[개발환경] 브라우저 동작 방식, 브라우저 기본 구조

구현 어려움 정도	개념
구현 완료	☑
기간	@December 10, 2024
정리 완료	☑

브라우저?

사용자가 선택한 자원을 서버에 요청하고 브라우저에 표시하는 것

이 때 자원은 HTML 문서, PDF, 이미지 등 다양한 형태가 있는데 자원의 주소는 URL(Uniform Resource Identifier)에 의해 정해짐

- 예) 크롬, 파이어폭스, 사파리 등

브라우저의 동작 과정

사용자가 웹사이트를 요청하고, 그 사이트가 화면에 표시되기까지의 과정

1. URL 입력 및 요청

사용자가 브라우저 주소창에 URL을 입력하거나 링크를 통해 들어가면, 브라우저는 해당 URL에 대한 HTTP 요청을 보냄

DNS 조회를 통해 도메인(URL)을 IP 주소로 변환하고 해당 서버와 연결을 시도

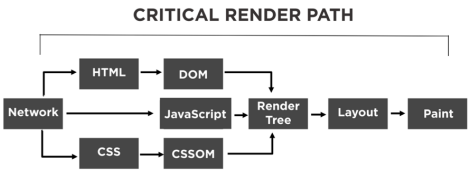
2. 서버 응답

서버는 요청에 따라 HTML, CSS, Javascript, 이미지와 같은 리소스를 브라우저로 전송

3. 렌더링

서버로부터 HTML, CSS, Javascript 등 작성한 파일을 받아 브라우저에 뿌려주는 것

렌더링 순서



1. 불러오기

로더(Loader)가 서버로부터 전달받은 리소스 스트림을 읽는 과정

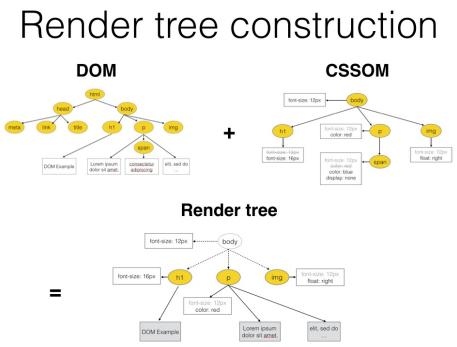
2. DOM, SCCOM 생성

웹 엔진의 HTML/XML 파서가 문서를 파싱해 DOM 트리,

CSS 파서가 CSSOM 트리를 생성

3. 생성된 DOM과 CSSOM으로 렌더 트리 생성

DOM Tree + CSSOM Tree, 렌더링에 필요한 노드만 선택해 페이지를 렌더링하는데 사용



4. CSS, 레이아웃

렌더 트리를 토대로 그려질 노드와 스타일, 크기를 계산

렌더링 트리에서 위치, 크기 등을 알 수 없기 때문에 객체들에게 위치 크기 등을 정해주는 과정, css는 선택자에 따라서 적용되는 태그가 다르기 때문에 모든 css 스타일을 분석해 태그에 스타일 규칙이 적용되게 결정

5. Javascript 실행

브라우저는 HTML을 읽는 도중 `<script>` 태그를 만나면 Javascript 엔진을 사용해 Javascript 실행

이 과정에서 DOM이나 CSSOM을 변경할 수 있음

5. 페인팅(그리기)

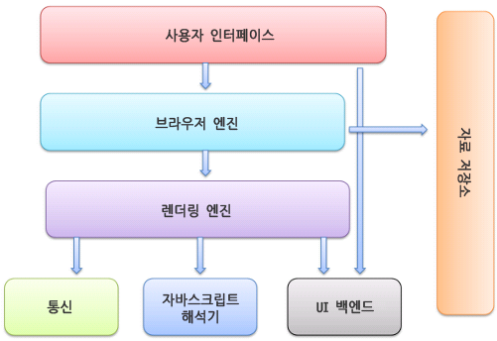
렌더 트리의 각 노드를 실제 픽셀로 변환 → 실제 그리는 작업을 실행

> 💡 참고로 자바스크립트가 DOM, CSSOM을 변경하는 경우를 리랜더링
> 리플로우: 레이아웃을 다시 계산하는 것
> 리페인트: 재결합된 렌더 트리를 기반으로 다시 페인트하는 것

요약

1. 사용자가 URL을 입력 → HTTP 요청 전송
2. 서버에서 HTML, CSS, JS 등의 리소스를 응답
3. 브라우저는 HTML과 CSS 파싱하여 DOM Tree, CSSOM Tree 생성
4. 렌더링 엔진이 레이아웃과 페인팅 작업

브라우저의 기본 구조



1. 사용자 인터페이스

주소 표시줄, 이전/다음 버튼, 북마크 메뉴 등 요청한 페이지를 보여주는 창을 제외한 나머지 모든 부분

2. 브라우저 엔진

사용자 인터페이스와 렌더링 엔진 사이 동작을 제어

3. 렌더링 엔진

요청한 콘텐츠를 표시

- 예) HTML을 요청하면 HTML과 CSS를 파싱해 화면에 표시

4. 통신

HTTP 요청과 같은 네트워크 호출에 사용

(이것은 플랫폼 독립적인 인터페이스이고 각 플랫폼 하부에서 실행됨)

5. UI 백엔드

콤보 박스와 창 같은 기본적인 장치를 그림

플랫폼에서 명시하지 않은 일반적인 인터페이스로서 OS 사용자가 인터페이스 체계를 사용

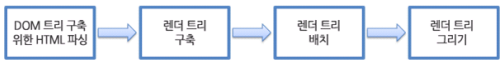
6. 자바스크립트 해석기

자바스크립트 코드를 해석하고 실행

7. 자료 저장소

- 자료를 저장하는 계층
- 쿠키를 저장하는 것과 같이 모든 종류의 자원을 하드 디스크에 저장할 필요가 있음
- HTML5 명세에는 브라우저가 지원하는 '웹 데이터 베이스'가 정의되어 있음

렌더링 엔진의 동작 과정



1. HTML 문서를 파싱
2. 콘텐츠 트리 내부에서 태그를 DOM 노드로 변환
3. 외부 CSS 파일, 함께 포함된 스타일 요소 파싱
4. 스타일 정보와 HTML 표시 규칙으로 렌더 트리라고 부르는 또 다른 트리 생성

파싱

- 브라우저가 코드를 이해하고 사용할 수 있는 구조로 변환하는 것을 의미

- 파싱 결과는 보통 노드 트리이며 파싱 트리 또는 문법 트리라고 부름
- HTML 파서, CSS 파싱을 통해 각각 파싱 트리로 변환
- 파싱이 끝난 이후의 동작
    - 브라우저는 문서와 상호 작용을 할 수 있게 되고, 문서 파싱 이후에 실행돼야 하는 “지연(defer)” 모드 스크립트를 파싱하기 시작

스크립트와 스타일 시트의 진행 순서

스크립트

- 웹은 파싱과 실행이 동시에 수행되는 동기화(synchronous) 모델
- 스크립트가 실행되는 동안 문서(HTML)의 파싱은 중단 → script 태그를 만나면 코드 실행을 위해 DOM 생성 프로세스 중단
- 스크립트가 외부에 있는 경우 우선 네트워크로부터 자원을 가져와야 하는데 이 또한 실시간으로 처리되고 자원을 받을 때까지 파싱은 중단됨
- 스크립트를 “지연(defer)”으로 표시할 수 있는데 지원으로 표시하게 되면 문서 파싱은 중단되지 않고 문서 파싱이 완료된 이후에 스크립트가 실행됨
- 즉, HTML 구문 분석이 완료되면 스크립트 파일이 실행
- HTML5는 스크립트를 비동기(asynchronous)로 처리하는 속성을 추가했기 때문에 별도의 맥락에 의해 파싱되고 실행됨

스타일 시트

이론적으로 스타일 시트는 DOM 트리를 변경하지 않기 때문에 문서 파싱을 기다리거나 중단할 이유가 없음

그러나 스크립트가 파싱되는 동안 스타일 정보를 요청하는 경우라면 문제가 생김

참고

https://d2.naver.com/helloworld/59361

https://velog.io/@zaman17/기술면접대비-브라우저-렌더링-순서와-원리