HTML에서 Javascript 사용하기

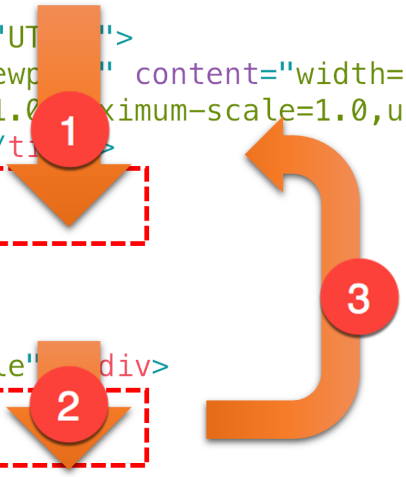
#01. Javascript 소스코드의 위치

JS 소스코드는 HTML 페이지의 어느 위치에나 올 수 있다.

대체로 가장 많이 사용하는 위치는 <head>태그가 끝나기 전과 <body>태그가 끝나기 전이다.

1) <head> 태그에 JS가 위치하는 경우

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UT    ">
    <meta name="viewp    " content="width=device-width,initial-scale=1.0,
    minimum-scale=1.0    ximum-scale=1.0,user-scalable=no">
    <title>jQuery</ti    >

    JS 코드

</head>

<body>
    <div id="console"    div>

    HTML 코드

</body>

</html>
```



1. 웹 브라우저가 HTML 파일을 실행하는 과정에서 JS소스를 인식하고 JS코드를 해석한다.
    - 만약 <script src="…"> 형태로 외부 스크립트 파일을 참조하는 처리가 있다면 해당 파일을 다운로드 받기 전까지 해석이 완료되지 않는다.
2. HTML 코드를 해석한다.
    - 해석이 끝나면 웹 브라우저 화면에 결과가 표시된다.
3. JS 코드를 실행하기 위해 다시 JS 코드가 정의된 블록으로 이동한다.

2) <body> 태그 마지막에 JS가 위치하는 경우

```
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta charset="UT    ">
    <meta name="viewp    " content="width=device-width,initial-scale=1.0,
    minimum-scale=1.0    ximum-scale=1.0,user-scalable=no">
    <title>jQuery</ti    >
    JS코드 없음
</head>

<body>
    <div id="console"    div>
    HTML+JS코드
</body>

</html>
```

1. JS 코드가 없기 때문에 웹 브라우저는 <body>안의 HTML 태그를 바로 해석하기 시작한다.
    - <body>안의 내용이 웹 브라우저 화면에 표시된다.
    - 즉, JS 파일을 다운로드 받거나 JS 코드를 해석하기 전에 사용자에게 UI를 먼저 제시할 수 있기 때문에 체감 실행 속도가 빨라진다.
2. HTML 코드의 해석을 먼저 끝낸 후에 JS 코드의 해석이 진행되고 실행된다.
    - JS코드를 위해 되돌아갈 필요가 없기 때문에 전체적으로 실행속도가 단축된다.

> JSP, PHP등과의 연계시 코드 구성이 복잡해 질 수 있기 때문에 JS를 <head>에 넣는 경우도 많다.

#02. 웹 페이지를 위한 Javascript 객체 모델

이름	설명
BOM (Browser Object Model)	웹 브라우저를 통해 실행될 때 Javascript가 갖게되는 기본 객체 구조. 모든 객체는 window 객체의 하위 객체로서 존재한다.
DOM (Document Object Model)	HTML 문서 구조(DOM)을 제어하기 위해 제공되는 객체 구조. BOM의 하위 요소중 하나이다.

#03. HTML 태그를 객체로 가져오기

1) 태그 이름으로 가져오기

특정 태그 모두를 가져오기 때문에 반한되는 객체는 항상 배열 형식이다.

반환되는 객체의 수가 너무 많기 때문에 잘 사용하지 않는다.

```
const 객체 = document.getElementsByTagName("태그이름")
```

2) ID값으로 가져오기

ID이름에 "#"을 붙이지 않는다.

ID값은 HTML 문서 안에서 고유한 요소이므로 항상 단일 객체로 반환된다.

```
const 객체 = document.getElementById("ID이름");
```

3) CSS 클래스 이름으로 가져오기

클래스 이름에 점을 붙이지 않는다.

특정 클래스가 적용된 모두를 가져오기 때문에 반한되는 객체는 배열 형식이다.

```
const 객체 = document.getElementsByClassName("CLASS이름");
```

4) CSS 선택자로 가져오기 (권장)

단일 객체

CSS 선택자의 형태에 상관 없이 단일 객체로 반환된다.

만약 동일한 셀렉터를 적용받는 요소가 두 개 이상인 경우 가장 첫 번째 요소만 반환한다.

```
const 객체 = document.querySelector("CSS선택자");
```

복수 객체

CSS 선택자의 형태에 상관 없이 복수 객체를 배열로 반환한다.

```
const 객체 = document.querySelectorAll("CSS선택자");
```

선택자 사용시 참고

- JS에서 CSS의 셀렉터를 사용한다고 해서 반드시 그 셀렉터가 `<style>` 블록에 정의되어 있어야 하는 것은 아니다.
- JS와 CSS모두 셀렉터를 사용하기 때문에 작성 규칙을 정해두지 않으면 혼란스러워 진다.
    - ex) CSS는 class와 가상클래스 기반으로만 구성하고 Javascript에서는 ID값을 주로 사용.
    - ex) JS에서는 한번에 여러 개의 요소를 제어해야 하는 상황에 한해서만 class 기반을 사용하고 대부분의 경우는 id나 태그의 속성 기반으로만 사용

#04. HTML 태그안에 내용 넣기

Javascript 가져온 HTML 객체는 innerHTML 이라는 속성값을 갖는다.

이 값에 할당한 내용은 실제 HTML의 시작태그와 끝 태그 사이에 적용된다.

```
const 객체 = document.querySelector("#hello");
객체.innerHTML = "Hello World";
```

```
<div id="hello"> Hello World </div>
                 -----------
                   innerHTML
```