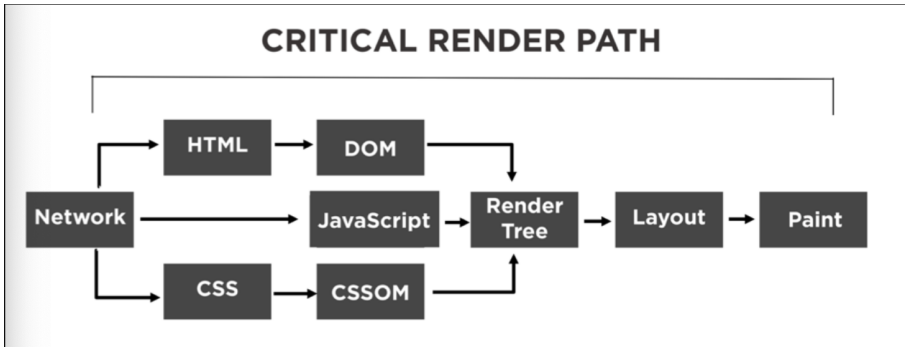
[JS Deep Dive] 38장 . 브라우저의 렌더링 과정

Status 진행 중

브라우저의 렌더링 과정



1.요청	2.트리생성	3.layout	4.페인팅
브라우져→ 서버에 필요한 리소스 요청	브라우져의 렌더링 엔진 → html,css ⇒ DOM, CSSOM = 렌더 트리 생성	브라우져의 자바스크립트 엔진 → js의 경우 DOM API를 사용해 렌더트리 변경, 리플로우 리페인트	렌더트리를 기반으로 HTML 요소의 레이아웃 계산, HTML 요소 페인팅

1 . 요청

> ▫**브라우져는 렌더링에 필요한 리소스(html , css , 자바스크립트, 이미지, 폰트, 파일등)을 요청하고 서버로부터 응답을 받는다.**

❗**주체 : 브라우져의 렌더링 엔진**

▫ 실행 플로우

1.주소창에 URL입력 ⇒ 2.서버에 리소스 요청 ⇒ 3.서버가 리소스를 응답 ⇒ 4.리소스 파싱

🔘주소창에 URL 입력

- 서버에 요청을 전송하기 위해 브라우져는 주소창을 제공함
- URL 호스트이름이 DNS를 통해 IP로 변환

🔘 서버에 리소스 요청

- DNS를 통해 변환된 해당 IP주소를 갖는 서버에 리소스 요청
- 렌더링에 필요한 리소스는 모두 서버에 존재

🔘 서버가 리소스를 응답

- 일반적으로 서버는 루트 요청에 대해 암묵적으로 index.html을 클라이언트에 응답
- index.html 이 아닌 다른 정적파일 요청시 요청할 정적 파일의 경로와 파일이름을 path 에 기술하여 서버에 요청함.
- 반드시 정적파일만 요청이아니라 정적/ 동적 데이터를 요청 가능

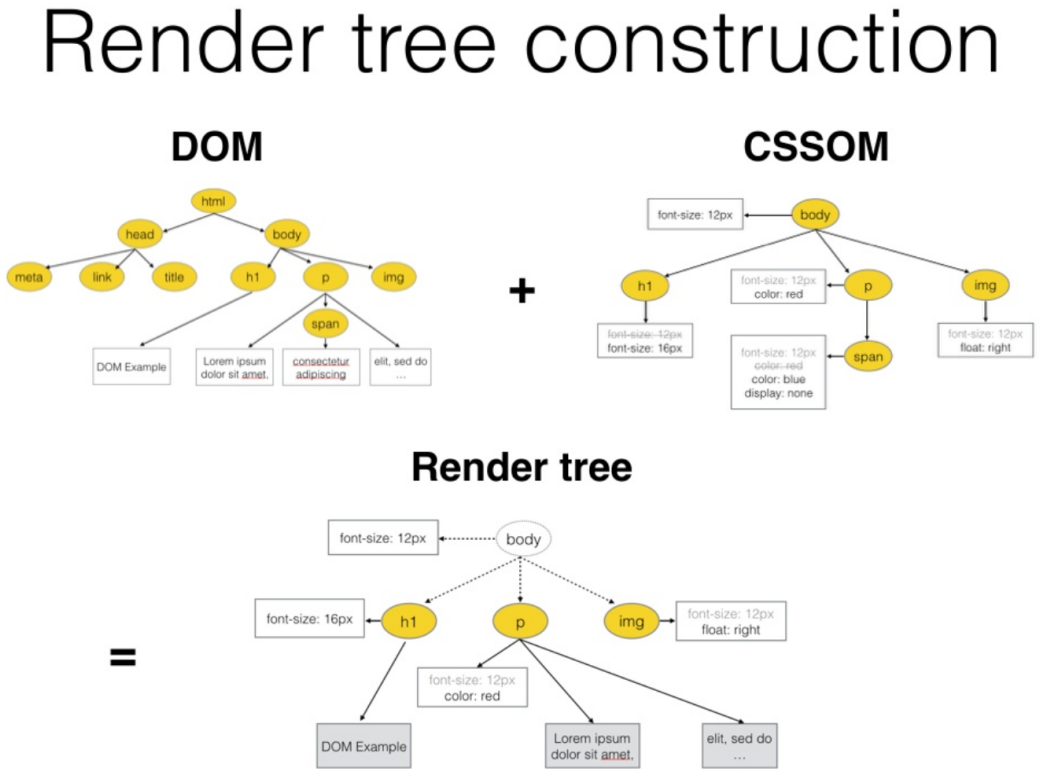
🔘 리소스 파싱

- HTML을 파싱하는 도중, 외부 리소스를 로드 하는 태그 `link , img, scrip` 를 만나면 HTML 파싱을 일시 중단하고 해당 리소스 파일을 서버로 요청함.

---

2 . 트리생성

> ▫브라우져의 렌더링 엔진은 서버로부터 응답된 html과 css 를 파싱하여 DOM과 CSSOM 을 생성하고 이들을 결합하여 렌더 트리를 생성



- 렌더트리는 브라우져 화면에 렌더링 되는 노드만으로 구성, 각 html 요소의 레이아웃을 계산하는데 사용, 페인팅 처리에 입력됨

❗주체 : 브라우져의 렌더링 엔진

▫ 실행 플로우

DOM 생성 (HTML 파일) :

파일 → 메모리 → 바이트 → 문자열(인코딩) → 토큰 →객체 → 노드생성 → DOM

CSSOM 생성 (html 내 link, style 태그) :

파일 → 메모리 → 바이트 → 문자열(인코딩) → 토큰 →객체 → 노드생성 → CSSOM

🔘DOM 생성

- 서버에 존재하던 html파일이 응답
- 서버가 응답한 HTML 은 브라우져에 시각적인 픽셀로 렌더링하기위해서는 브라우져가 이해할수 있는 자료구조로 변환하여 메모리에 저장해야함.
- 파일을 읽어 메모리에 저장
- 메모리에 저장된 바이트를 인터넷을 경유하여 응답
- 브라우져는 서버가 응답한 바이트를 meta 태그에 의해 지정된 인코딩 방식을 기준으로 문자열로 변환
- 문자열로 변환된 html을 문법적 의미를 갖는 코드의 최소 단위인 토큰으로 분해함.

- 각 토큰들 → 객체 → 노드생성 을 거쳐 노드는 DOM을 구성하는 기본 요소가 됨.
- html은 요소들끼리 중첩관계를 가지므로 부모자식관계가 형성된다.
- html 요소간의 부자관계를 반영하여 모든 노드들을 트리 자료 구조로 구성 == DOM 이라고 부름

🔘 CSSOM

- DOM 생성을 일시 중단
- 지정된 css 파일을 서버에 요청하여 로드한 css ,style 파일을 html과 동일한 파싱과정을 거쳐 CSSOM을 생성
- 바이트 → 문자 → 토큰 → 노드 → CSSOM
- CSSOM 생성 완료후 중단지점으로 돌아가 다시 DOM 생성을 재개

---

3 . 레이아웃

> ▫**Render Tree 루트 부터 시작하여 모든 객체들에 대한 위치와 크기를 계산**
>
> ▫ **아래의 이유로 리 플로우가 실행 될 수 있음.**

✅ 리플로우

▫자바스크립트는 DOM API를 통해 DOM이나 CSSOM을 변경가능→ 변경된 DOM과 CSSOM은 다시 렌더 트리로 결합, 변경된 렌더트리를 기반으로 레이아웃을 다시 계산함

❗ 주체 : 자바스크립트 엔진

▫실행 플로우

자바스크립트 코드에서 DOM 조작 ⇒ 자바스크립트 파싱 ⇒ AST(추상적 구문트리) 생성 ⇒ 바이트코드로 변환 ⇒ DOM or CSSOM 변경 ⇒ 렌더트리에 반영

🔘 DOM 조작

- 자바스크립트 코드에서 DOM API를 사용하면 이미 생성된 DOM을 동적으로 조작할수 있다.
- DOM 을 파싱하다가 script, 자바 스크립트 코드를 만나면 DOM 생성을 일시 중단함.

🔘 자바스크립트 파싱

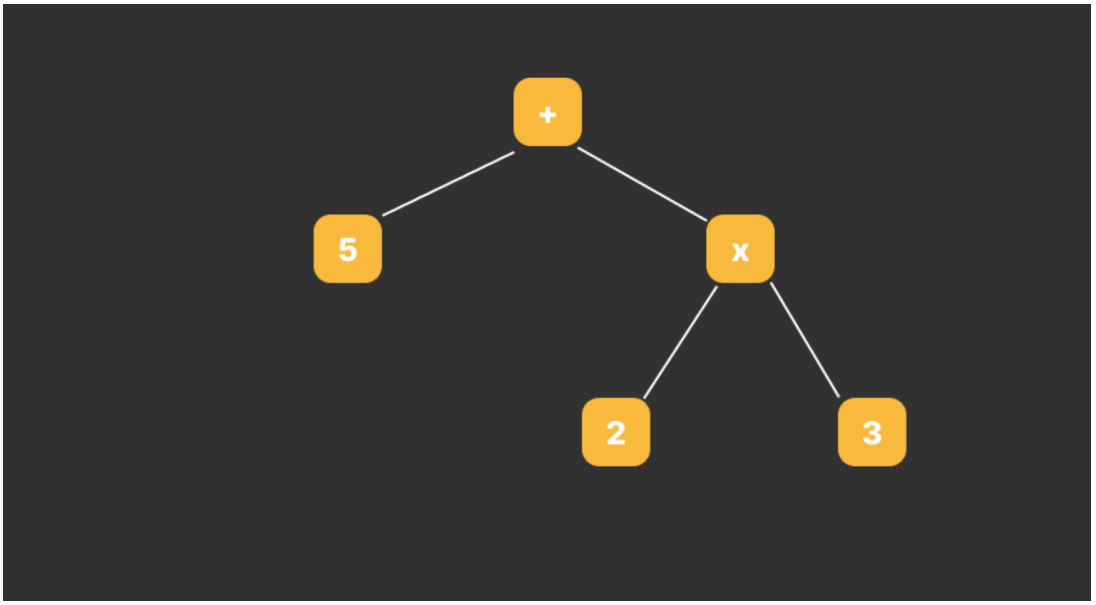
- 해당 자바스크립트 파일을 서버에 요청하고, 파싱을 위해 자바스크립트 엔진에 제어권을 넘김

🔘 AST 생성

- 자바스크립트 엔진은 코드를 해석하여 AST (추상적 구문 트리)를 생성

▼ 추상적 구문 트리

https://ww8007-learn.tistory.com/14



🔘 바이트 코드로 변환

- AST를 기반으로 인터프리터가 실행 할수 있는 중간 코드인 바이트 코드를 생성하여 실행
- 자바스크립트 파싱이 종료되면 렌더링 엔진으로 다시 제어권을 넘겨 DOM이 중단된 지점부터 생성을 재개한다.

🔘 DOM or CSSOM 변경

- 변경된 렌더트리를 기반으로 다시 레이아웃을 계산함 ⇒ 리플로우

---

4 . 페인팅

> **계산된 레이아웃을 바탕으로 브라우져 화면에 HTML 요소를 페인팅**

🔘 리페인트

리플로우 에서 재결합된 렌더 트리를 기반으로 다시 페인트

리플로우와 리페인트가 반드시 순차적으로 동시에 실행되는 것은 아님.

자바스크립트 파싱에 의한 HTML 파싱 중단

> ▫ **렌더링 엔진과 자바스크립트 엔진은 직렬적으로 파싱을 수행함.**
>
> ▫ **자바스크립트 코드에서 DOM API를 사용할 경우 DOM, CSSOM이 이미 생성되어 있어야함.**

✅ body 요소의 가장 아래에 script 태그를 위치시키는 이유

- DOM 이 완성되지 않은 상태에서 자바스크립트가 DOM API를 사용하여 DOM을 조작하면 에러가 발생할 수 있다.
- 자바스크립트 로딩/파싱/실행 으로 인해 DOM 파싱이 중단되지 않아 페이지 로딩 시간이 단축됨.

✅ async / defer 로 DOM 생성 중단 해결하기

- src 어트리 뷰트가 없는 인라인 자바스크립트에는 사용할수 없다.
- HTML 파싱과 외부 자바스크립트 파일의 로드가 비동기적으로 동시에 진행된다.

🔘 async

- HTML 파싱과 외부 자바스크립트 파일의 로드가 비동기 적으로 동시에 진행된다.
- 자바스크핍트의 파싱과 실행은 자바스크립트 파일의 로드가 완료된 직후 진행되며, 이땐 html파싱이 중단됨.
- 여러개의 async 어트리뷰트를 지정하면 script 태그의 순서와 상관없이 로드가 완료된 자바스크립트부터 먼저 실행되므로 순서가 보장되지 않으므로 순서 보장이 필요한 script 태그에는 async 어트리 뷰트를 지정하지 않아야 한다.

🔘 defer

- html 파싱과 외부 자바스크립트 파일의 로드가 비동기 적으로 동시에 진행된다.
- 자바스크립트의 파싱과 실행은 html 파싱이 완료된 직후, 즉 DOM 생성이 완료된 직후 진행됨, 따라서 DOM 생성이 완료된 이후 실행되어야할 자바스크립트에 유용

렌더링 성능 최적화

✅ CSS 및 JavaScript 최적화

- CSS 코드를 압축하고, 중복된 스타일 코드를 제거한다.
- JavaScript 코드를 압축하고, 불필요한 코드를 제거한다.
- JavaScript 코드를 지연 로딩하거나 비동기 로딩을 이용하여 페이지 로딩 시간을 최적화한다.

✅ 이미지 최적화

- svg를 사용한다. https://velog.io/@oneook/웹-환경에서-SVG-더-잘-이해하고-사용하기
- 불필요한 이미지를 제거하거나, CSS 스프라이트 기술을 이용하여 여러 이미지를 하나의 이미지 파일로 통합한다.
- Lazy Loading 기술을 이용하여 페이지 스크롤링에 따라 이미지를 로딩한다.

✅ 레이아웃 최적화

- CSS Flexbox 또는 CSS Grid를 이용하여 요소를 배치한다.
- 요소의 크기와 위치를 계산할 때, 불필요한 계산을 줄이는 최적화 기술을 이용한다.
- 불필요한 CSS 요소를 제거하거나, CSS 속성을 줄인다.

참고: https://velog.io/@tnehd1998/주소창에-www.google.com을-입력했을-때-일어나는-과정

https://opendeveloper.tistory.com/entry/FrontEnd-%EC%A7%80%EC%8B%9D%EB%B8%8C%EB%9D%BC%EC%9A%B0%EC%A0%80%EB%A0%8C%EB%8D%94%EB%A7%81-%EC%9B%90%EB%A6%AC%EC%99%80-%EC%88%9C%EC%84%9C%EC%84%B1%EB%8A%A5-%EC%B5%9C%EC%A0%81%ED%99%94-%EA%B3%A0%EB%A0%A4%EC%82%AC%ED%95%AD

https://velog.io/@jeju_daun/브라우저-브라우저-렌더링-방식