model의 약자로, 웹 페이지에 대한 프로그래밍 인터페이스입니다. DOM은 원본 HTML 문서의 객체 기반 표현으로, 자바스크립트와 같은 스크립트 언어를 통해 웹 페이지의 콘텐츠, 구조, 스타일을 읽고 조작할 수 있게 해줍니다.

DOM의 개념과 구조

- DOM의 구조는 트리 형태로 되어 있으며, 각각의 요소들은 노드라고 부릅니다. 노드에는 여러 종류가 있지만 가장 중요한 것은 문서 노드(document node), 요소 노드(element node), 속성 노드(attribute node), 텍스트 노드(text node)입니다.
- DOM을 통해 자바스크립트에서 웹 페이지의 내용과 스타일을 동적으로 변경할 수 있습니다. DOM을 사용하는 방법은 다양하지만 가장 기본적인 것은 DOM API를 이용하는 것입니다. DOM API는 문서 내의 요소들을 선택하고, 속성과 스타일을 변경하고, 요소를 추가하거나 삭제하는 등의 작업을 할 수 있는 메서드와 프로퍼티들의 집합입니다.
    - `document.getElementById(id)` : id 속성으로 요소를 선택하는 메서드
    - `document.querySelector(element)` : id, class, 태그 속성으로 요소를 선택하는 메서드
    - `document.querySelectorAll(element)` : 같은 (id, class, 태그) 속성을 공유하는 모든 요소를 선택하는 메서드
    - `element.classList.add(class)` : class 속성에 클래스를 추가하는 메서드
    - `element.style.color = "red"` : style 속성에 색상을 지정하는 프로퍼티
    - `element.appendChild(child)` : 자식 요소를 추가하는 메서드
    - `element.removeChild(child)` : 자식 요소를 삭제하는 메서드

이벤트와 DOM을 활용한 동적 웹페이지 제작 예제

이벤트와 DOM을 활용하면 사용자와 상호작용하는 동적인 웹페이지를 제작할 수 있습니다. 예를 들어 다음은 버튼을 클릭하면 배경색이 바뀌는 간단한 예제입니다.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        body {
            background-color: white;
        }
    </style>
</head>
<body>
    <button id="change-color">색상 변경</button>
    <script>
        // 버튼 요소에 접근
        var button = document.getElementById("change-color");
        // 버튼에 클릭 이벤트 핸들러 등록
        button.addEventListener("click", function() {
            // 현재 배경색을 가져옴
            var currentColor = document.body.style.backgroundColor;
            // 배경색을 변경할 색상 배열
            var colors = ["red", "green", "blue", "yellow"];
            // 현재 색상과 다른 색상을 랜덤하게 선택
            var newColor;
            do {
                newColor = colors[Math.floor(Math.random() * colors.length)];
            } while (newColor === currentColor);
            // 배경색을 변경함
            document.body.style.backgroundColor = newColor;
        });
    </script>
</body>
</html>
```

이벤트

DOM과 이벤트 처리하기 위해서는 이벤트란 무엇인지 알아야 합니다. 이벤트란 웹 페이지에서 발생하는 사용자의 행동이나 시스템의 변화 등을 말합니다. 예를 들어,

- 마우스 클릭(click)
- 키보드 입력(keydown)
- 화면 크기 변경(resize)
- 페이지 로딩(load)

등이 있습니다.

이벤트 리스너는 특정 요소에서 발생한 이벤트에 반응하도록 설정하는 함수입니다. 이벤트 리스너는 다음과 같은 방식으로 등록하거나 제거할 수 있습니다.

- `element.addEventListener(type, listener)` : type에 해당하는 이벤트가 element에서 발생하면 listener 함수가 실행되도록 등록하는 메서드
- `element.removeEventListener(type, listener)` : type에 해당하는 이벤트가 element에서 발생해도 listener 함수가 실행되지 않도록 제거하는 메서드

이벤트 위임과 버블링은 이벤트 전파 과정에서 사용되는 개념입니다.

이벤트 버블링(event bubbling)은 하위 요소에서 발생한 이벤트가 상위 요소로 전파되는 현상을 말합니다. 예를 들어,

```
<div id="parent">
  <button id="child">Click me</button>
</div>
```

```
document.getElementById("parent").addEventListener("click", function () {
  console.log("Parent clicked");
});
document.getElementById("child").addEventListener("click", function () {
  console.log("Child clicked");
});
```

위 코드에서 `child` 버튼을 클릭하면 다음과 같은 결과가 나옵니다.

```
Child clicked
Parent clicked
```

child 버튼에서 발생한 `click` 이벤트가 parent `div`까지 전파되었기 때문에 그렇습니다.

이벤트 위임

이벤트 위임(event delegation)은 상위 요소에 이벤트 리스너를 등록하여 하위 요소들의 이벤트를 처리하는 방식을 말합니다. 예를 들어,

```
<ul id="list">
  <li>Apple</li>
  <li>Banana</li>
  <li>Cherry</li>
</ul>
```

```
document.getElementById("list").addEventListener("click", function (e) {
  console.log(e.target.textContent);
});
```

위 코드에서 list의 각 항목을 클릭하면 다음과 같은 결과가 나옵니다.

```
Apple
Banana
Cherry
```

list에 click 이벤트 리스너를 등록하여 각 li 요소의 click 이벤트를 처리하였습니다.

이렇게 DOM과 이벤트 처리에 대해 간단히 알아보았습니다. 더 자세한 내용은 다음 문서들을 참고하세요.

- DOM MDN

- DOM API MDN
- 이벤트 MDN