



# Extrakce dat z webu

Aka web scraping

**doc. Ing. Radek Burget, Ph.D.**

[burgetr@fit.vutbr.cz](mailto:burgetr@fit.vutbr.cz)

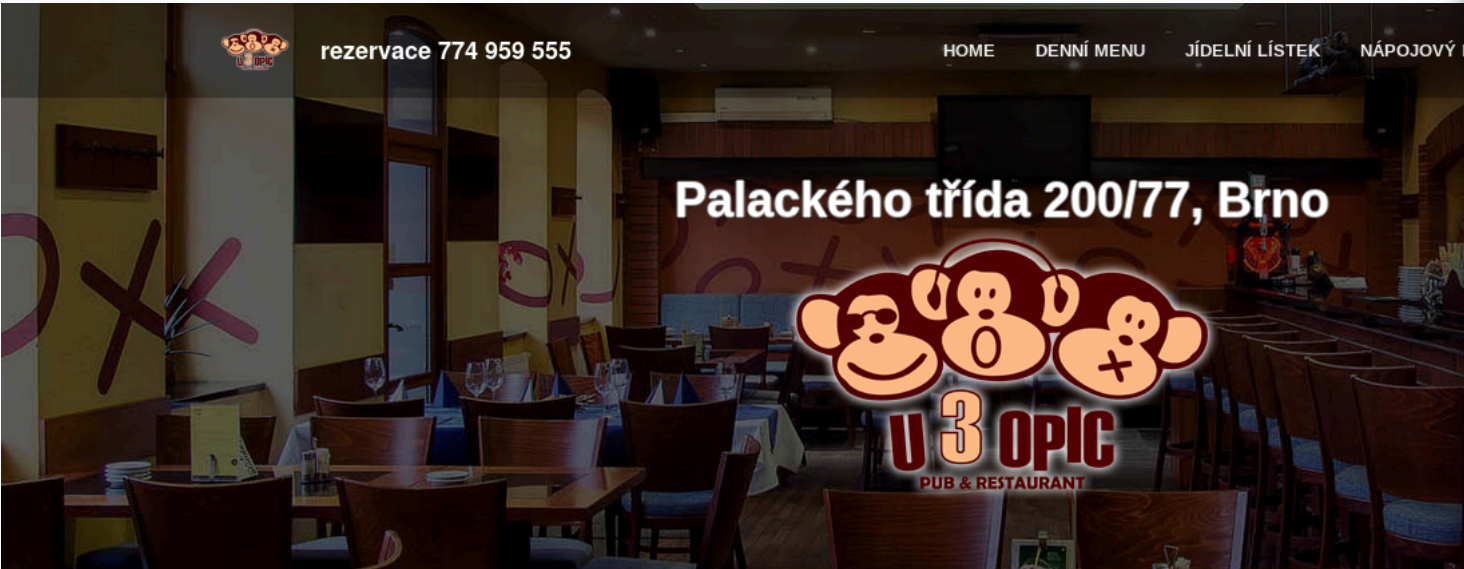
# Motivace

- **Na webu je spousta dat** ukrytých ve webových stránkách
- Potřebujeme dále zpracovat tato data v počítačových aplikacích
  - Propojení s vlastními daty
  - Agregace – spojení výsledků z různých zdrojů
  - Analýza – statistiky, získávání znalostí
  - ... a mnohé další
- Potřebujeme **strukturovaná data**
  - Reprezentovatelná tabulkami relační databáze
  - Nebo alespoň XML, JSON, apod. *s pevnou strukturou*

# Data na webu

- Webové stránky nejsou silně strukturované
- Nejčastěji v HTML
  - Prvotním cílem je *vizuální prezentace*
  - Kód je druhotný, *podřízený prvnímu cíli*
  - Není určen k dalšímu zpracování
  - Často proměnlivé schéma
- **Slabě strukturované dokumenty**

# Vzhled a kód



```
<section class="menu">
  <div class="container">
    <div class="row">
      <div class="col-md-12">
        <div class="page-header wow fadeInDown">
          <h1>Dnešní polední menu</h1>
        </div>
      </div>
    </div>
    <div class="food-menu wow fadeInUp">
      <div class="row menu-items">
        <div class="menu-item col-sm-12 col-xs-12 starter"><div class="clearfix menu-wrapp
        </div>
      </div>
      <div class="row">
        <div class="col-md-12">
          <div class="menu-btn">
            <a class="btn btn-default btn-lg" href="/denni-menu/" role="button">Zobraz
          </div>
        </div>
      </div>
    </div>
  </div>
</section>
```

## DNEŠNÍ POLEDNÍ MENU

Hrachová polévka s opečenou houskou

1. Indické butter chicken se smaženou cizrnou, rýže Basmati ..... 149 Kč

2. Smažený česnekový řízek z vepřové kotlety, bramborový salát s majonézou ..... 159 Kč

3. Barbecue Pulled Pork Hot Dog, smažené bramborové hranolky (máslová houska plněná trhaným vepřovým masem se sýrem  
čedar a křupavou cibulkou) ..... 179 Kč

ZOBRAZIT MENU

# Další zdroje

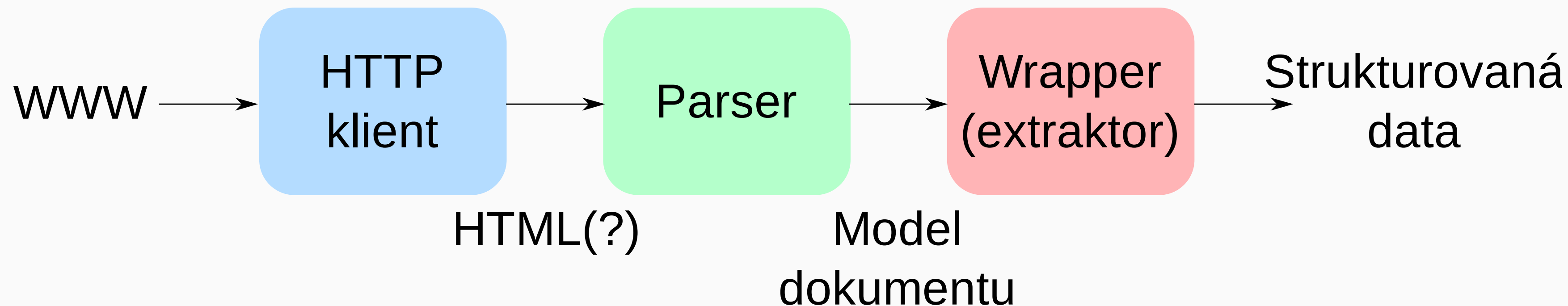
- Komerční
  - e-shopy, realitní servery, letenky, sportovní výsledky, sledování konkurence
- Výsledky vyhledávání
  - Např. hlídání pozice
- Veřejné rejstříky
  - jízdní řády
  - živnostenský rejstřík
  - statistický úřad
  - weby zastupitelstev
- Kontrola reklamy
- ∞ dalších

# Dílčí problémy

- Získání zdrojových dat
  - Jak stáhnout potřebné dokumenty z WWW (aby obsahovaly to, co mají?)
  - Paralelizace
- **Nalezení a extrakce dat**
  - Identifikace požadovaných údajů ve stránce
- Uložení výsledků
  - Může jich být **opravdu mnoho**

# Architektura

## Základní architektura



# Quick & Dirty

Když potřebujeme rychle a jednorázově data z jednoduššího webu a nikdo se nás nebude ptát, jak jsme to udělali.



# Shell je přítel

Motivace: [Evolution of a programmer](#)

Co se hodí, než začneme programovat:

- `wget`, `curl`
- `cat`, `grep`, `sed`, `cut`
- `awk` (jen pro fajnšmekry :-)

```
wget https://www.fit.vut.cz/study/courses/ -O out.html  
cat out.html | grep 'list-links__link' | sed 's/<[^<>]*>/;/g' | sed 's/;;*/;/g'  
cat data.csv | cut -f2 -d';'
```

```
wget https://www.fit.vut.cz/study/courses/ -O - | grep 'list-links__link'
```

# Totéž v pythonu

```
import urllib.request
import re

fid = urllib.request.urlopen('https://www.fit.vut.cz/study/courses/')
webpage = fid.read().decode('utf-8')
for line in webpage.split('\n'):
    if ('list-links__link') in line:
        line = re.sub(r"<[^<>]*>", ";", line);
        line = re.sub(r";;*;", ";", line);
        print(line)
```

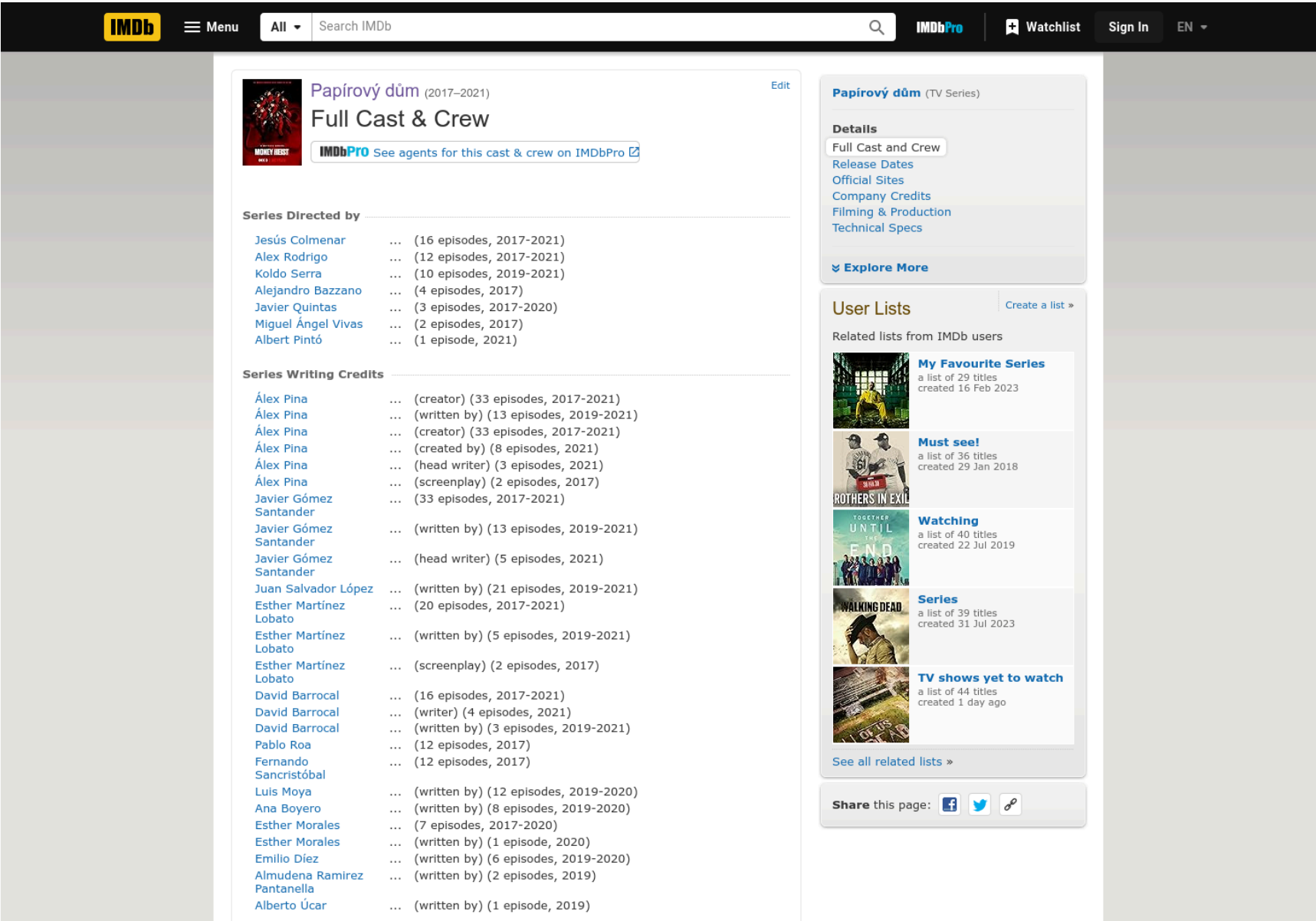
```
import java.io.*;
import java.net.*;

public class Courses {

    public static void main(String[] args) {
        try {
            URI url = new URI("https://www.fit.vut.cz/study/courses/");
            HttpURLConnection con = (HttpURLConnection) url.toURL().openCon


            BufferedReader in = new BufferedReader(
                new InputStreamReader(con.getInputStream()));
```

# Omezení jednoduchého přístupu




Zdrojová stránka – regex?!

# Omezení jednoduchého přístupu



UNION  
CYCLISTE  
INTERNATIONALE

[Inside UCI](#) [Cycling for All](#) [Para](#) [Women's](#) [Development](#) [Cycling Esports](#)



CENTRE MONDIAL DU CYCLISME  
UCI  
WORLD CYCLING CENTRE

ROADTRACKMOUNTAIN BIKEBMX RACINGBMX FREESTYLETRIALSCYCLO-CROSSINDOOR

# MOUNTAIN BIKE

[Overview](#) [About](#) [News](#) [Calendar](#) [Events](#) [Teams](#) [Results](#) [Rankings](#) [Rules and regulations](#)


## Rankings

RACE TYPE

CATEGORY

SEASON


AllMen Elite2020

 **Cross-country Ranking Men Elite**


**Individual Ranking**

Last Update  
06/10/2020


Leader

 **SCHURTER Nino (SSR)**  
1989 points

2nd rank

 **AVANCINI Henrique (CFR)**  
1945 points


3rd rank

 **KERSCHBAUMER Gerhard (BTU)**  
1339 points


**Nation Ranking**

Last Update  
06/10/2020


Leader

 **SWITZERLAND**  
4411 points

2nd rank

 **FRANCE**  
3675 points

3rd rank

 **ITALY**  
3153 points

UPA -- Extrakce dat z webu

13 / 50

# Omezení jednoduchého přístupu

Rankings

Cross-country Ranking Men Elite

SEASON

2020

DATE

Most Recent Ranking

EXPORT RANKINGS

Individual Ranking

Ranking Date

06/10/2020

Leader

SCHURTER Nino (SSR)

1989 points

Nation Ranking

Ranking Date

06/10/2020

Leader

SWITZERLAND

4411 points

NAME

TEAM

NATION

All

Rank	Rider	Nation	Team	Age	Points
1	SCHURTER Nino	<div></div> SUI	SCOTT - SRAM MTB RACING	34	1989
2	AVANCINI Henrique	<div></div> BRA	CANNONDALE FACTORY RACING	31	1945
3	KERSCHBAUMER Gerhard	<div></div> ITA	TORPADO URSUS	29	1339
4 <div>↑+1</div>	KORETZKY Victor	<div></div> FRA	KMC - ORBEA	26	1318
5 <div>↑+3</div>	VADER Milan	<div></div> NED	KMC - ORBEA	24	1279
6 <div>↓-2</div>	FLUECKIGER Mathias	<div></div> SUI	THÖMUS RN SWISS BIKE TEAM	32	1172
7 <div>↓-1</div>	TEMPIER Stephane	<div></div> FRA	TREK FACTORY RACING XC	34	1141
8 <div>↓-1</div>	SARROU Jordan	<div></div> FRA	ABSOLUTE - ABSALON - BMC	28	1096
9 <div>↑+3</div>	CINK Ondřej	<div></div> CZE	KROSS RACING TEAM	30	988
10 <div>↑+4</div>	COOPER Anton	<div></div> NZL	TREK FACTORY RACING XC	26	971
11 <div>↓-1</div>	FORSTER Lars	<div></div> SUI	SCOTT - SRAM MTB RACING	27	950

# Omezení jednoduchého přístupu

**CAS FIT VUT Login**

Zadejte [FIT login](#) a [heslo](#) a klikněte na Log In.  
Please enter your FIT login and password and click the "Log In".

Přihlášení si vyžádala aplikace/Login requested by application **cosign-FIT-Email**

**FIT Login**

**FIT Password**

Need [an account or password help?](#)

**Pokud jste tak ještě neučinili, [nainstalujte si certifikát CA VUT](#)**

[CAS FIT](#) je "Centrální Autentizační Server" FIT pro Webové aplikace. Zajišťuje společnou autentizaci pro Webové aplikace na různých serverech. Pro ověření autentizace používá "Cookie" (ukládání Cookie musí být povoleno v prohlížeči alespoň pro doménu .fit.vutbr.cz).

Vaše IP adresa: 147.229.12.203

- Přihlašovací formulář s přesměrováním (a možným zabezpečením proti strojovému vyplnění)
- Zvládnutelné, ale komplikované

# Modely dokumentů

Když regulární výrazy nestačí.



# Modely HTML dokumentů

- Řetězec znaků
  - Jednoduchá implementace, rychlost, škálovatelnost
  - Regulární výrazy, HLRT wrappery
- Řetězec tokenů
  - Lexikální analyzátor rozpozná tagy, entity, text, apod.
  - HLRT wrappery
- Hierarchické modely
  - Nejčastěji DOM
  - Případně „odlehčená“ varianta

# Wrapper

- Uvažujeme  $n$  datových polí, které se mají extrahovat z dokumentu
- Různé třídy wrapperů: LR, HLRT, ...
- HLRT wrapper:
  - **H**ead – podřetězec před datovým blokem
  - **L**eft – levý oddělovač (pro každé datové pole)
  - **R**ight – pravý oddělovač (pro každé datové pole)
  - **T**tail – podřetězec za datovým blokem

$$wrapper = (h, t, l_1, r_1, l_2, r_2, \dots, l_n, r_n)$$

# Řetězce tokenů

- Událostmi řízené parsery, např. `html.parser` v pythonu

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print("Encountered a start tag:", tag)

    def handle_endtag(self, tag):
        print("Encountered an end tag :", tag)

    def handle_data(self, data):
        print("Encountered some data  :", data)
```

<https://docs.python.org/3/library/html.parser.html>

# Document Object Model

- HTML kód reprezentovaný jako **strom objektů**
- Různé typy objektů (viz [přednášky IIS](#))

# HTML Element (prvek)

- Úsek dokumentu vymezený *značkami* (tagy)

```
<p>Obsah elementu</p>
```

```
<div class="menu" id="mainmenu">  
Obsah elementu<br> Další obsah elementu.  
</div>
```

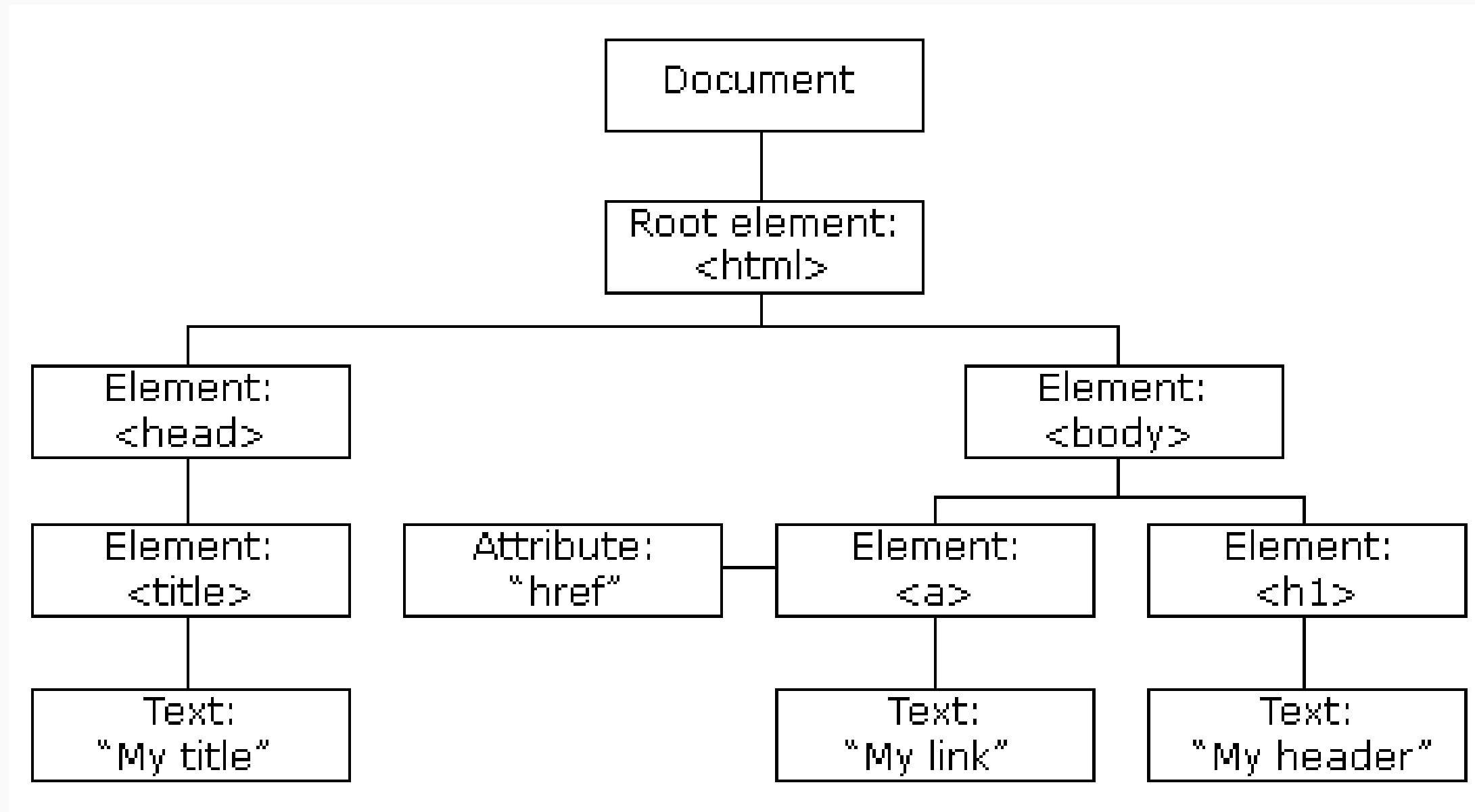
```
<div>Nějaký <em>zvýrazněný</em> text.</div>
```

- Vždy právě jeden *kořenový element*

# DOM strom

- Kořenový element je typu **Document**
- Má jednoho potomka typu **Element** ~ Document element (root)
- Element může mít potomky typu
  - **Element** – vnořené prvky
  - **text** – textový obsah, vždy listový uzel
  - výjimečně jiné (např. **Entity**)

# DOM strom



[https://www.w3schools.com/whatis/whatis\\_htmlidom.asp](https://www.w3schools.com/whatis/whatis_htmlidom.asp)

# Navigace v DOM stromu

- Standardní metody DOM tříd `Document` a `Element`
  - Vyhledání elementů: `getElementById()`, `getElementsByTagName()`
  - Navigace ve stromu: `parentNode`, `childNodes`, ...
  - Přístup k obsahu: `textContent`
- CSS selektory
  - Výsledkem je vždy *množina elementů*
  - `#main header .info`
- XPath
  - Výsledkem je také *množina elementů*
  - `*[@id="main"]//table/tr[position() > 3]`



# XPath

- Původně pro XML dokumenty, ale podporováno i některými knihovnamy pro HTML
- Oproti CSS složitější syntaxe, ale více možností:
  - Obecný výraz pro vlastnosti elementu v `[]` zahrnující hodnoty atributů, pořadí elementu a další
  - Navigace různými směry („osy“)
- Viz např. [Dokumentace na MDN](#)

```
var res = document.evaluate('//head/title', document.documentElement,  
    null, XPathResult.ANY_TYPE, null);  
console.log(res.iterateNext().textContent);
```

# Praktické použití DOM

- Plnohodnotný HTML 5 DOM parser je obtížné najít
  - Prakticky jen ve webovém prohlížeči
- V praxi často zjednodušené parsery s vlastním rozhraním
  - Python: [BeautifulSoup](#)
  - Java: [jSoup](#)
  - JavaScript: [cheerio](#)

# BeautifulSoup

```
from bs4 import BeautifulSoup
from urllib.request import urlopen

page = urlopen("https://www.fit.vut.cz/study/courses/")
html = page.read().decode("utf-8")
soup = BeautifulSoup(html, "html.parser")
rows = soup.select("#list")[0].find_all("tr")
for row in rows:
    cells = row.find_all('td')
    out = "";
    for cell in cells:
        out = out + cell.text + ";
```

```
Document doc = Jsoup.connect("https://en.wikipedia.org/").get();
log(doc.title());
Elements newsHeadlines = doc.select("#mp-itn b a");
for (Element headline : newsHeadlines) {
    log("%s\n\t%s",
        headline.attr("title"), headline.absUrl("href"));
}
```

- Také disponuje podmnožinou standardního DOM rozhraní

# cheerio

```
const cheerio = require('cheerio');
const request = require('request');

request({
  method: 'GET',
  url: 'https://www.fit.vut.cz/study/courses/'
}, (err, res, body) => {
  let $ = cheerio.load(body);

  let rows = $('#list tr');
  rows.each(function(i, tr) {
    let line = ";
```

# Automatizace prohlížeče

Když se ke stránce nedá dostat na jeden HTTP GET.

# Mechanical Soup

- Automatizace „prohlížeče“ pro [BeautifulSoup](#)
  - [Stránky projektu](#)
- Třídy a metody simulující základní operace HTTP
  - „Klikání na odkazy“ – zjištění cíle a generování GET požadavku
  - „Vyplnění formulářů“ – zjištění `action` a `method`, vyplnění hodnot polí a vyslání příslušného požadavku.
- JavaScript není podporován

# Puppeteer

- Chrome browser dálkově ovládaný z node.js  
<https://github.com/puppeteer/puppeteer>
- Lze řídit navigaci prohlížeče
  - Zadání URL, klikání na odkazy, vyplňování formulářů
  - [API dokumentace](#)
- Vykonání JS kódu v prohlížeči (`page.evaluate()`)
  - Např. pro extrakci obsahu z DOM



# Playwright

- Novější alternativa k puppeteer (původní vývojový tým)  
<https://playwright.dev>
- Velmi podobné API i funkčnost
- Podpora více prohlížečů
  - Chromium/Chrome, Firefox, WebKit
- Více vývojových platforem
  - Node.js, Java, Python, .NET
- Mírně lepší dokumentace, možnosti konfigurace, ...

# Puppeteer Pros & Cons

- + Možnost navigace
- + Pohodlná extrakce dat
  - DOM, CSS Selektory, XPath, jakýkoliv JavaScript
- - Časově i prostorově náročné řešení
  - Spouští se celý Chrome
- - Náročné ošetření vnějších podmínek
  - Např. časové souběhy, regionální verze stránek, ...
- - Obtížnější ladění
  - Část JS kódu běží v node.js, část v prohlížeči (různá prostředí)

# Alternativy

Je analýza HTML kódu jediná možnost?

# Využití API

- Některé stránky načítají zajímavý obsah dynamicky JavaScriptem
  - `XMLHttpRequest` nebo `fetch()`
  - (aka AJAX)
- Zdrojem dat je *HTTP endpoint*, který typicky vrátí
  - Útržky HTML kódu
  - *Serializovaná strukturovaná data* – JSON, XML, ...

Opět např. <https://www.uci.org/road/rankings>

# Využití API

- Mírně jednodušší přístup k datům
  - Obvykle stačí jeden HTTP dotaz (GET nebo POST)
  - Parsujeme strukturovaný formát
- Formát dat může být ještě proměnlivější, než webová stránka
  - Čistě interní formát tvůrců aplikace
- Snaha komplikovat přístup třetích stran
  - Autorizační tokeny apod.
- Existují veřejné endpointy s dobře dokumentovaným formátem dat
  - Např. [Portál veřejně přístupných dat EU](#)

# Anotace webových stránek

- [Microformats](#)
  - Anotace HTML elementů pomocí předdefinovaných hodnot **class**
  - Úzká množina definovaných formátů
  - Snadná implementace do existujícího webu
- Sémantické technologie, např. [RDFa](#)
  - Rozšíření HTML o nové atributy (**resource**, **property**, ...)
  - Umožňuje transformaci HTML na *linked data* reprezentovaná pomocí RDF
  - Identifikace objektů a vlastností pomocí **URI**
  - Existuje celá řada slovníků (*ontologií*) pro různé domény
    - Např. [FOAF](#), [schema.org](#), ...
- Viz přednáška [Sémantický web](#)

# Budoucnost

Mohou stroje pracovat za nás (programátory)?



# Současný stav

- Manuální programování, aka **Webscraping**
  - Ruční tvorba programů, které z HTML kódu extrahují, co je třeba
- Platformy pro zpřístupnění obsahu dokumentů přes API
  - Integrace ručně vytvořených scraperů
  - **Přístup „manufaktura“**





# Inteligentní extrakce

Tzn. bez „ruční práce“ v podobě hledání elementů, regulárních výrazů, CSS selektorů, XPath výrazů, apod.

## 1. **Strojové učení**

- „Naučení“ extraktoru na anotovaných příkladech

## 2. **Jazykové modely**

- Text dokumentu nebo kód jako součást promptu

## 3. **Modelem řízená extrakce**

- Specifikace předpokládané struktury dat (ER diagram?, *ontologie*, ...)
- Nalezení výskytu požadovaných skupin dat ve zdrojové stránce

# Strojové učení – scénář

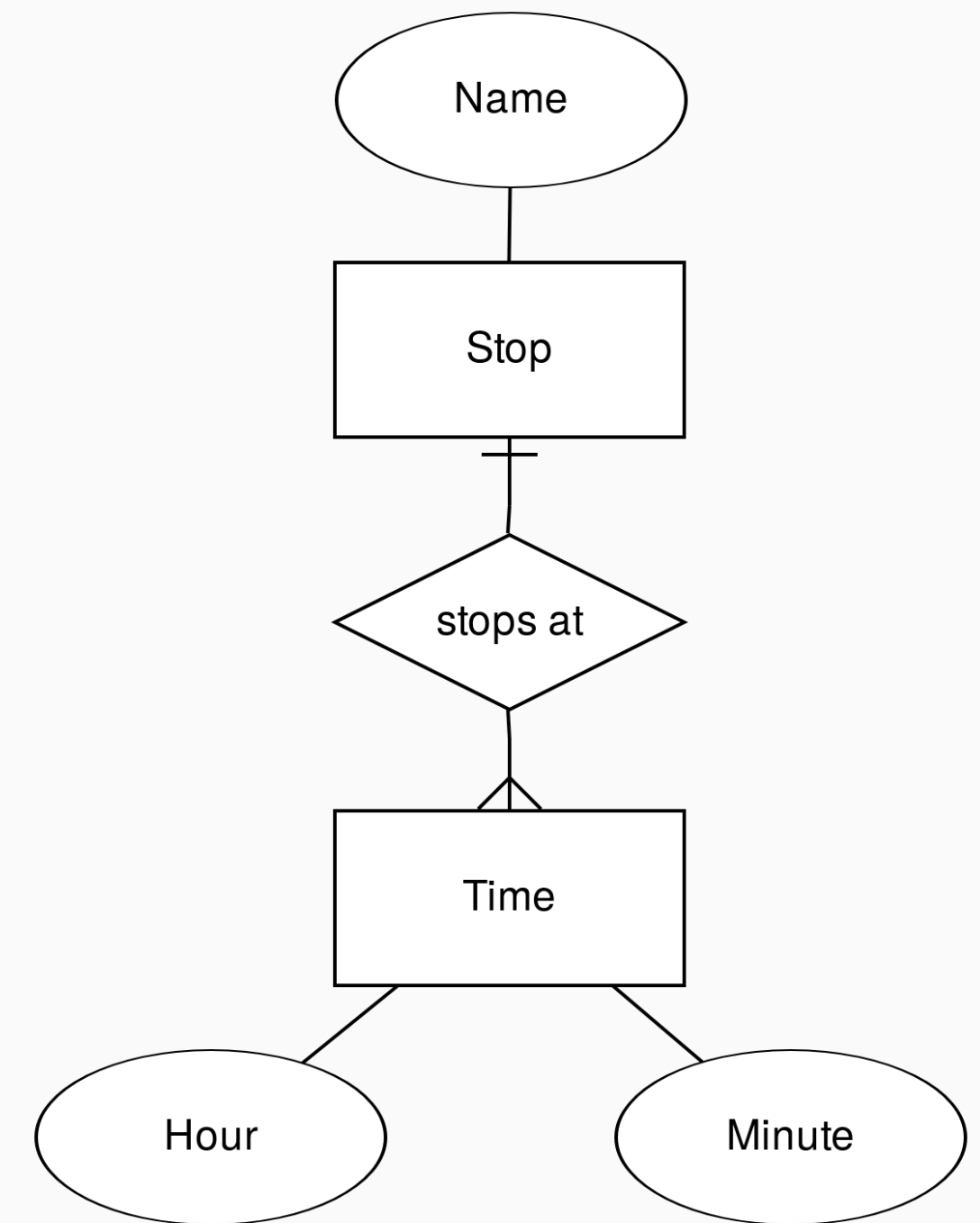
- Trénovací množina dokumentů
  - Obvykle dokumenty z jednoho zdroje
  - Anotace částí obsahu, které se mají extrahovat
  - Odvození pravidel pro extrakci
- Množina nových, neznámých dokumentů
  - Extrakce dat na základě odvozených pravidel

# Strojové učení – metody

- Sekvenční modely stránky (znaky, tokeny)
  - Inference gramatik (*wrapper induction*), skryté Markovovy modely, ...
- Hierarchické modely
  - Zobecněný DOM (odstranění implementačních detailů)
  - Stromové automaty
- Vizuální modely dokumentů
  - Segmentace stránek
  - Klasifikace na základě vizuálních rysů

# Modelem řízená extrakce

- Vstup: Entity, atributy, vztahy
- Přibližné rozpoznání jednotlivých údajů
  - Regulární výrazy
  - Klasifikace textu nebo vizuálních vlastností
  - Mapování na databázi
- Nalezení datových záznamů
  - Využití pravidelnosti, opakující se vzory



# Jazykové modely

- Předáme text nebo kód dokumentu a ptáme se

Je-li toto stránka produktu, jak se tento produkt jmenuje a kolik stojí?

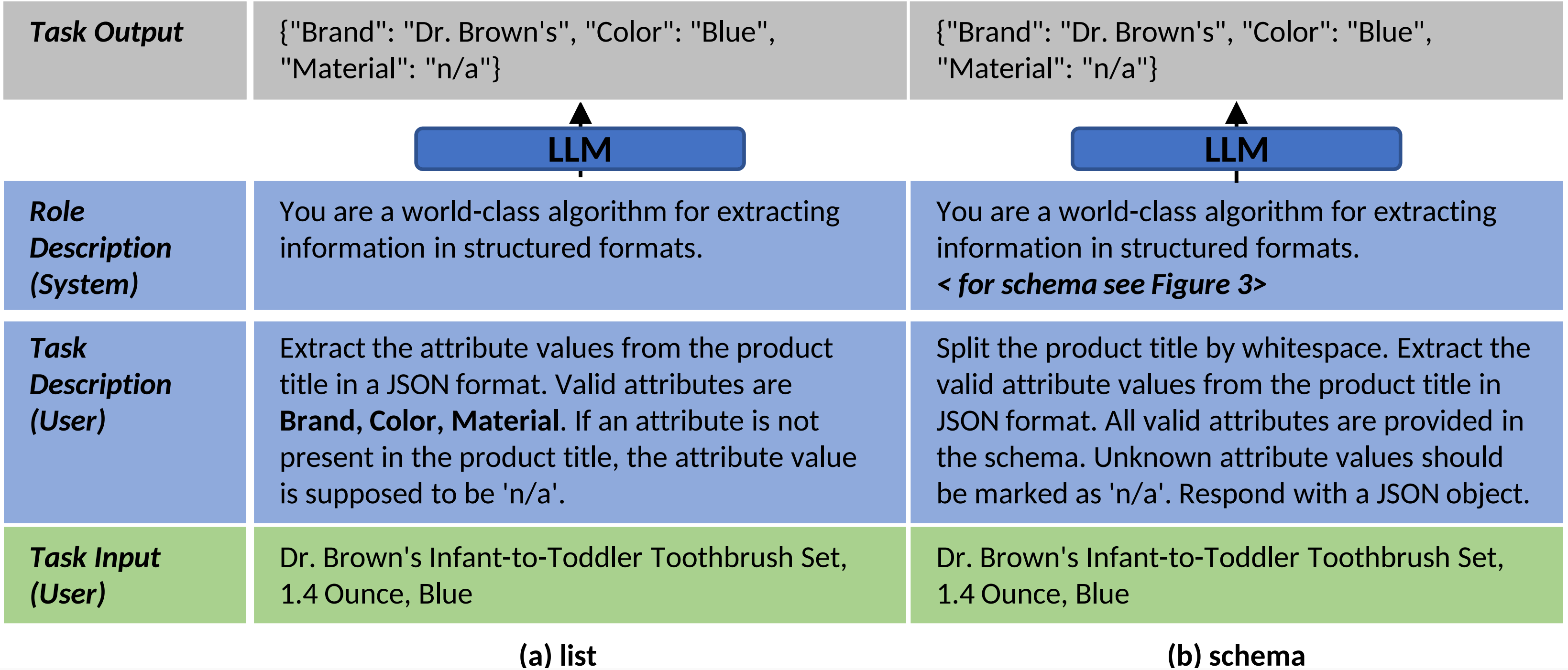
- V případě webových stránek poněkud drahý přístup
  - Použití LLM se platí per-token
  - Lokální použití se platí výpočetní náročností

# Jazykové modely - prompty

- Zero-shot prompt
  - Pouze instrukce (cíl extrakce), žádné příklady
- One (few)-shot prompt (~ )
  - Příklady hodnot atributů
  - Příklady zdrojových dat a očekávaných výsledků
- Specifikace cílového formátu odpovědi
  - Instrukce (použij JSON, názvy atributů)
  - Schéma (Ukázka JSON s příklady, JSON Schema, ...)

A. Brinkmann et al.: [ExtractGPT: Exploring the Potential of Large Language Models for Product Attribute Value Extraction](#)

# Příklad promptu - zero shot



# Příklad promptu s příklady

<i><b>Role Description (System)</b></i>	You are a world-class algorithm for extracting information in structured formats.
<i><b>Task Description (User)</b></i>	Extract the attribute values from the product title in a JSON format. Valid attributes are <b>Brand, Color, Material</b> . If an attribute is not present in the product title, the attribute value is supposed to be 'n/a'.
<i><b>Demonstration – Task Input (User)</b></i>	Quip Kids Electric Toothbrush Set - Electric toothbrush with multi-use cover (Green)
<i><b>Demonstration – Task Output (Assistant)</b></i>	{"Brand": "Quip", "Color": "Green", "Material": "n/a"}
<i><b>Task Description (User)</b></i>	Extract the attribute values from the product title in a JSON format. Valid attributes are <b>Brand, Color, Material</b> . If an attribute is not present in the product title, the attribute value is supposed to be 'n/a'.
<i><b>Task Input (User)</b></i>	Dr. Brown's Infant-to-Toddler Toothbrush Set, 1.4 Ounce, Blue



# AI agenti

- LLM nejen analyzuje vstup, ale i řídí nástroje
  - Popíšeme schopnosti a API dostupných nástrojů textově
  - Dáme instrukci k vykonání činnosti
  - LLM generuje sekvenci příkazů pro nástroje a analyzuje výstup
- Mnoho dostupných nástrojů
  - Např. [LangChain](#) obsahuje i [rozhraní pro Playwright](#)

# A to je vše!

Demo kód:

<https://github.com/DIFS-Teaching/webscraping>

Dotazy?