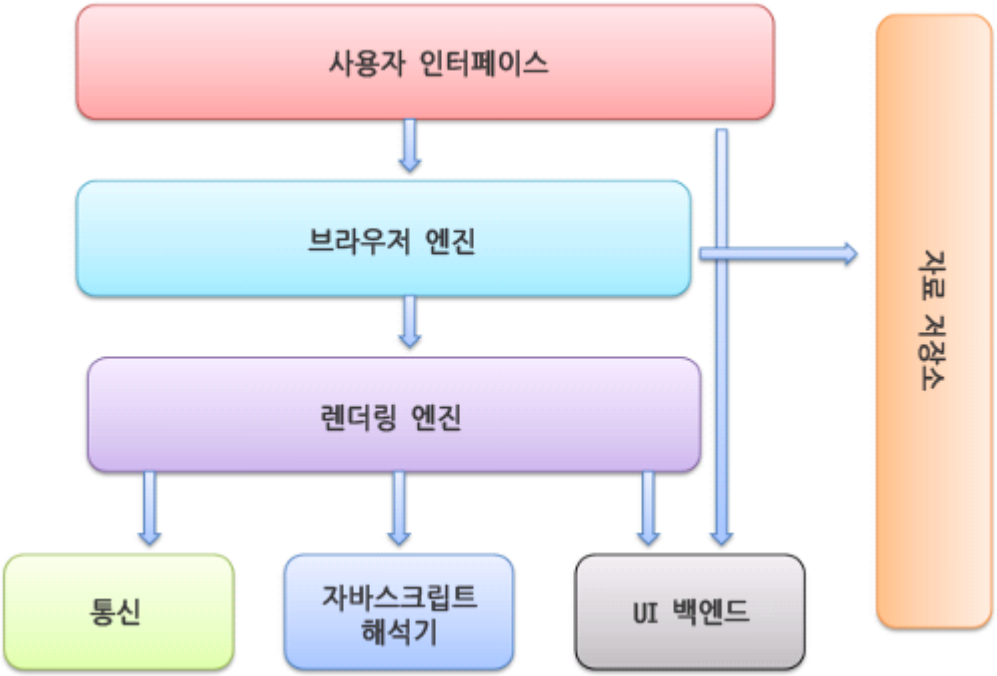
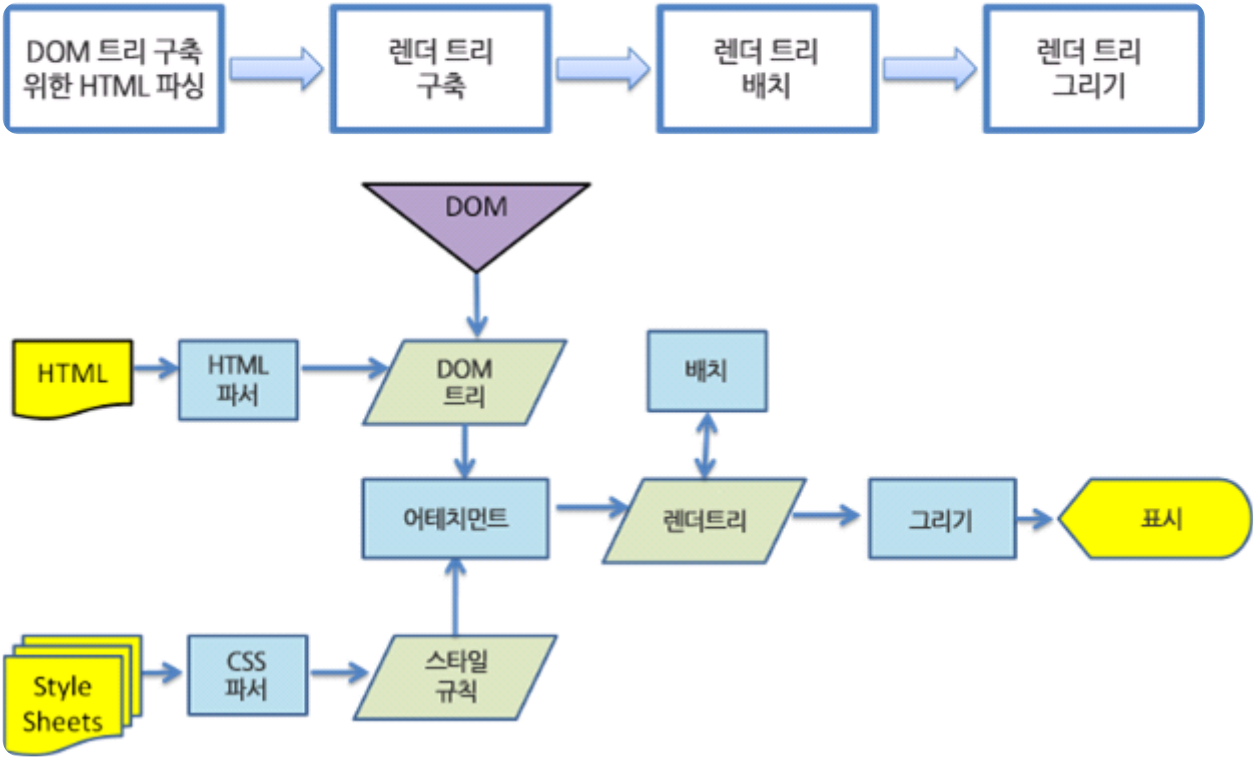
브라우저 동작 원리와 <script> 태그의 위치

브라우저의 기본 구조

1. 브라우저 엔진 : 사용자 인터페이스와 렌더링 엔진 사이의 동작을 제어한다.
2. 렌더링 엔진 : 요청한 콘텐츠를 표시한다.
3. 사용자 인터페이스 : 주소 표시줄, 이전/다음 버튼, 북마크 메뉴 등. 요청한 페이지를 보여주는 창(View port)를 제외한 나머지 모든 부분이다.
4. 통신 : HTTP 요청과 같은 네트워크 호출에 사용됨. 이것은 플랫폼 독립적인 인터페이스이고 각 플랫폼 하부에서 실행됨.
5. UI 백엔드 - 콤보 박스와 창 같은 기본적인 장치를 그림. 플랫폼에서 명시하지 않은 일반적인 인터페이스로서, OS 사용자 인터페이스 체계를 사용.
6. 자바스크립트 해석기 - 자바스크립트 코드를 해석하고 실행.
7. 자료 저장소 - 이 부분은 자료를 저장하는 계층이다. 쿠키를 저장하는 것과 같이 모든 종류의 자원을 하드 디스크에 저장할 필요가 있다.



렌더링 엔진의 동작과정



HTML 파싱과 DOM Tree 구축

일반적인 의미의 파싱

파싱 혹은 구문분석이란, 문장을 그것을 이루고 있는 구성 성분으로 분해하고 그들 사이의 위계 관계를 분석하여 문장의 구조를 결정하는 것을 말한다.

컴퓨터 과학에서의 파싱은 일련의 문자열을 의미있는 토큰으로 분해하고, 이들로 이루어진 Parse tree 를 만드는 과정을 의미한다.

파싱은 문서에 작성된 언어 또는 형식의 규칙에 따른다. 즉 파싱할 수 있는 모든 형식은 정해진 용어와 구문 규칙에 따라야 한다. 이것을 문맥 자유 문법이라고 한다.

파싱의 과정은 어휘 분석과 구문 분석이라는 두 과정으로 분류할 수 있다.

- 어휘 분석 : 자료를 토큰으로 분해하는 과정이다. 여기서 말하는 토큰이란, 유효하게 구성된 단위의 집합체를 뜻하며 용어집이라고 할 수 있다. 즉 사전의 모든 단어들과 의미가 같다.
- 구문 분석 : 언어의 구문 규칙을 적용하는 과정이다.

정의된 언어가 존재하고, 해당 언어들이 구문에 의한 규칙을 반영하여 하나의 문서로 작성되었다면 해당 문서는 파싱 가능하다. 즉 문맥 자유 문법인 모든 문서는 파싱이 가능하다.

- 어휘의 예시 : 수학 언어는 정수, 더하기 기호, 빼기 기호를 포함한다.

```
INTEGER : 0|[1-9][0-9]*
PLUS : +
MINUS : -
```
JAVASCRIPT

- 구문의 예시 :
  1. 언어 구문의 기본적인 요소는 표현식, 항, 연산자이다.
  2. 언어에 포함되는 표현식의 수는 제한이 없다.
  3. 표현식은 "항" 뒤에 "연산자" 그 뒤에 또 다른 항이 따르는 형태로 정의한다.
  4. 연산자는 더하기 토큰 또는 빼기 토큰이다.
  5. 정수 토큰 또는 하나의 표현식은 항이다.

```
expression := term operation term
operation := PLUS | MINUS
term := INTEGER | expression
```
JAVASCRIPT

HTML 파싱

브라우저에는 HTML, CSS, JS 각각에 대한 파서가 렌더링 엔진에 존재하고, 각각의 파서들이 어휘 분석과 구문 분석을 통해 파싱 작업을 진행하게 된다.

즉 렌더링 엔진은 HTML 과 같은 Document를 파싱하여 그 결과물인 DOM Tree 라는 파싱트리를 만들어 낸다.

일반적으로 대부분의 파서들은 어휘에 대한 규정은 정규표현식을 이용하고, 구문 규정은 BNF원칙을 활용한다.

그리고 특정인이 파서를 직접적으로 제작하기는 매우 힘들기 때문에 대부분 파서 생성기를 통해 원하는 어휘 규정과 구문 규정을 삽입하여 해당 규정들이 적용되는 파서를 만든다.

그러나 HTML 문서는 해당 방법을 통한 파서의 제작이 불가능하다. 이유는 다음과 같다.

HTML은 조금 더 유연하고 너그러워야 하는 특성을 가진다. 우리가 HTML 코드를 작성할 때, 일부 잘못 작성된 문법 구조가 있더라도, 런타임 이전에 에러를 잡아주지 않는다. 이는 내부적으로 잘못 작성된 구문에 대한 여러가지 처리과정을 거치기 때문이다.

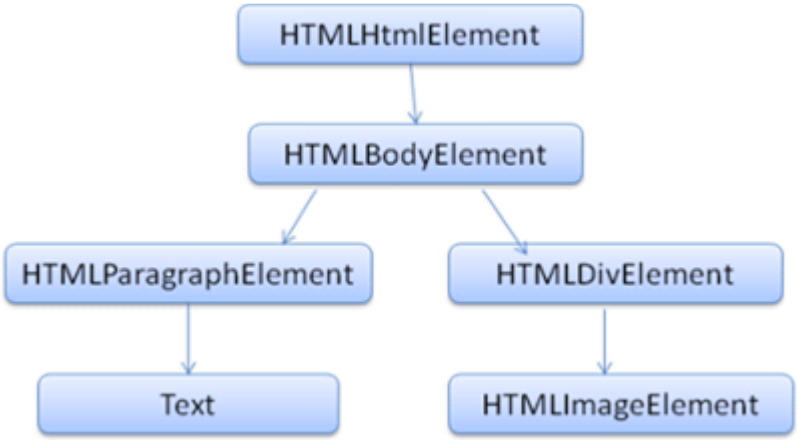
즉 이런 특성은 HTML이 전통적인 구문 분석 방법으로는 파싱하기 어렵게 만들고 이는 문맥 자유 문법에 어긋나게 만든다.

HTML 파싱 알고리즘

HTML 파싱 알고리즘을 이해하려면 DOM에 대한 이해가 선행되어야 한다.
DOM(Document Object Model)이란 HTML 문서의 객체 표현이다. 즉 HTML을 자바스크립트와 연결시키기 위한 지점이다. HTML 파싱의 결과인 파싱트리가 DOM(혹 DOM Tree)이다.

```HTML
<html>
  <body>
    <p>Hello World</p>
    <div><img src="example.png" /></div>
  </body>
</html>
```

이것은 아래와 같은 DOM 트리로 변환할 수 있다.

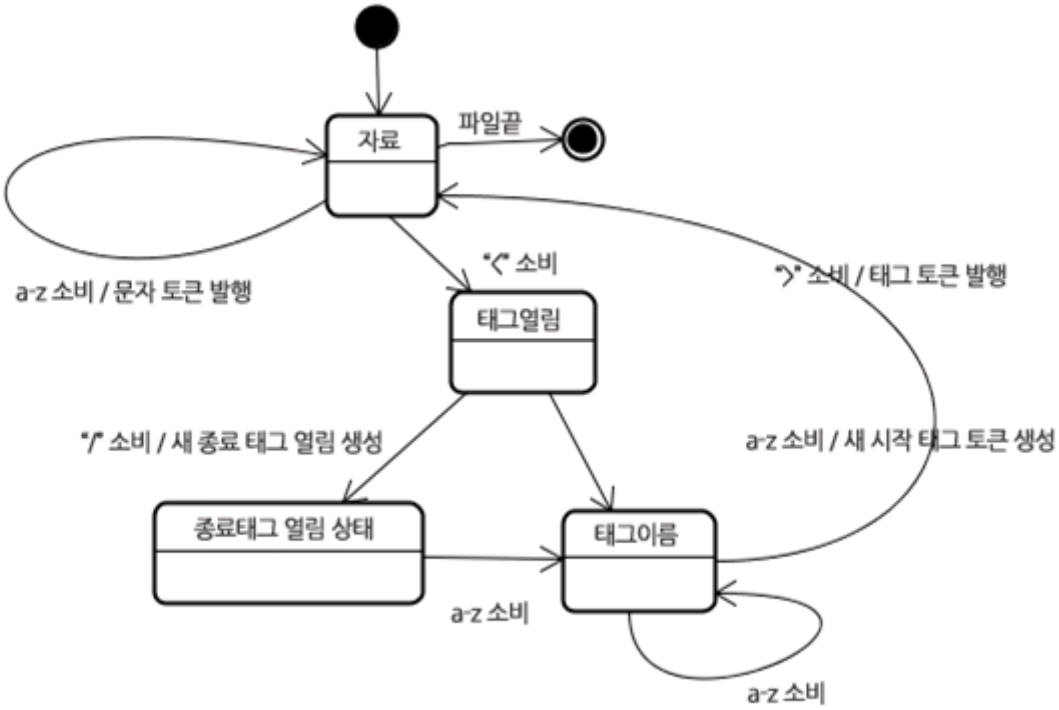


여기서 HTML이 일반적인 파싱 방법으로 파싱이 어려운 이유가 추가된다. 바로 변경에 의한 재파싱이 이루어 질 수 있는 가능성 때문이다.
일반적인 문서는 파싱하는 동안 변하지 않지만 HTML에서 document.write을 포함하고 있는 스크립트 태그 혹은 Javascript 내부의 DOM 조작 명령어는 토큰을 추가할 수 있기 때문에 실제로는 입력 과정에서 파싱이 수정된다.

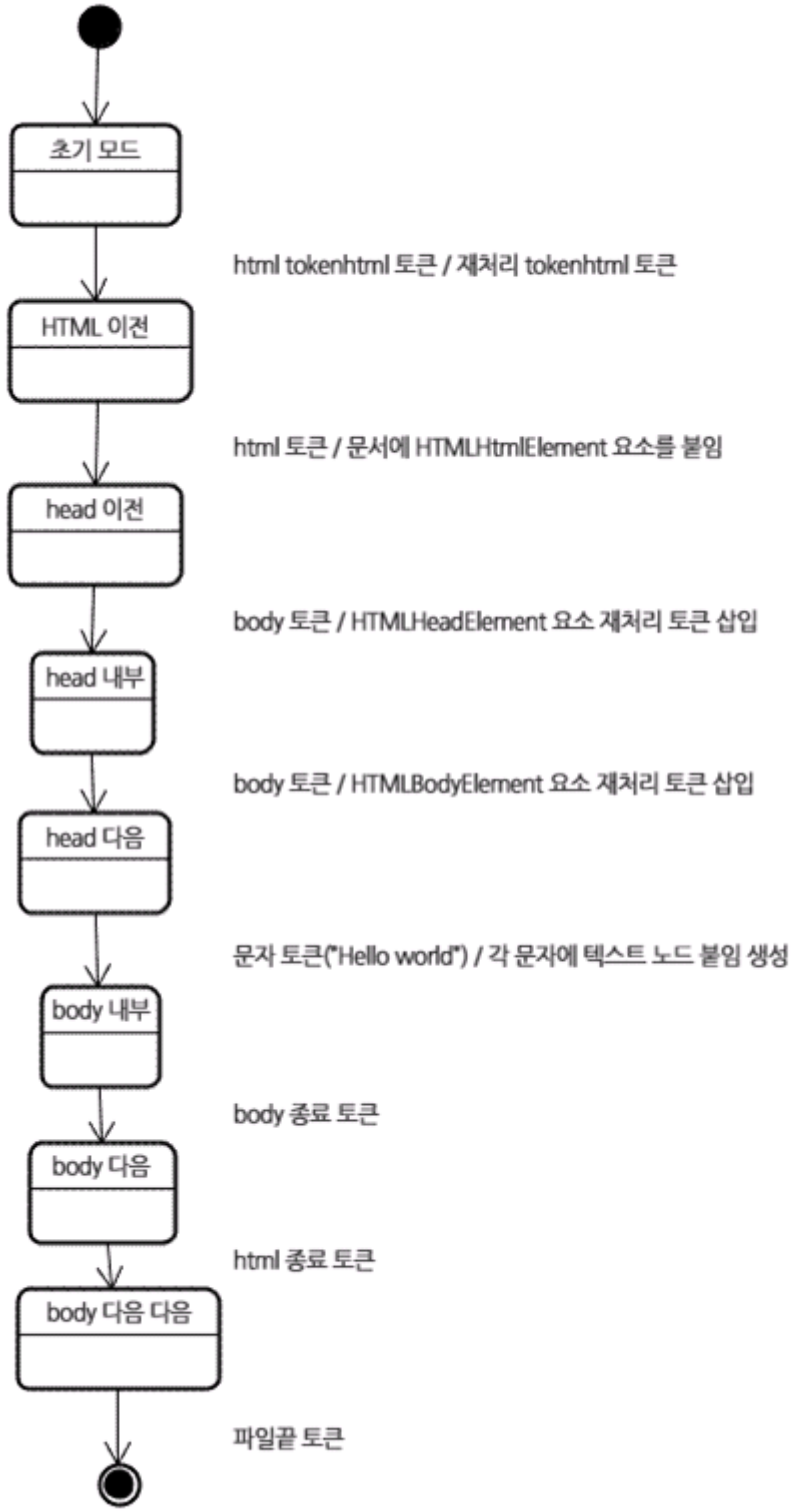
그래서 HTML 파싱 알고리즘은 토큰화와 트리 구축의 두 단계로 이루어진다.

- 토큰화 : 토큰을 인지해서 트리 생성자로 넘기고 다음 토큰을 확인하기 위해 다음 문자를 확인한다. 입력의 마지막 까지 해당 과정을 반복한다.

토큰화는 State를 기반으로 과정을 수행한다. 때문에 State Machine 이라고 볼 수 있다. 각 상태는 하나 이상의 연속된 문자를 입력받아 이 문자에 따라 다음 상태를 갱신한다. 결과는 현재의 토큰화 상태와 트리 구축 상태의 영향을 받는데. 이것은 같은 문자를 읽어 들여도 현재 상태에 따라 다음 상태의 결과가 다르게 나온다는 것을 의미한다.



트리 구축 단계에서 DOM이 생성된다. 트리 구축이 진행되는 동안 문서 최상단에서는 DOM 트리가 수정되고 요소가 추가된다. 토큰화에 의해 발행된 각 노드는 트리 생성자에 의해 처리된다. 즉 HTML의 각 엘리먼트가 토큰화를 통해 토큰이 만들어지면 해당 토큰에 대한 노드를 DOM에 추가한다. 각 토큰과 매핑되는 DOM을 이루는 요소의 명세는 정의되어 있다.



토큰화와 트리 구축의 단계를 통해 HTML의 파싱이 끝나고 나면 DOM이 생성되며, 브라우저는 해당 DOM을 통해 문서와 상호작용 할 수 있게 된다.