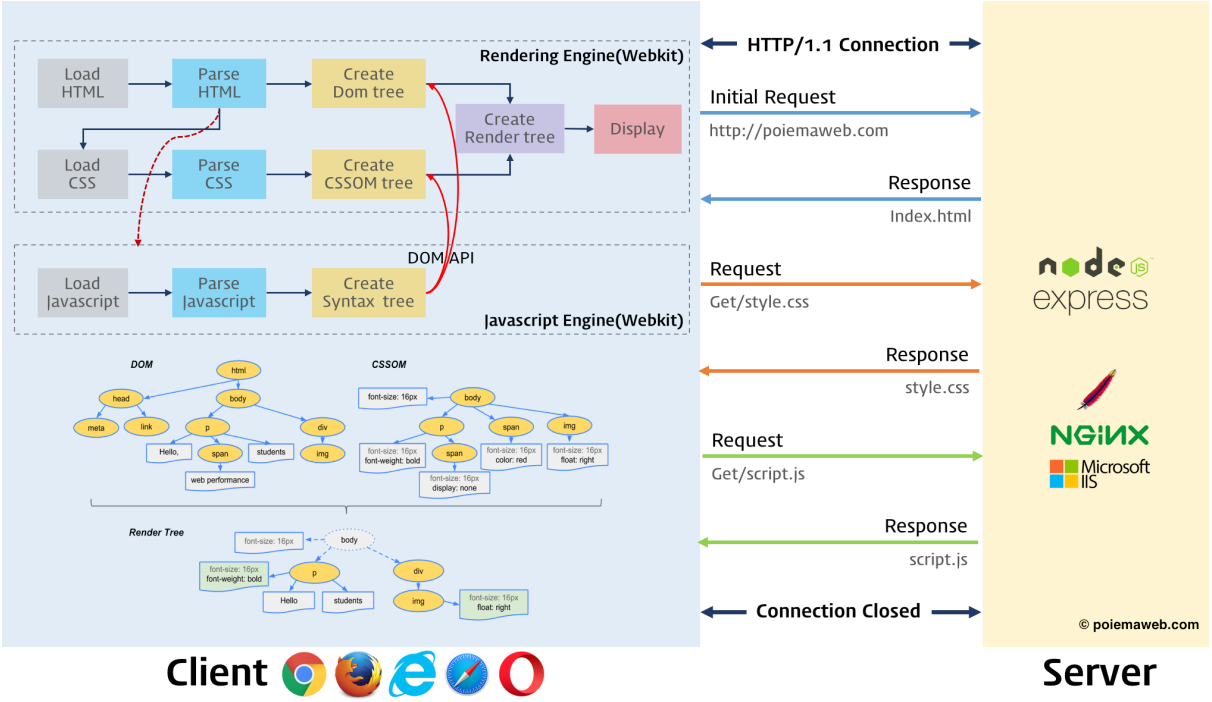
브라우저 동작 원리

구글의 Chrome V8 자바스크립트 엔진으로 빌드된 자바스크립트 런타임 환경(Runtime Environment)인 **Node.js**의 등장으로 자바스크립트는 웹 브라우저를 벗어나 서버 사이드 애플리케이션 개발에서도 사용되는 범용 개발 언어가 되었다. 하지만 자바스크립트가 가장 많이 사용되는 분야는 역시 웹 브라우저 환경에서 동작하는 웹 페이지/애플리케이션이다.

대부분의 프로그래밍 언어는 운영체제(Operating System, OS) 위에서 실행되지만 웹 애플리케이션의 자바스크립트는 브라우저에서 HTML, CSS와 함께 실행된다. 따라서 브라우저 환경을 고려할 때 보다 효율적인 자바스크립트 프로그래밍이 가능하다.

브라우저의 핵심 기능은 사용자가 참조하고자 하는 웹페이지를 서버에 요청(Request)하고 서버의 응답(Response)을 받아 브라우저에 표시하는 것이다. 브라우저는 서버로부터 HTML, CSS, Javascript, 이미지 파일 등을 응답받는다. HTML, CSS 파일은 렌더링 엔진의 HTML 파서와 CSS 파서에 의해 파싱(Parsing)되어 DOM, CSSOM 트리로 변환되고 렌더 트리로 결합된다. 이렇게 생성된 렌더 트리를 기반으로 브라우저는 웹페이지를 표시한다.



자바스크립트는 렌더링 엔진이 아닌 자바스크립트 엔진이 처리한다. HTML 파서는 script 태그를 만나면 자바스크립트 코드를 실행하기 위해 DOM 생성 프로세스를 중지하고 자바스크립트 엔진으로 제어 권한을 넘긴다. 제어 권한을 넘겨 받은 자바스크립트 엔진은 script 태그 내의 자바스크립트 코드 또는 script 태그의 src 어트리뷰트에 정의된 자바스크립트 파일을 로드하고 파싱하여 실행한다. 자바스크립트의 실행이 완료되면 다시 HTML 파서로 제어 권한을 넘겨서 브라우저가 중지했던 시점부터 DOM 생성을 재개한다.

이처럼 브라우저는 **동기(Synchronous)**적으로 HTML, CSS, Javascript을 처리한다. 이것은 script 태그의 위치에 따라 블로킹이 발생하여 DOM의 생성이 지연될 수 있다는 것을 의미한다. 따라서 script 태그의 위치는 중요한 의미를 갖는다.

body 요소의 가장 아래에 자바스크립트를 위치시키는 것은 좋은 아이디어이다. 그 이유는 아래와 같다.

- HTML 요소들이 스크립트 로딩 지연으로 인해 렌더링에 지장 받는 일이 발생하지 않아 페이지 로딩 시간이 단축된다.

- DOM이 완성되지 않은 상태에서 자바스크립트가 DOM을 조작한다면 에러가 발생한다.