How browsers work

Browser

- web browser is a program on your computer that allows you to visit websites

  [What is a broswer? - Google]

▼ browse

둘러보다 , 검색하다, 인터넷을 돌아다니다

▼ website

웹사이트는 인터넷 프로토콜 기반의 네트워크에서 도메인 이름이나 IP 주소, 루트 경로만으로 이루어진 일반 URL을 통하여 보이는 웹 페이지들의 의미 있는 묶음이다.

> 쉽게말해, 인터넷에 있는 웹사이트들을 방문하기 위해 사용되는 프로그램들

Ther Browsers’s main functionality

- 브라우저의 주요 기능은 사용자가 선택한 웹 리소스를 서버에 요청하여 브라우저 창에 표시하는 것
- 이때 리소스는 일반적으로 HTML 그러나 PDF, 이미지 같은 다른 유형일 수 도 있음.
- 리소스의 위치는 사용자가 URI(Uniform Resource Identifier)를 사용하여 지정

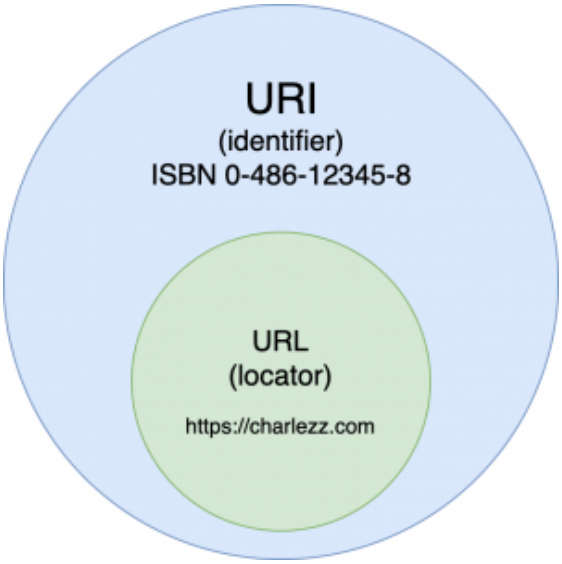
▼ URI vs URL

URI

웹 기술에서 사용하는 논리적 또는 물리적 리소스를 식별하는 고유한 문자열 시퀀스다.

URL

URL은 흔히 웹 **주소**라고도 하며, 컴퓨터 네트워크 상에서 리소스가 **어디 있는지 알려주기** 위한 규약이다. URI의 서브셋이다.



> URI는 식별하고, URL은 위치를 가르킨다.

Ex

- elancer.co.kr은 URI입니다. 리소스의 이름만 나타내기 때문입니다.
- https://elancer.co.kr는 URL이자, URI이다. 어떻게 도달 할 수 있는지 까지 함께 나타냈기 때문이다.

- 보편적인 UI
  1. Address bar for inserting a URI
  2. Back and forward buttons
  3. Bookmarking options
  4. Refresh and stop buttons for refreshing or stopping the loading of current documents
  5. Home button that takes you to your home page
- 공식적인 규격은 없지만 서로 모방하면서 보편적인 브라우저의 UI가 만들어짐

The browser's high level structure

1. UI: 사용자 인터페이스

2. browser engine: UI와 렌더링 엔진 간의 작업을 관리
3. rendering engine: 요청된 콘텐츠 표시를 담당
4. Networking: HTTP요청과 같은 네트워크 호출 수행
5. UI Backend: 기본 위젯을 그리는데 사용
6. JavaScript interpreter: 자바 코드를 구문 분석하고 실행하는데 사용
7. Data Storage: 데이터 저장소

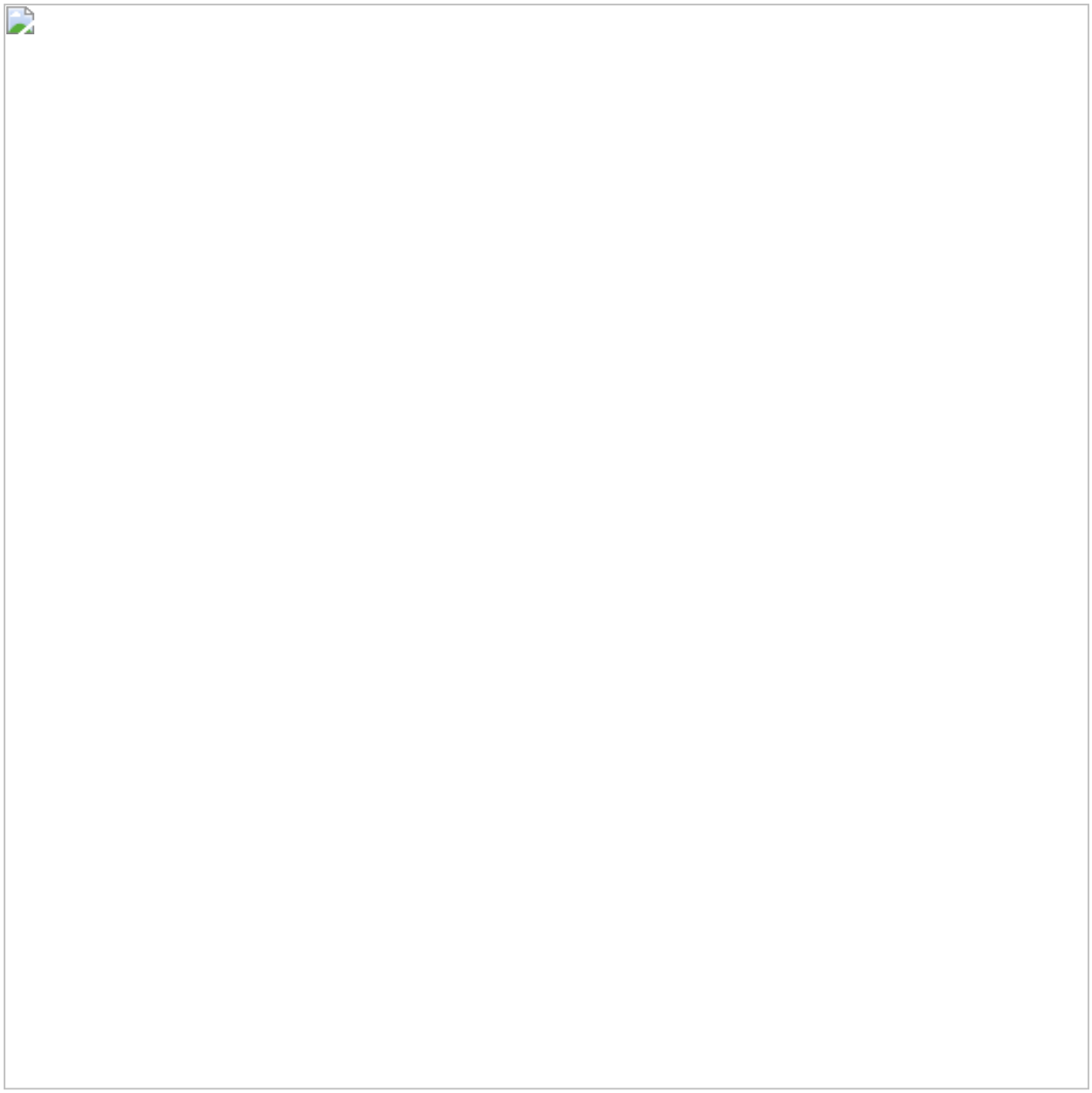
The rendering engine

- 렌더링 엔진의 역할을 요청된 콘텐츠를 브라우저 화면에 표시하는 것
- HTML, XML 문서와 이미지를 표시할 수 있음
- 브라우저마다 다른 랜더링 엔진을 이용함
  - IE → Trident
  - Firefox → Gecko
  - Safari → Webkit
  - Chrome, Opera → Blink

The main flow (rendering engine)

1. 렌더링 엔진이 네트워크 계층(3계층?)에서 요청된 문서의 콘텐츠를 가져옴
2. HTML 문서를 Parsing하고 elements를 ‘contents tree’라는 트리(DOM Tree?)의 DOM nodes로 변환 시킴, CSS 파일들과 style elements에서 style data를 parsing해서 스타일 정보를 얻음(CSSOMTREE?). DOM nodes와 스타일 정보를 토대로 render tree 구성(? 언어는 각 브라우저마다 다를 수 있음)
3. render tree 구성후 화면에 표시되어야 할 정확할 좌표를 지정하는 레이아웃 프로세스를 거침 (뷰포트에 맞춰서
4. 랜더 트리를 traverse하고 UI Backend Layer를 사용하여 각 노드를 페인팅

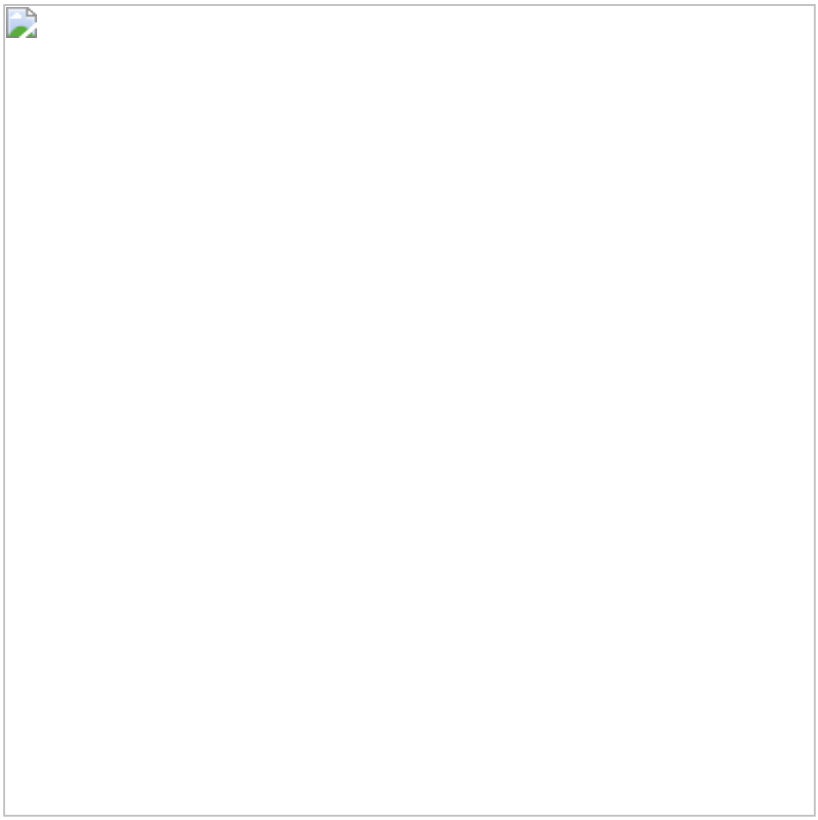
Webkit main flow



Parsing - general

문서를 파싱한다는 것은 코드가 사용가능한 구조로 변한다는 것을 의미

ex) 2 + 3 - 1



Grammar

Parsing은 문서가 작성된 언어 또는 형식을 기반으로 한다. 구문 분석을 할 수 있는 모든 형식은 어휘와 구문 규칙으로 이뤄진 deterministic grammar가 있어야 한다. 이때 이를 문맥 자유 문법이라 한다.

Grammar

- **Specify a language** in terms of **rules** that allow the generation of "legal" strings

- Example) Spring grammar
  - Spring-Sent ➔ Spring-Subj Spring-Verb
  - Spring-Subj ➔ 새가 | 꽃이 | 싹이
  - Spring-Verb ➔ 노래한다 | 핀다 | 난다

  - Spring-Sent ➔ Spring-Subj Spring-Verb ➔ 새가 Spring-Verb ➔ 새가 노래한다

  - L(Spring grammar) = {새가 노래한다, 새가 핀다, 새가 난다, 꽃이 노래한다, 꽃이 핀다, 꽃이 난다, 싹이 노래한다, 싹이 핀다, 싹이 난다}

The Chomsky Hierarchy (촘스키 계층 구조)

- Four types of formal grammars (Chomsky hierarchy)
- American linguist Noam Chomksy

Type	Grammar	Production Rules
0	Unrestricted	$\alpha \rightarrow \beta$
1	Context-Sensitive	$\alpha \rightarrow \beta$ $\lvert\alpha\rvert \leqq \lvert\beta\rvert, \alpha \in V^{+}, \beta \in V^{*}$
2	Context-Free	$\alpha \rightarrow \beta$ $\alpha \in V_N, \beta \in V^{*}$
3	Regular	$\alpha \rightarrow t\beta \mid t$ (right linear) $\alpha \rightarrow \beta t \mid t$ (left linear) $\alpha, \beta \in V_N, t \in V_T^{*}$

Parser

Parsing은 두 가지 하위 프로세스로 나눌 수 있다.

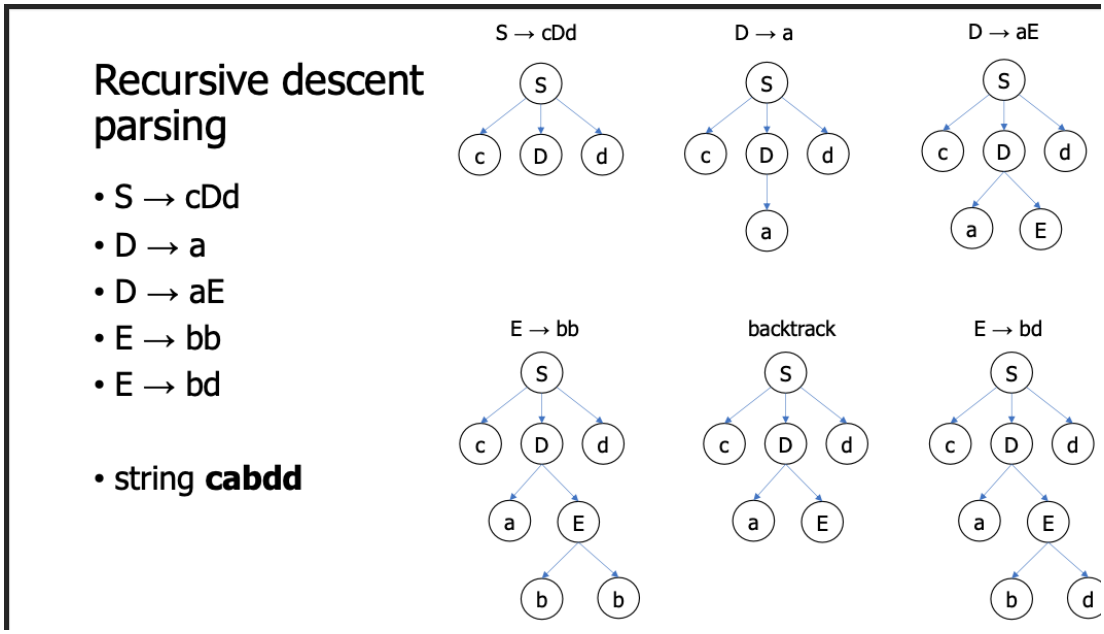
- Lexical analysis: 입력을 토큰으로 분해하는 과정 이때 토큰은 the collection of valid building blocks
  - **Tokens are sequences of characters with a collective meaning - Compilier**
- Syntax analysis: 언어 구문 규칙을 적용하는 작업

Translation

- Parser에서 Parsing 해서 나온 parsing tree를 Machine code로 변환해주는 작업

Types of parsers

- Top-Down
  - 컴퓨터 과학에서 하향식 구문 분석은 먼저 구문 분석 트리의 가장 높은 수준을 살펴보고 형식 문법의 재작성 규칙을 사용하여 구문 분석 트리를 아래로 내려가는 구문 분석 전략입니다



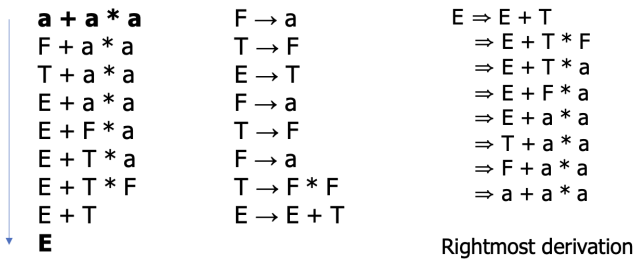
- Bottom-Up
  - 상향식 구문 분석은 텍스트의 가장 낮은 수준의 작은 세부 사항을 중간 수준의 구조보다 먼저 인식하고 가장 높은 수준의 전체 구조는 마지막에 남겨 둡니다.

Bottom up parsing (상향식 구문 분석)

• Reduces **a string to the start symbol** by inverting productions:

• E → E + T | T
• T → T * F | F
• F → (E) | a

• string **a + a * a**



어떻게 Parsing 하고 painting 하고 layout 하는지는 추후에 공부함

www.naver.com을 브라우저에 쳤을 때 생기는 과정, 그리고 DNS에 대하여 설명

1. 리다이렉트가 있다면 리다이렉트를 진행하고 없다면 그대로 해당 요청에 대한 과정이 진행

   ▼ 리다이렉트란?

   웹 브라우저가 서버에 URL을 요청했을 때 서버가 HTTP Respond를 통해 브라우저에게 다른 URL을 요청하라고 지시 하는 기법

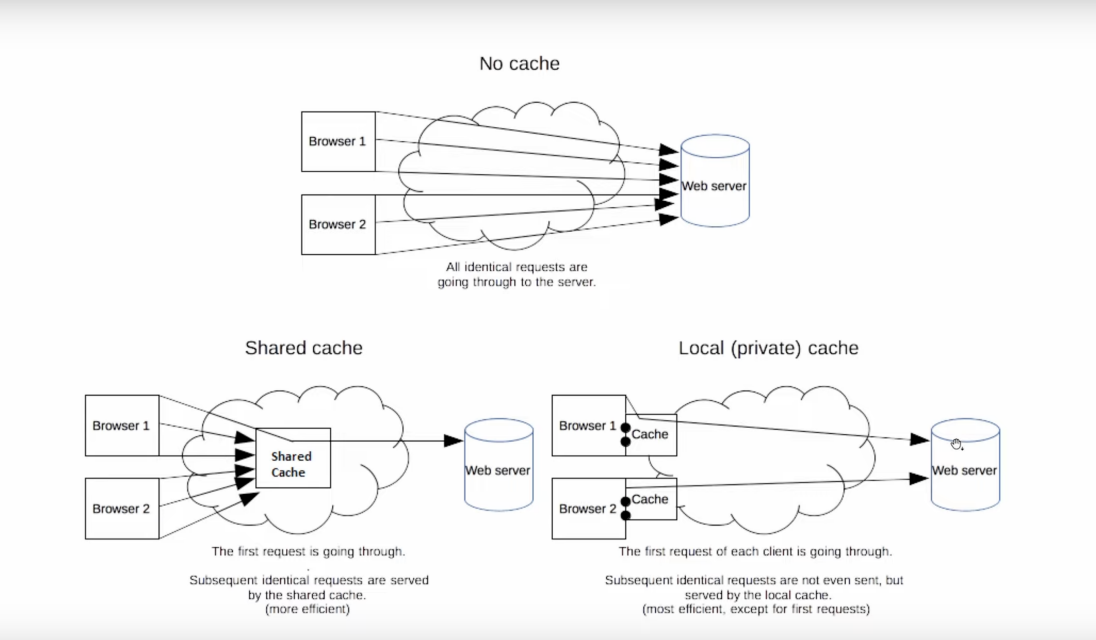
   ex:) 로그인을 하지않고 결제를 시도할려 할때

   결제 → 로그인하고 결제하도록 유도 → 로그인페이지(리다이렉트)

2. 캐싱

   a. 브라우저캐시: 쿠키, 로컬스토리지 등을 포함한 캐시 개인캐시라고 불림

   b. 공유캐시 : 클라이언트와 서버 사시에 있으며 사용자간에 공유할 수 있는 응답을 저장하는 캐시 ex:) 서버 앞단에 프록시 서버가 캐싱



3. DNS
    - DNS란?
        - 계층적인 도메인 구조와 분산된 데이터베이스를 이용한 시스템으로 FQDN을 인터넷 프로토콜인 IP로 바꿔주는 시스템 FQDN은 호스트와 도메인이 합쳐진 완전한 도메인 응답값을 클라이언트에게 전달하는 리졸버, 도메인을 IP로 변환하는 네임서버로 구성
    - DNS 캐싱
        - 미리 해당 도메인 이름을 요청했다면 로컬 pc에 자동적으로 저장 이때 브라우저 캐싱과 OS 캐싱이 있음
    - www.naver.com을 역순으로 매핑
4. IP라우팅 & ARP
    - 해당 IP를 기반으로 라우팅, ARP 과정을 거쳐 실제 서버를 찾음
        - ARP: IP 주소로부터 MAC 주소를 구하는 IP와 MAC 주소의 다리 역할을 하는 프로토콜
5. TCP 연결 구축
    a. 3way handshakes & SSL(사용자 및 시스템 간의 데이터를 암호화하는 통신 프로토콜입니다. 네트워크를 통해 연결된 두 당사자를 인증하므로 데이터를 안전하게 교환할 수 있습니다.) 연결등을 통해 연결설정
6. 콘텐츠 다운로드
    a. HTTP request를 서버에게 보내고 서버는 HTTP Respond를 전송

    b. TTFB(Time to First byte) 첫번째 패킷의 첫번째 바이트를 수신받을때가 TTFB 종료시점
7. 브라우즈 렌더링
8. 웹 브라우저가 서버로부터 응답을 받으면 응답 헤더를 검사하여 리소스를 렌더링

REF

- www.naver.com을 브라우저에 쳤을 때 생기는 과정, 그리고 DNS에 대하여 설명
    - 널널한 개발자 브라우저 동작과정
    - 큰돌 www.naver.com을 주소창에 치면 무슨일이 일어날까요?
    - 아마존 웹 브라우저에 URL을 입력하면 어떤 일이 일어날까요?
- URI vs URL
    - https://www.charlezz.com/?p=44767
    - https://inpa.tistory.com/entry/WEB-🌐-URL-URI-차이#uri_/_url_/_urn_구분하기
    - https://stackoverflow.com/questions/4913343/what-is-the-difference-between-uri-url-and-urn