브라우저 동작 방법

"브라우저가 어떻게 동작하는지 아세요?"

웹 서핑하다보면 우리는 여러 url을 통해 사이트를 돌아다닌다. 이 url이 입력되었을 때 어떤 과정을 거쳐서 출력되는걸까?

web의 기본적인 개념이지만 설명하기 무지 어렵다.. 렌더링..? 파싱..?

브라우저 주소 창에 http://naver.com을 입력했을 때 어떤 과정을 거쳐서 네이버 페이지가 화면에 보이는 지 알아보자

> 오픈 소스 브라우저(크롬, 파이어폭스, 사파리 등)로 접속했을 때로 정리

브라우저 주요 기능

사용자가 선택한 자원을 서버에 요청, 브라우저에 표시

자원은 html 문서, pdf, image 등 다양한 형태

자원의 주소는 URI에 의해 정해짐

브라우저는 html과 css 명세에 따라 html 파일을 해석해서 표시함

이 '명세'는 웹 표준화 기구인 `W3C(World wide web Consortium)`에서 정해짐

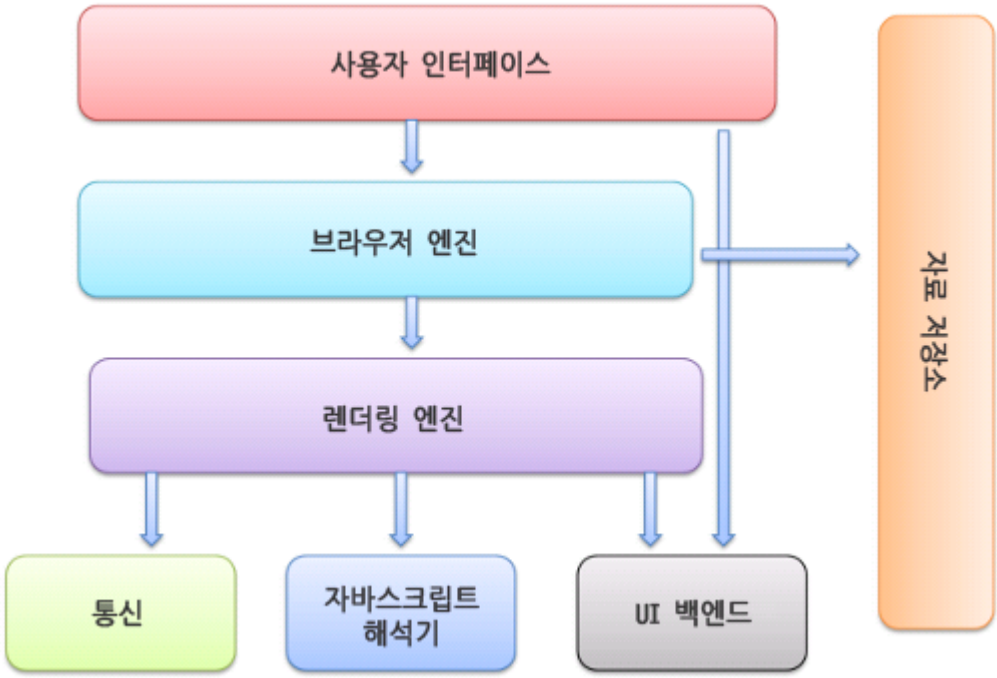
> 예전 브라우저들은 일부만 명세에 따라 구현하고 독자적 방법으로 확장했음
>
> (결국 **심각한 호환성 문제** 발생... 그래서 요즘은 대부분 모두 표준 명세를 따름)

브라우저가 가진 인터페이스는 보통 비슷비슷한 요소들이 존재

> 시간이 지나면서, 사용자에게 필요한 서비스들로 서로 모방하며 갖춰지게 된 것

- URI 입력하는 주소 표시 줄
- 이전 버튼, 다음 버튼
- 북마크(즐겨찾기)
- 새로 고침 버튼
- 홈 버튼

브라우저 기본 구조



사용자 인터페이스

주소 표시줄, 이전/다음 버튼, 북마크 등 사용자가 활용하는 서비스들 (요청한 페이지를 보여주는 창을 제외한 나머지 부분)

브라우저 엔진

사용자 인터페이스와 렌더링 엔진 사이의 동작 제어

렌더링 엔진

요청한 콘텐츠 표시 (html 요청이 들어오면? → html, css 파싱해서 화면에 표시)

통신

http 요청과 같은 네트워크 호출에 사용 (플랫폼의 독립적인 인터페이스로 구성되어있음)

UI 백엔드

플랫폼에서 명시하지 않은 일반적 인터페이스. 콤보 박스 창같은 기본적 장치를 그림

자바스크립트 해석기

자바스크립트 코드를 해석하고 실행

자료 저장소

쿠키 등 모든 종류의 자원을 하드 디스크에 저장하는 계층

렌더링이란?

웹 분야를 공부하다보면 **렌더링**이라는 말을 많이 본다. 동작 과정에 대해 좀 더 자세히 알아보자

렌더링 엔진은 요청 받은 내용을 브라우저 화면에 표시해준다.

기본적으로 html, xml 문서와 이미지를 표시할 수 있음

추가로 플러그인이나 브라우저 확장 기능으로 pdf 등 다른 유형도 표시가 가능함

(추가로 확장이 필요한 유형은 바로 뜨지 않고 팝업으로 확장 여부를 묻는 것을 볼 수 있을 것임)

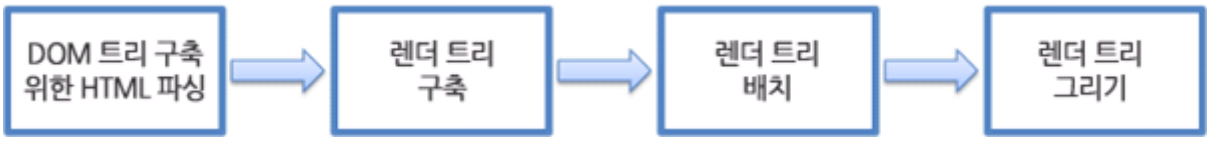
렌더링 엔진 종류

크롬, 사파리 : 웹킷(Webkit) 엔진 사용

파이어폭스 : 게코(Gecko) 엔진 사용

웹킷(Webkit) : 최초 리눅스 플랫폼에 동작하기 위한 오픈소스 엔진 (애플이 맥과 윈도우에서 사파리 브라우저를 지원하기 위해 수정을 더했음)

렌더링 동작 과정



먼저 html 문서를 파싱한다.

그리고 콘텐츠 트리 내부에서 태그를 모두 DOM 노드로 변환한다.

그 다음 외부 css 파일과 함께 포함된 스타일 요소를 파싱한다.

이 스타일 정보와 html 표시 규칙은 렌더 트리라고 부르는 또 다른 트리를 생성한다.

이렇게 생성된 렌더 트리는 정해진 순서대로 화면에 표시되는데, 생성 과정이 끝났을 때 배치가 진행되면서 노드가 화면의 정확한 위치에 표시되는 것을 의미한다.

이후에 UI 백엔드에서 렌더 트리의 각 노드를 가로지으며 형상을 만드는 그리기 과정이 진행된

```
다.

이러한 과정이 점진적으로 진행되며, 렌더링 엔진은 좀더 빠르게 사용자에게 제공하기 위해 모
든 html을 파싱할 때까지 기다리지 않고 배치와 그리기 과정을 시작한다. (마치 비동기처
럼..?)

전송을 받고 기다리는 동시에 받은 내용을 먼저 화면에 보여준다
(우리가 웹페이지에 접속할 때 한꺼번에 뜨지 않고 점점 화면에 나오는 것이 이 때문!!!)
```

DOM 이란?

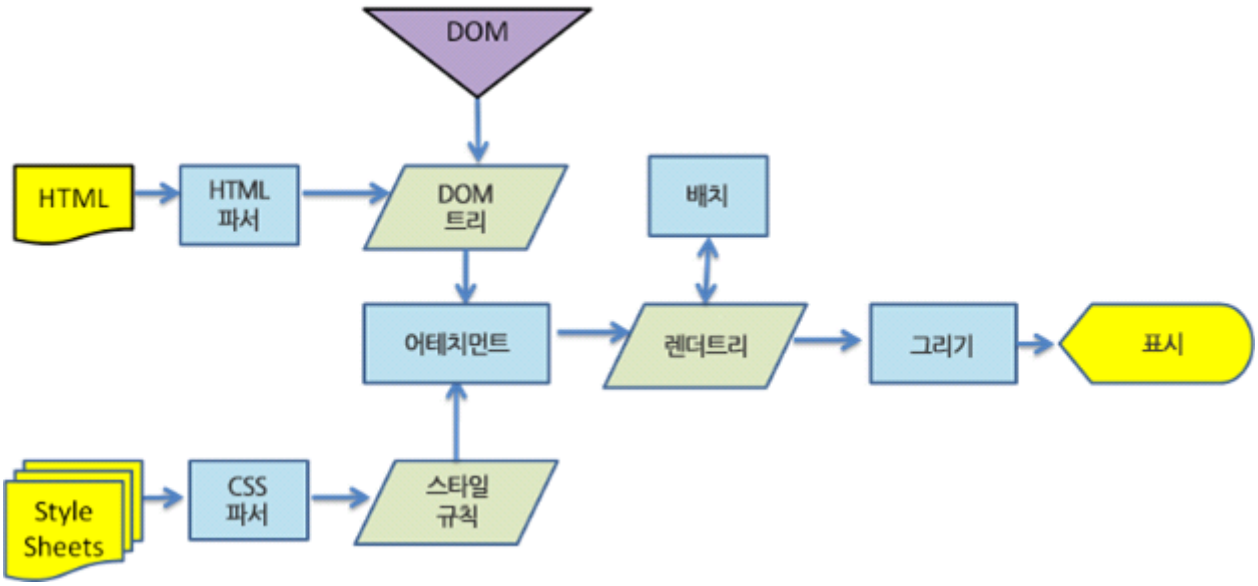
Document Object Model(문서 객체 모델)

웹페이지 소스를 까보면 `<html>, <body>`와 같은 태그들이 존재한다. 이를 Javascript가 활용할 수 있는 객체로 만들면 문서 객체가 된다.

모델은 말 그대로, 모듈화로 만들었다거나 객체를 인식한다라고 해석하면 된다.

즉, **DOM은 웹 브라우저가 html 페이지를 인식하는 방식**을 말한다. (트리구조)

웹킷 동작 구조



> **어태치먼트** : 웹킷이 렌더 트리를 생성하기 위해 DOM 노드와 스타일 정보를 연결하는 과정

이제 조금 트리 구조의 진행 방식이 이해되기 시작한다..ㅎㅎ

파싱과 DOM 트리 구축

파싱이라는 말도 많이 들어봤을 것이다.

파싱은 렌더링 엔진에서 매우 중요한 과정이다.

파싱(parsing)

문서 파싱은, 브라우저가 코드를 이해하고 사용할 수 있는 구조로 변환하는 것

문서를 가지고, **어휘 분석과 구문 분석** 과정을 거쳐 파싱 트리를 구축한다.

조금 복잡한데, 어휘 분석기를 통해 언어의 구문 규칙에 따라 문서 구조를 분석한다. 이 과정에서 구문 규칙과 일치하는 지 비교하고, 일치하는 노드만 파싱 트리에 추가시킨다. (끝까지 규칙이 맞지 않는 부분은 문서가 유효하지 않고 구문 오류가 포함되어 있다는 것)

파서 트리가 나왔다고 해서 끝이 아니다.

컴파일의 과정일 뿐, 다시 기계코드 문서로 변환시키는 과정까지 완료되면 최종 결과물이 나오게 된다.

보통 이런 파서를 생성하는 것은 문법에 대한 규칙 부여 등 복잡하고 최적화하기 힘드므로, 자동으로 생성해주는 파서 생성기를 많이 활용한다.

> 웹킷은 플렉스(flex)나 바이슨(bison)을 이용하여 유용하게 파싱이 가능

우리가 head 태그를 실수로 빠뜨려도, 파서가 돌면서 오류를 수정해줌 (head 엘리먼트 객체를 암묵적으로 만들어준다)

결국 이 파싱 과정을 거치면서 서버로부터 받은 문서를 브라우저가 이해하고 쉽게 사용할 수 있는 DOM 트리구조로 변환시켜주는 것이다!

요약

- 주소창에 url을 입력하고 Enter를 누르면, **서버에 요청이 전송**됨
- 해당 페이지에 존재하는 여러 자원들(text, image 등등)이 보내짐
- 이제 브라우저는 해당 자원이 담긴 html과 스타일이 담긴 css를 W3C 명세에 따라 해석할 것임
- 이 역할을 하는 것이 **'렌더링 엔진'**
- 렌더링 엔진은 우선 html 파싱 과정을 시작함. html 파서가 문서에 존재하는 어휘와 구문을 분석하면서 DOM 트리를 구축
- 다음엔 css 파싱 과정 시작. css 파서가 모든 css 정보를 스타일 구조체로 생성

- 이 2가지를 연결시켜 **렌더 트리**를 만듬. 렌더 트리를 통해 문서가 **시각적 요소를 포함한 형태로 구성**된 상태
- 화면에 배치를 시작하고, UI 백엔드가 노드를 돌며 형상을 그림
- 이때 빠른 브라우저 화면 표시를 위해 '배치와 그리는 과정'은 페이지 정보를 모두 받고 한꺼번에 진행되지 않음. 자원을 전송받으면, **기다리는 동시에 일부분 먼저 진행하고 화면에 표시**함

[참고 자료]

네이버 D2 : 링크