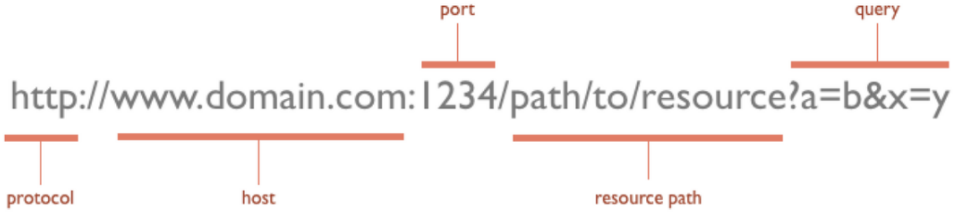
2023.05.07 브라우저의 렌더링 과정 (동작 과정)

브라우저의 주요 구성 요소

1. 사용자 인터페이스: 주소 표시줄, 이전/다음 버튼, 북마크 메뉴 등, 요청한 문서를 표시하는 창을 제외한 나머지 모든 부분입니다.
2. **렌더링 엔진**: 요청한 컨텐츠를 표시하기 위한 엔진으로, HTML, XML 문서나 이미지, PDF 등을 표시하며, Webkit(사파리, 크롬), Gecko(파이어폭스) 등이 있습니다.
    a. 통신: 네트워크 호출에 사용되는 통신입니다.
    b. UI 백엔드: 기본 input 등, 개발자가 명시하지 않은 일반적인 인터페이스입니다.
    c. 자바스크립트 해석기: 자바스크립트 코드를 해석하고 실행하는 부분입니다.
3. 브라우저 엔진: 사용자 인터페이스와 렌더링 엔진 사이의 동작을 제어하는 엔진입니다.
    a. 자료 저장소: 쿠키 등의 자료를 저장하는 로컬 저장소입니다.

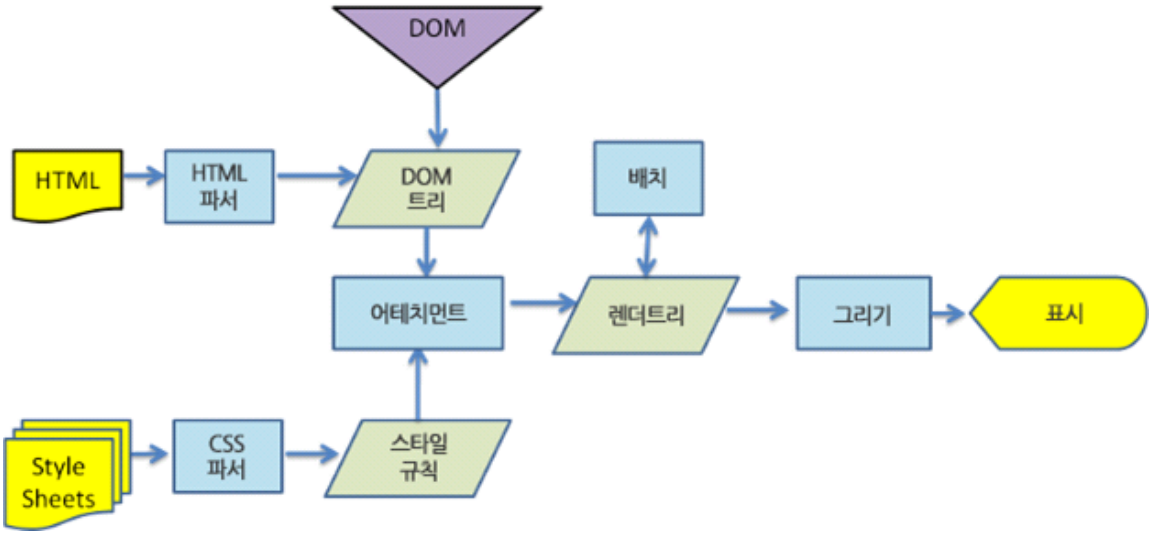
리소스 요청 및 응답 수신

1. 가장 먼저 브라우저는 주소 표시줄에 입력된 주소를 통해 서버에 필요한 **리소스를 요청**하고 이에 대한 **응답을 수신**합니다.
    a. url의 host명이 DNS 서버를 통해 실제 주소인 IP 주소로 변환되고, 이에 해당하는 서버에 요청을 보냅니다. DNS 서버는 사용자가 쉽게 인식할 수 있는 host명과 실제 주소인 IP 주소를 매핑하고 있는 서버입니다.

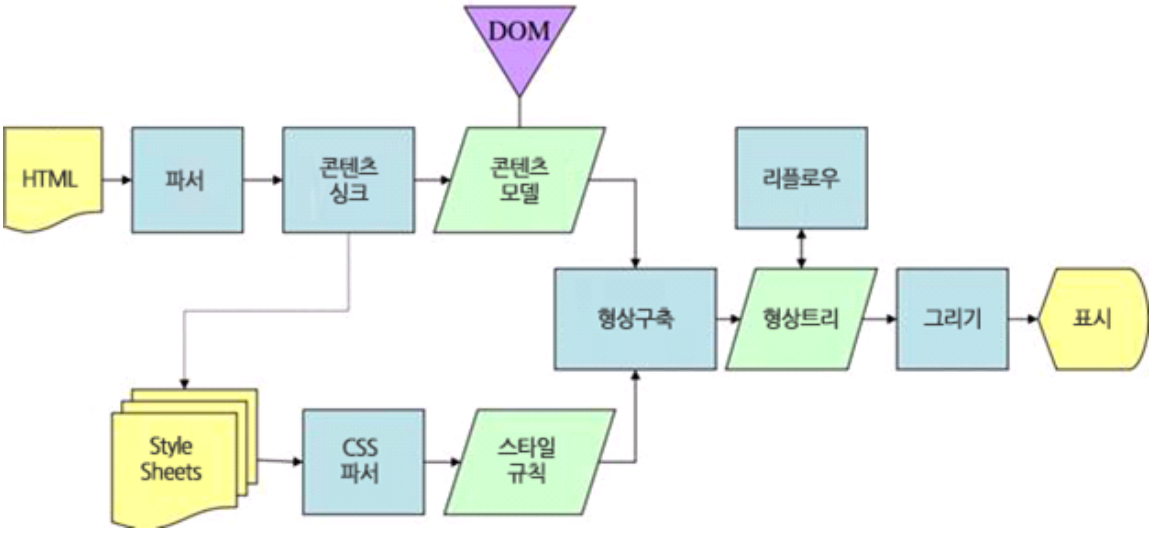


브라우저의 렌더링 과정(렌더링 엔진)

- Webkit 엔진



- Gecko 엔진



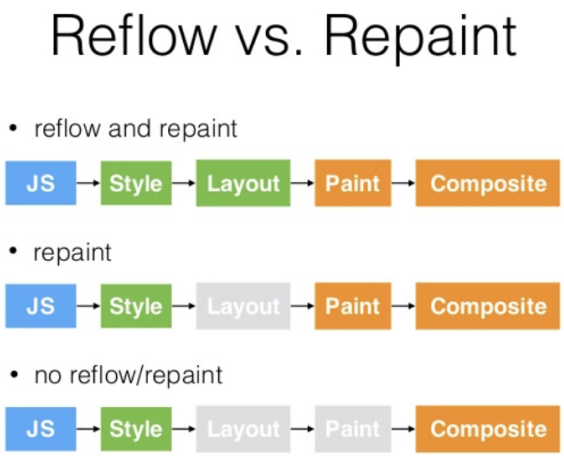
- 브라우저에 따라 용어에 약간 차이가 있으나, 아래와 같은 과정을 따릅니다.
  1. **Load**: HTML, CSS, JS, 이미지, 폰트 등의 **리소스를 로드**하는 과정입니다.
  2. **Parsing**: HTML과 CSS 파일을 해석하여 각각 **DOM Tree와 CSSOM Tree를 구축**하는 과정입니다.

a. 이 과정에서 자바스크립트 파일이 존재한다면, 자바스크립트 엔진에 의해 Parsing됩니다. 이 때, 브라우저의 제어권은 렌더링 엔진에서 자바스크립트 엔진으로 이동하므로, 자바스크립트의 해석이 완료된 후에 HTML과 CSS의 해석이 재개됩니다.

b. 자바스크립트 엔진은 리소스를 이용해 **Abstract Syntax Tree(AST)를 구축**하고 기계어로 변환합니다.

c. AST는 인터프리터가 해석할 수 있는 바이트 코드로 변환되고, 인터프리터에 의해 실행된다.

3. **Styling**: DOM Tree와 CSSOM Tree를 매칭시켜 실제로 화면에 그려질 **Render Tree(Gecko: Frame Tree)를 구성**합니다.
4. **Layout**: 각 Node의 **배치와 크기를 계산**하여 Render Tree에 반영합니다.
5. **Paint**: 계산된 값들을 실제 픽셀로 변환하고 레이어를 생성하여 화면에 그립니다.
6. **Reflow & Repaint**: 자바스크립트에 의해 DOM이나 CSSOM이 변경되었을 때, 이를 **다시 계산하는 것**을 Reflow, **다시 화면에 그리는 것**을 Repaint라고 합니다.
   a. Reflow가 발생하면 Repaint가 필연적으로 발생하지만, CSS만 변경되는 경우에는 Repaint만 발생하기도 합니다.



b. Reflow는 레이아웃을 다시 계산해야 하므로 성능 최적화를 위해서는 이를 최소화할 수 있도록 코드를 작성하는 것이 좋습니다. 다만, 모던 웹 페이지들은 대체로 인터렉션이 복잡하고 다양하기 때문에 Reflow는 필연적으로 발생하게 되므로, 개인적으로는 아래와 같은 몇 가지 가이드라인을 염두하고 적용하는 정도로만 생각하면 될 것 같다고 판단하고 있습니다.

i. DOM의 **심도 줄이기**: DOM의 심도가 깊을수록 최하위 요소까지 도달하는 데 관련된 Node의 수가 많아집니다. 이는 즉, 특정 Node의 변경에 연관되어 변경의 여지가 있는 Node의 수가 많아진다는 의미이므로 Reflow에 오랜 시간이 걸릴 수 있습니다.

ii. CSS 최소화 및 사용하지 않는 CSS 삭제

iii. **불필요하고 복잡한 CSS Selector, 특히 하위 요소 Selector 사용 피하기**: Selector를 일치시키기 위해 더 많은 처리가 필요하게 됩니다.

iv. 애니메이션 등, 복잡한 렌더링 변경이 필요한 경우 **문서의 흐름 밖에서 변경**하기: 이 경우, `position`을 `absolute` 또는 `fixed`로 설정해야 다른 요소들에 영향을 주지 않으므로 Reflow를 최소화할 수 있습니다.

v. **레이아웃 변경 피하기**: `width`나 `height` 등의 속성은 레이아웃에 관여하는 속성이므로, 다른 요소들에도 영향을 미칩니다. 이는 곧 레이아웃 계산에 더 많은 리소스를 사용해야 한다는 의미입니다. `visibility`나 `display` 역시 마찬가지입니다. 각각 `transform`, `opacity`로 대체가 가능한 경우에는 대체하는 것이 좋습니다.

vi. 오래된 레이아웃 모델 지양하기: `float`와 같은 오래된 레이아웃 모델은 `flexbox`나 `grid`보다 reflow 시간이 깁니다. 반드시 필요한 경우가 아니라면 대체하는 것이 좋습니다.

7. Composite: 생성된 레이어를 합성하여 실제 화면에 그립니다.