Parsing HTML

```
bsObj.findAll("table")[4].findAll("tr")[2].find("td").findAll("div")[1].find("a")
```

- So what's wrong in doing this?!
- What are our options?
  - You need to start looking for ways to differentiate how data is organized
  - Data organization could be by means of:
    - Different types of HTML tags
    - Same HTML tag but different entities
    - Same HTML tag but different attributes
    - Advent of CSS (Cascading Style Sheets) – is it a boon or a bust for web scraping?

CSS

- CSS relies on the differentiation of HTML elements by styling them differently

```
<span class="green"></span>
```

```
<span class="red"></span>
```

- Example@ http://cs.unh.edu/~anarayan/it780/scraping/warandpeace.html
- We can easily separate these two different tags based on their class
- CSS is used heavily in modern websites and this is almost guaranteed to give you success while scraping

Example using CSS

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html = urlopen("http://cs.unh.edu/~anarayan/it780/Scraping/warandpeace.html")
bsObj = BeautifulSoup(html,"html.parser")

nameList = bsObj.findAll("span",{"class":"green"})
for name in nameList:
    print(name)

for name in nameList:
    print(name.get_text())
```

- Preserve tags
  - Keep info in a BeautifulSoup object for as long as you can
- get_text()
  - Strips out everything and leaves you with blocks of text
  - Should be the last thing you do, typically since that's likely to be your final data

find() and findall()

- Most common BeautifulSoup functions that are used

```
findAll(tag, attributes, recursive, text, limit, keywords)
find(tag, attributes, recursive, text, keywords)
```

- First two arguments are the ones mostly used
- For both green and red span tags (in the previous example):

```
.findAll("span", {"class":{"green", "red"}})
```

- Multiple tags can be queried:

```
.findAll({"h1","h2","h3","h4","h5","h6"})
```

- recursive argument is a Boolean
  - Setting it to True (default) will look into children, and children's children, for tags that match your parameters
  - Setting it to False will look into top-level tags only in your document

find() and findAll()

```
findAll(tag, attributes, recursive, text, limit, keywords)
find(tag, attributes, recursive, text, keywords)
```

- text argument will make a match based on the text content of the tags, rat than the properties of the tags themselves
  - Example to find number of times “the prince” was surrounded by tags

```
nameList = bsObj.findAll(text="the prince")
print(len(nameList))
```

- limit argument is used only in findAll method to limit the number of items retrieved (the first items from the page in the order that they occurred)
- keyword argument allows you to select tags that contain a particular attribute

```
allText = bsObj.findAll(id="text")
print(allText[0].get_text())
```

Navigating Trees

- findAll function finds tags based on their name and attribute
- How do you find a tag based on its location in a document?
  - We need to do tree navigation
  - Children & Descendants
    - Children are always exactly one tag below a parent
    - Descendants can be at any level in the tree below a parent
    - All children are descendants but not all descendants are children

Navigating Trees

- BeautifulSoup functions always deal with the descendants of the current tag selected
  - bsObj.body.h1 selects the first h1 tag that is a descendant of the body tag (not any tags located outside of the body)
  - bsObj.div.findAll("img") will find the first div tag in the document, then retrieve a list of all img tags that are descendants of that div tag
  - To find only descendants that are children, you can use the .children function

Navigating Trees

Totally Normal Gifts

Here is a collection of totally normal, totally reasonable gifts that your friends are sure to love! Our collection is hand-curate

We haven't figured out how to make online shopping carts yet, but you can send us a check to:
123 Main St.
Abuja, Nigeria
We will then send your totally amazing gift, pronto! Please include an extra $5.00 for gift wrapping.

Item Title	Description	Cost	Image
Vegetable Basket	This vegetable basket is the perfect gift for your health conscious (or overweight) friends! ***Now with super-colorful bell peppers!***	$15.00	
Russian Nesting Dolls	Hand-painted by trained monkeys, these exquisite dolls are priceless! And by "priceless," we mean "extremely expensive"! ***8 entire dolls per set! Octuple the presents!***	$10,000.52	
Fish Painting	If something seems fishy about this painting, it's because it's a fish! ***Also hand-painted by trained monkeys!***	$10,005.00	
Dead Parrot	This is an ex-parrot! ***Or maybe he's only resting?***	$0.50	

Navigating Trees

- HTML for cs.unh.edu/~anarayan/it780/Scraping/Navigation.html shown in tree form (some tags are omitted for simplicity):

```
• html
  — body
      — div.wrapper
          —  h1
          —  div.content
          — table#giftList
              — tr
                  —  th
                  —  th
                  —  th
                  —  th
              — tr.gift#gift1
                  —  td
                  — td
```

Navigating Trees

- Find only descendants that are children of the "table" tag (to get the list of product rows):

```
for child in bsObj.find("table",{"id":"giftList"}).children:
    print(child)

for sibling in bsObj.find("table",{"id":"giftList"}).tr.next_siblings:
    print(sibling)
```

- If the descendants() function is used instead of children() a lot more tags would be found within the table and printed (including img, span, etc.)

Navigating Trees

- Navigating through siblings
  - next_siblings() and previous_siblings()
- Same example, use next_siblings() for the first "tr" tag in the table:
  - Gives us all the rows of products from the product table, except the first title row
  - ***Will the above work if the first row is referenced as: bsObj.table.tr?***

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html = urlopen("http://cs.unh.edu/~anarayan/it780/Scraping/Navigation.html")
bsObj = BeautifulSoup(html,"html.parser")

for sibling in bsObj.find("table",{"id":"giftList"}).tr.next_siblings:
    print(sibling)
```

Navigating Trees

- Navigating through siblings – can also use functions next_sibling() and previous_sibling()
- Navigating through parents
  - Not very often used but after drilling down, you may want to go “up” to navigate further