

자바스크립트 브라우저 렌더링

렌더링이란 HTML, CSS, JavaScript 등으로 이루어진 문서를 브라우저에서 그래픽 형태로 출력하는 과정을 말합니다. 우리가 자주 보는 브라우저는 렌더링 성능에 따라 사용자가 느끼는 체감속도를 개선할 수 있으므로 렌더링 과정을 최적화하여 성능 개선을 해야합니다.

렌더링 엔진

대부분의 브라우저는 렌더링을 수행하는 렌더링 엔진을 가지고 있습니다. 모든 브라우저가 같은 렌더링 엔진을 사용하지는 않습니다. 파이어폭스는 게코, 사파리는 웹킷, 크롬은 웹킷을 사용하다가 웹킷을 Fork하여 블링크 엔진을 자체적으로 구현하여 사용하고있습니다.

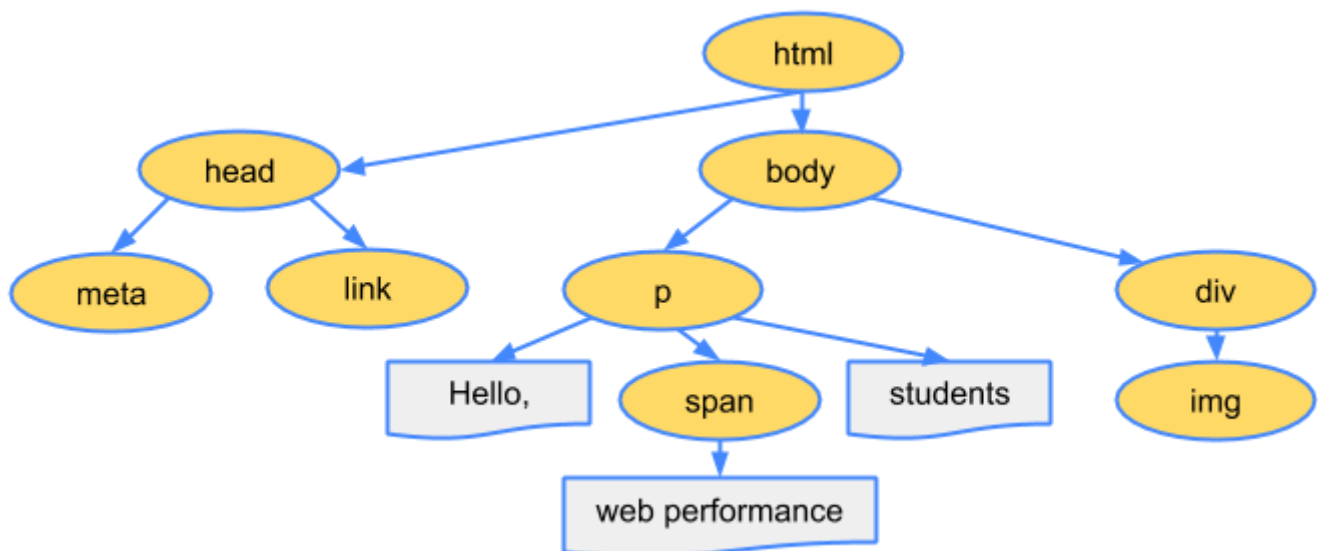
렌더링 과정

1. DOM(Document Object Model), CSSOM(CSS Object Model) 생성

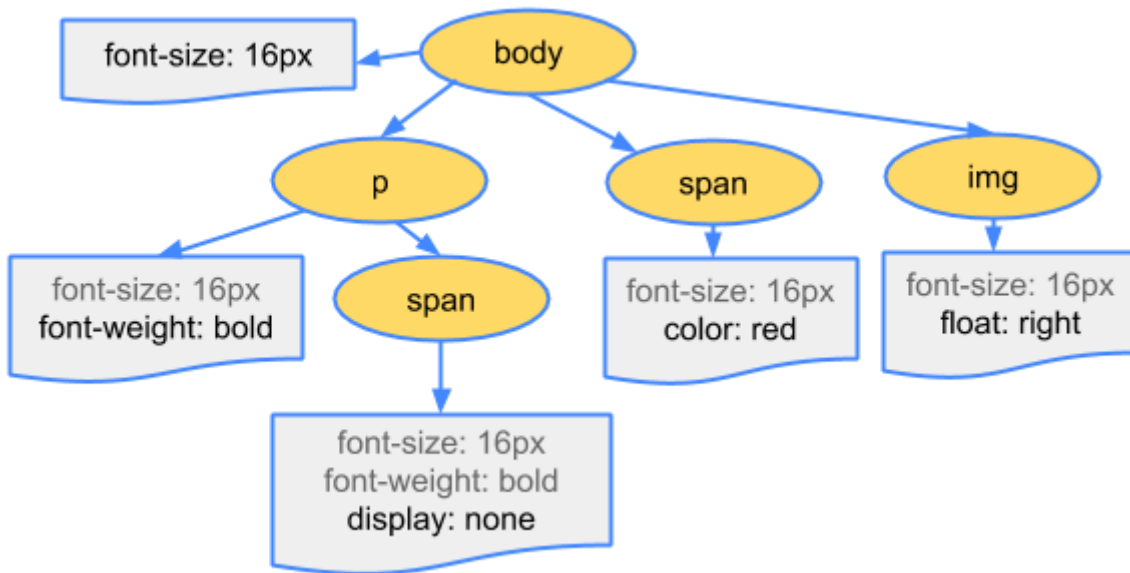
브라우저가 페이지를 렌더링하려면 DOM 및 CSSOM 트리를 생성해야 한다.

HTML 및 CSS를 가능한 빨리 브라우저에 제공해야하며 HTML 마크업은 DOM, CSS 마크업은 CSSOM으로 변환된다. DOM과 CSSOM은 서로 독립적인 데이터 구조이다. Chrome DevTools Performance를 사용하면 DOM및 CSSOM의 생성 및 처리 비용을 수집 및 점검할 수 있다.

DOM 트리 예제 이미지



CSSOM 트리 예제 이미지



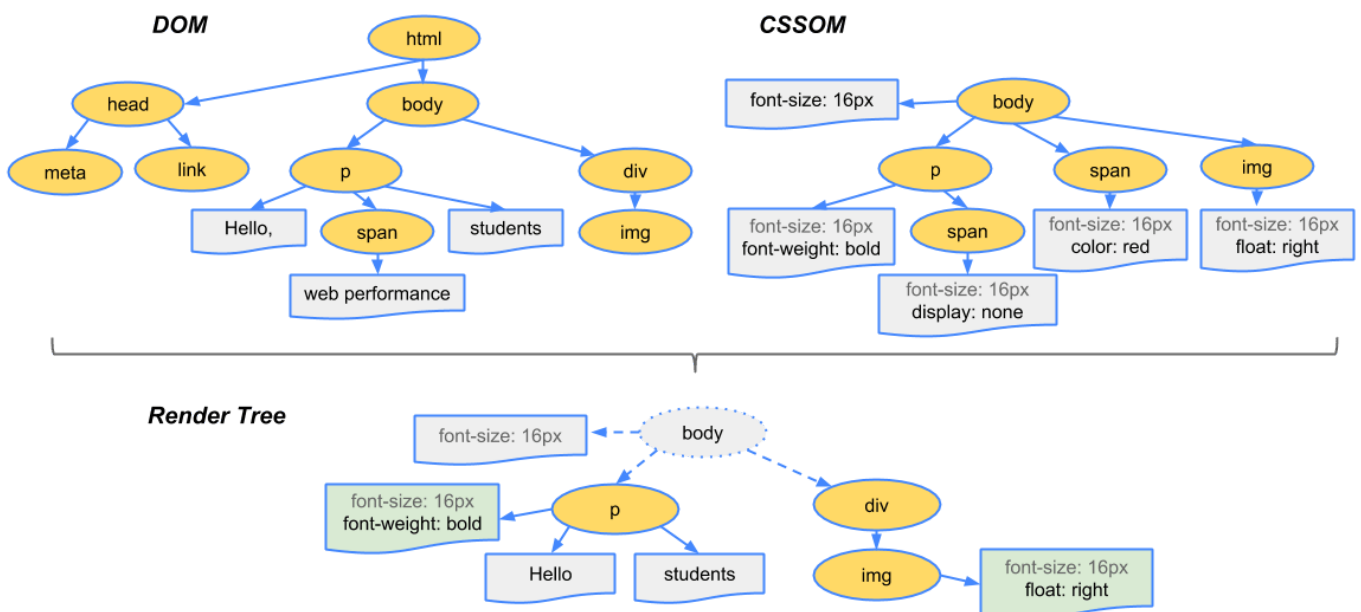
(이미지 출처 : <https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-tree-construction?hl=ko>)

렌더링 엔진은 더 나은 사용자 경험을 위해 가능한 빠르게 내용을 표시하게 만들어 졌는데, 따라서 모든 HTML, CSS파싱이 끝나기도 전에 이후 과정을 수행하여 미리 사용자에게 보여줄 수 있는 내용들을 출력한다.

2. 렌더 트리 생성

DOM Tree와 CSSOM Tree가 만들어 지면 이 두개의 트리를 이용하여 Render Tree를 생성합니다. Render Tree는 스타일 정보가 설정되어 있으며 실제 화면에 표현되는 노드들로만 구성됩니다.

렌더트리 예제 이미지



여기서 실제 화면에 구성된다는 말은 **display:none**같은 스타일 속성이 설정된 노드는 화면에 어떠한 공간도 차지하지 않기때문에 Render Tree를 만드는 과정에서 제외됩니다. (visibility:invisible은 display:none과 비슷하게 동작하지만, 공간은 차지하고 요소만 보이지 않게 하기 때문에 Render Tree에 포함됩니다.)

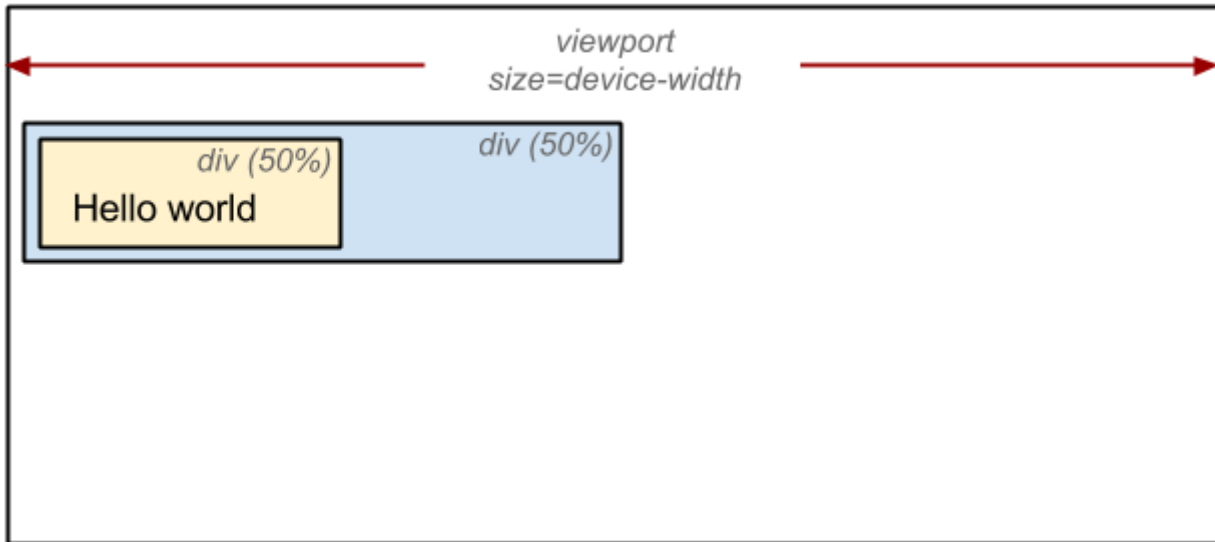
Render Tree가 생성되었으므로 Layout단계로 넘어갑니다.

3. Layout

Layout 단계는 브라우저의 뷰포트(Viewport) 내에서 각 노드들의 정확하 위치와 크기를 계산합니다. 즉 Render Tree 노드들이 가지고 있는 스타일과 속성에 따라 브라우저 화면의 어느 위치에 어느 크기로 출력될지 계산하는 단계입니다.

Layout 단계를 통해 \$, vh, vw와 같은 상대적인 위치, 크기 속성은 실제 화면에 그려지는 pixel단위로 변환됩니다.

Viewport 상대적인 요소 연산



여기서 뷰포트(Viewport)란 그래픽이 표시되는 브라우저 영역, 크기를 말합니다. 뷰포트는 모바일의 디스플레이 크기, 컴퓨터 모니터의 브라우저 창의 크기에 따라 달라집니다. 화면에 그려지는 각 요소들의 위치와 크기는 상대적으로 계산하여 그려지는 경우가 많아 뷰포트의 크기가 달라질 경우 매번 계산을 다시 해야합니다.

4. Paint

Layout 계산이 완료 되면 요소들을 실제 화면에 그리게 됩니다. 이전 단계에서 이미 요소들의 위치와 크기, 스타일 계산이 완료된 Render Tree를 이용해 실제 픽셀 값을 채워넣게 됩니다. 이 때 텍스트, 색, 이미지, 그림자 효과 등이 모두 처리되어 그려집니다.

이 때 처리해야 하는 스타일이 복잡할수록 Paint 단계에 소요되는 시간이 늘어나게 됩니다. 간단한 예시로 단순한 단색의 경우 paint 속도가 빠르지만 그라데이션이나 그림자 효과등은 painting 소요시간이 비교적 오래 소요됩니다.

5. Reflow, Repaint

모든 렌더링 과정이 끝난 후 자바스크립트를 이용하여 DOM, CSSOM을 수정하여 요소의 크기나 위치 등 레이아웃의 수치를 수정하면 그 동작에 영향을받는 자식노드나 부모 노드들을 포함하여 Layout 과정을 다시 수행하게 됩니다. 이 때 Render Tree가 다시 생성 되어서는데 이 과정을 Reflow라고 합니다.

Reflow가 발생하는 경우

- 페이지 초기 렌더링 시 (최초 Layout)
- 윈도우 리사이징 시 (ViewPort 크기 변경)
- 노드 추가/제거 시
- 요소의 위치, 크기 변경시

- 텍스트 내용 변경 및 이미지 크기 변경시

이후 Repaint 과정을 거쳐 Reflow되어 다시 계산된 Render Tree를 다시 화면에 그려주어야 하며 이 과정을 Repaint라고 합니다. 색상 변경과 같이 요소의 위치나 크기가 변하지 않는 경우 Reflow과정을 건너뛰며 Repaint만 수행하게 됩니다.

서버에 요청한 이미지가 동시에 렌더링 되는 이유?

위에서 알아보았듯이 브라우저에 렌더링 될때는 각 요소마다 렌더링되는것이 아니라 DOM트리와 CSSOM트리를 다시 구성하여 새로운 Render tree를 만들고 Reflow, Repaint 과정을 거쳐 새롭게 렌더링 되어지게 되므로 비슷한 시간T에 응답을 받아 DOM요소에 넣어주게 되면 동시에 렌더링이 된다.