Прикладные задачи анализа данных

Семинар 3

Краулинг, scrapy

Национальныи Исследовательский Университет
Высшая Школа Экономики

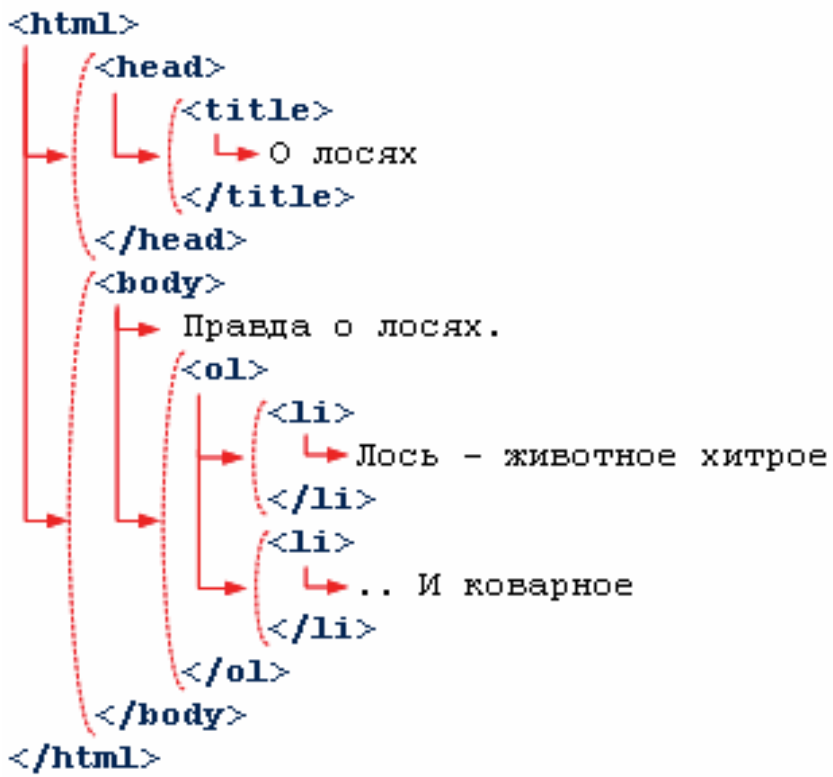
1 февраля 2018

Document Object Model

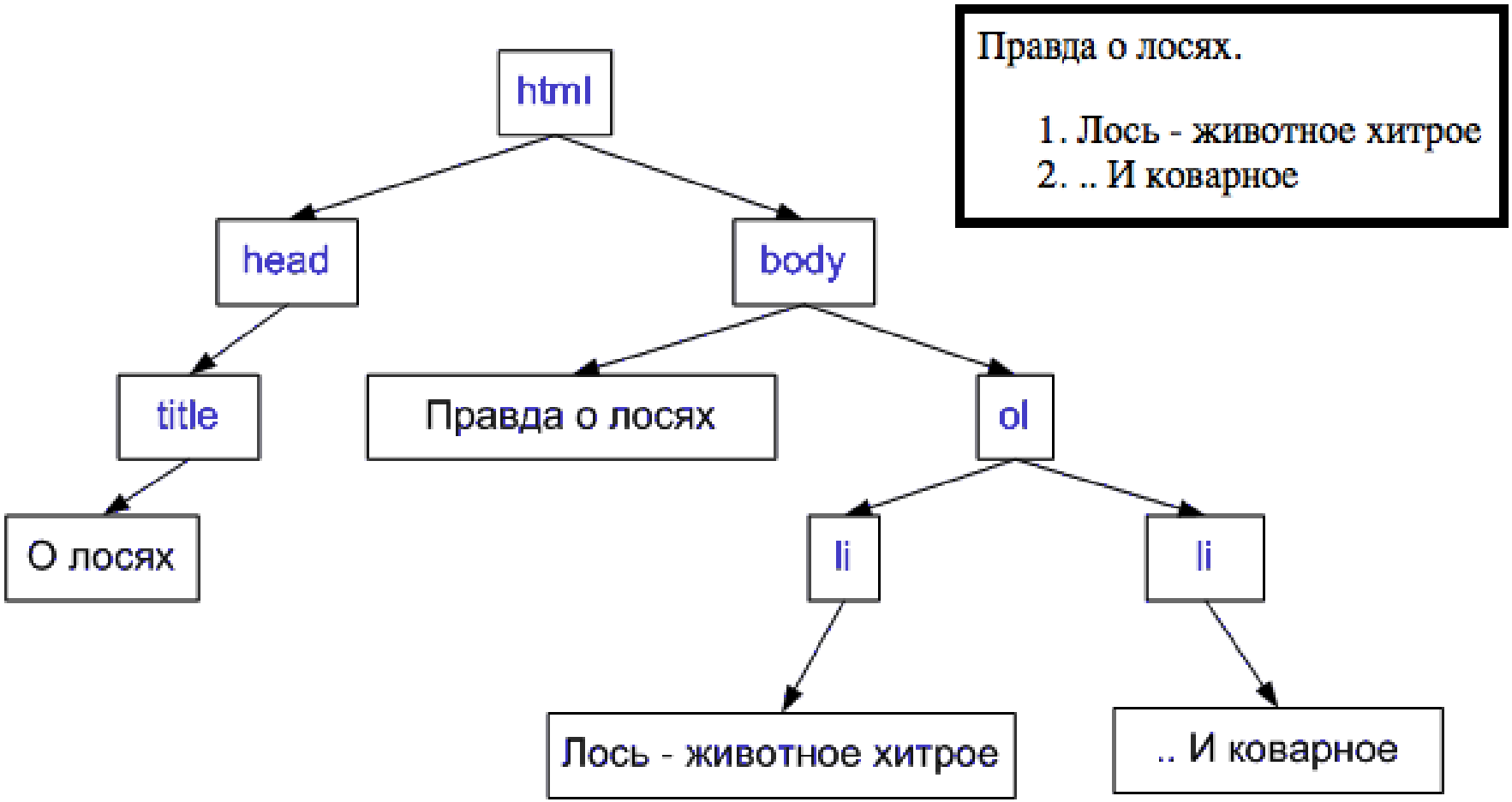
- Объектная модель, используемая для XML/HTML-документов.
- Представление документа в виде дерева тегов.
- `http://javascript.ru/tutorial/dom/intro`
- Чтобы ознакомиться с DOM страницы, откройте в браузере Developer Tools (Elements).

DOM-структура

```
01 <html>
02     <head>
03         <title>
04             О лосях
05         </title>
06     </head>
07     <body>
08         Правда о лосях.
09         <ol>
10             <li>
11                 Лось - животное хитрое
12             </li>
13             <li>
14                 .. И коварное
15             </li>
16         </ol>
17     </body>
18 </html>
```



DOM-дерево


Правда о лосях.
1. Лось - животное хитрое
2. .. И коварное
html
head
body
title
Правда о лосях
ol
О лосях
li
li
Лось - животное хитрое
.. И коварное

Атрибуты тэгов

```
<!-- begin photo one -->
<dl>
  <dd class="first-image">
    <dl>
      <dt>Inside the Sarkophargus</dt>
      <dd ><a href="#block-1"><img src="images/small/pic-one-small.jpg" width="150"
       height="225" alt="pic one small" ></a></dd>
    </dl>
  </dd>
  <dd class="second-image">
    <dl id="block-1">
      <dt>Inside the Sarkophargus</dt>
      <dd class="first-para"><img src="images/large/pic-one-large.jpg" width="400"
      height="600" alt="pic one large" class="animation"><span class=
      "spinning-wheel"></span></dd>
      <dd class="second-para">Quisque nisi risus, porta sit amet venenatis vitae,
      posuere eget mauris</dd>
      <dd class="third-para"><a href="#photo-gallery">X Close</a></dd>
    </dl>
  </dd>
</dl>
<!-- end photo one -->
```

Whenever you define a CSS code for .content like following

```
.content {
    width: 300px;
}
```

It means that a browser will find all items with class="content":

```
<div class="content">something</div>
```

and display it with width equal 300px.

Поиграем? `http://flukeout.github.io/`
(синтаксис:
`https://learn.javascript.ru/css-selectors`)

- асинхронный фреймворк для сбора данных
- создает «веб-паука»
- использует селекторы: XPath или CSS
- экспорт в json, csv, xml
- `http://www.w3ii.com/scrapy/scrapy_environment.html`

Пример сбора данных scrapy

- Источник примера: `https://www.8host.com/blog/web-scraping-s-pomoshhyu-scrapy-i-python-3/`
- Задача: создать «веб-паука», который обработает страницы Brickset и извлечёт данные о наборах LEGO.

Создаем паука

Передаем первую страницу результатов поиска наборов LEGO

```
import scrapy

class BrickSetSpider(scrapy.Spider):

name = "brickset_spider"

start_urls = ['http://brickset.com/sets/year-2016']
```

Набор из списка на странице

https://brickset.com/sets/year-2016

10252: Volkswagen Beetle

10252-1 | ADVANCED MODELS | VEHICLES | 2016

14 WIDE CAR | CREATOR EXPERT | D2C | LARGE SCALE VEHICLE

VOLKSWAGEN

★★★★★ 1 REVIEW

PIECES 1167

RRP $99.99, 89.99€ | More

PPP 8.6c, 7.7c

PACKAGING Box

AVAILABILITY LEGO exclusive

FIRST SOLD USA: Aug 16, UK/EU: Aug 16

INSTRUCTIONS Yes

SET TYPE Normal

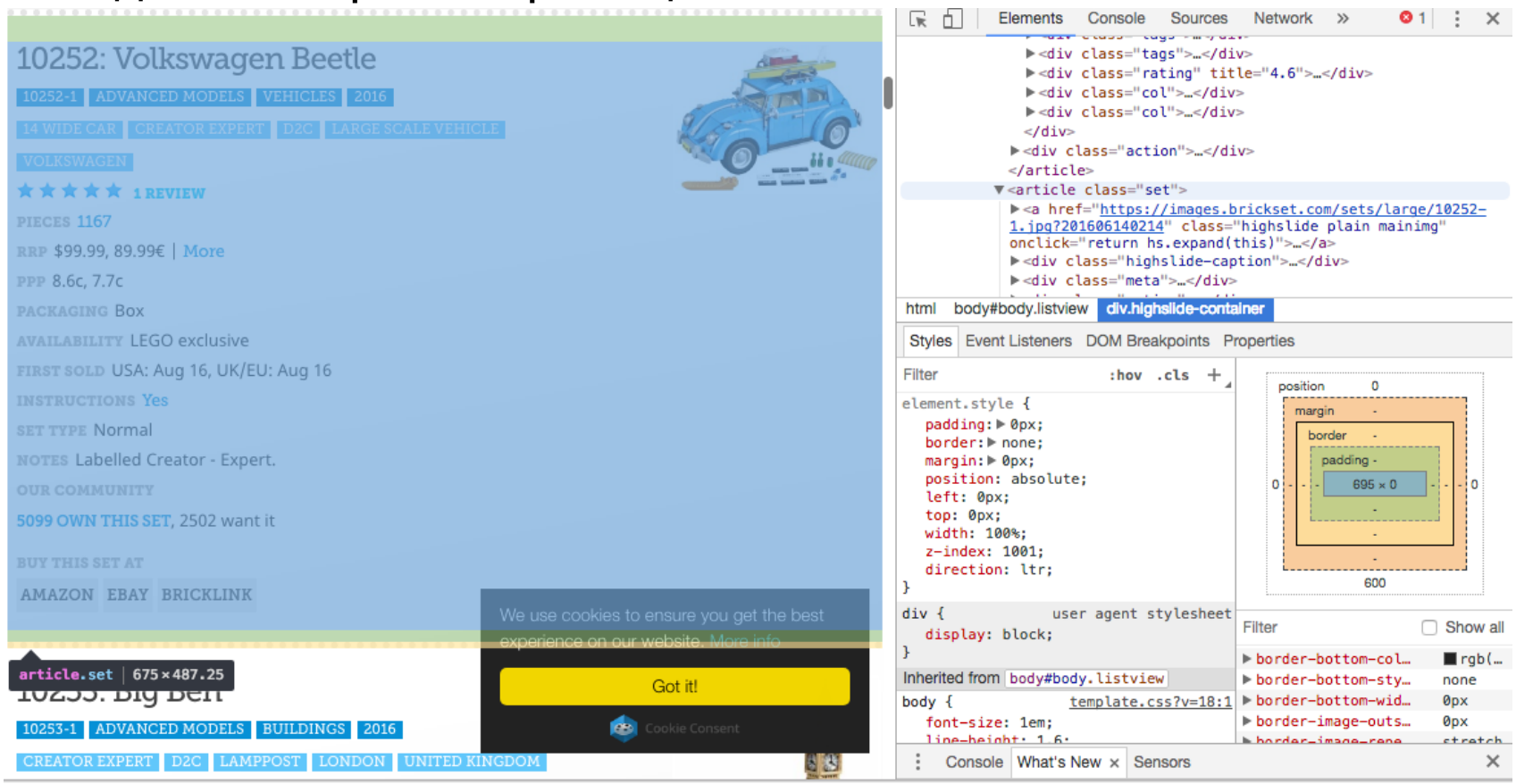
NOTES Labelled Creator - Expert.

OUR COMMUNITY

5099 OWN THIS SET, 2502 want it

Выбриаем селекторы

Каждый набор на странице имеет класс set.



Используем CSS-селекторы

Извлекаем все наборы со страницы

```
class BrickSetSpider(scrapy.Spider):
name = "brickset_spider"
start_urls = ['http://brickset.com/sets/year-2016']
def parse(self, response):
    SET_SELECTOR = '.set'
    for brickset in response.css(SET_SELECTOR):
        pass
```

Ищем и отображаем имя набора

Название наборов хранится в тегах a внутри тега h1.

```
class BrickSetSpider(scrapy.Spider):
name = "brickset_spider"
start_urls = ['http://brickset.com/sets/year-2016']
def parse(self, response):
  SET_SELECTOR = '.set'
  for brickset in response.css(SET_SELECTOR):
    NAME_SELECTOR = 'h1 a ::text'
    yield {
    'name': brickset.css(NAME_SELECTOR).extract_first()
    }
```

Фрагмент вывода

```
...
[scrapy] DEBUG: Scraped from <200
http://brickset.com/sets/year-2016>
{'name': 'Brick Bank'}
[scrapy] DEBUG: Scraped from <200
http://brickset.com/sets/year-2016>
{'name': 'Volkswagen Beetle'}
[scrapy] DEBUG: Scraped from <200
http://brickset.com/sets/year-2016>
{'name': 'Big Ben'}
[scrapy] DEBUG: Scraped from <200
http://brickset.com/sets/year-2016>
{'name': 'Winter Holiday Train'}
...
```

Ищем другие характеристики

```
NAME_SELECTOR = 'h1 a ::text'
PIECES_SELECTOR = './/dl[dt/text() =
"Pieces"]/dd/a/text()'
MINIFIGS_SELECTOR = './/dl[dt/text() =
"Minifigs"]/dd[2]/a/text()'
IMAGE_SELECTOR = 'img ::attr(src)'
yield {
'name': brickset.css(NAME_SELECTOR).extract_first(),
'pieces':
brickset.xpath(PIECES_SELECTOR).extract_first(),
'minifigs':
brickset.xpath(MINIFIGS_SELECTOR).extract_first(),
'image': brickset.css(IMAGE_SELECTOR).extract_first(),
}
```

Фрагмент вывода

```
2016-09-22 23:52:37 [scrapy] DEBUG: Scraped from <200
http://brickset.com/sets/year-2016>
{'minifigs': '5', 'pieces': '2380', 'name': 'Brick
Bank', 'image':
'http://images.brickset.com/sets/small/10251-1.jpg?
201510121127'}
2016-09-22 23:52:37 [scrapy] DEBUG: Scraped from <200
http://brickset.com/sets/year-2016>
{'minifigs': None, 'pieces': '1167', 'name':
'Volkswagen Beetle', 'image':
'http://images.brickset.com/sets/small/10252-1.jpg?
20160614021 4'}
```

Переход на следующие страницы

Определяется селектор для ссылки на следующую страницу и извлекается первое совпадение

```
NEXT_PAGE_SELECTOR = '.next a ::attr(href)'
next_page =
response.css(NEXT_PAGE_SELECTOR).extract_first()
if next_page:
    yield scrapy.Request(
    response.urljoin(next_page),
    callback=self.parse
    )
```

Смотрим в реальном времени: пример
В конце файла — упражнение!