Introduction to XPATH

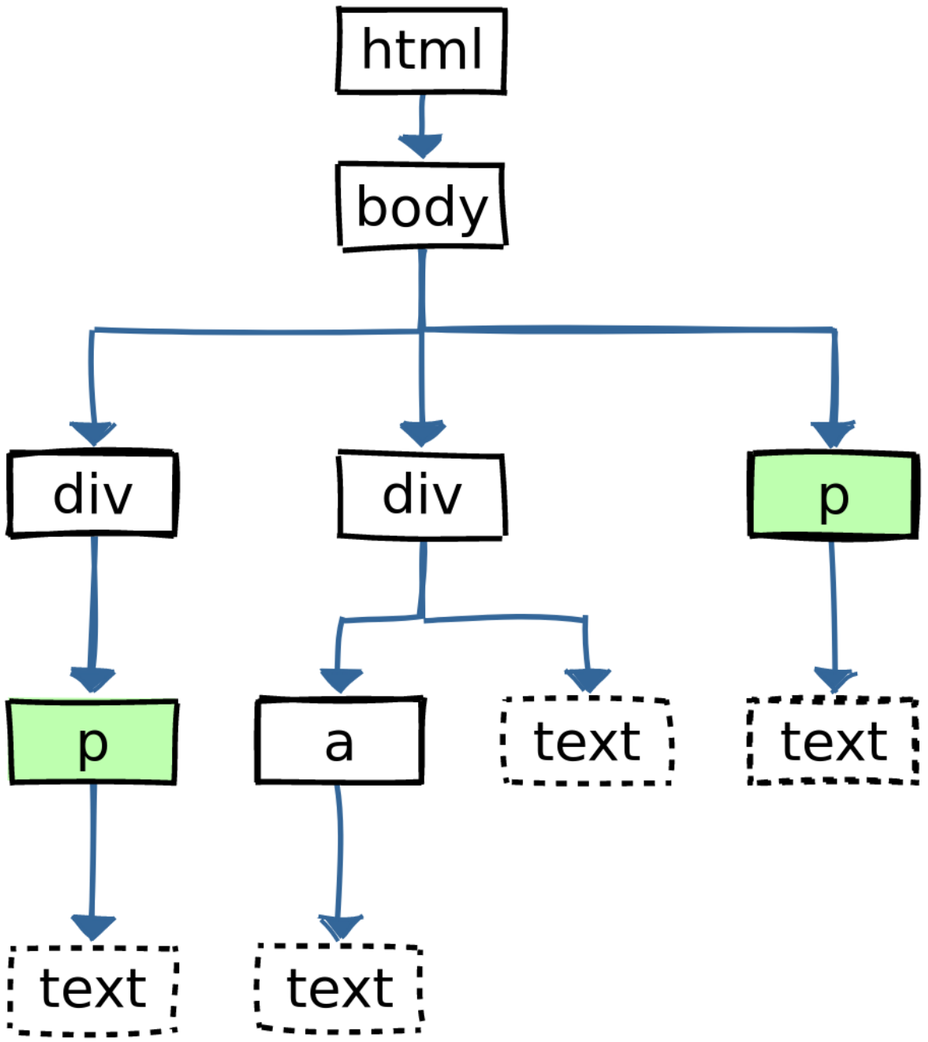
WEB SCRAPING IN R

Timo Grossenbacher
Instructor

datacamp

XML Path Language

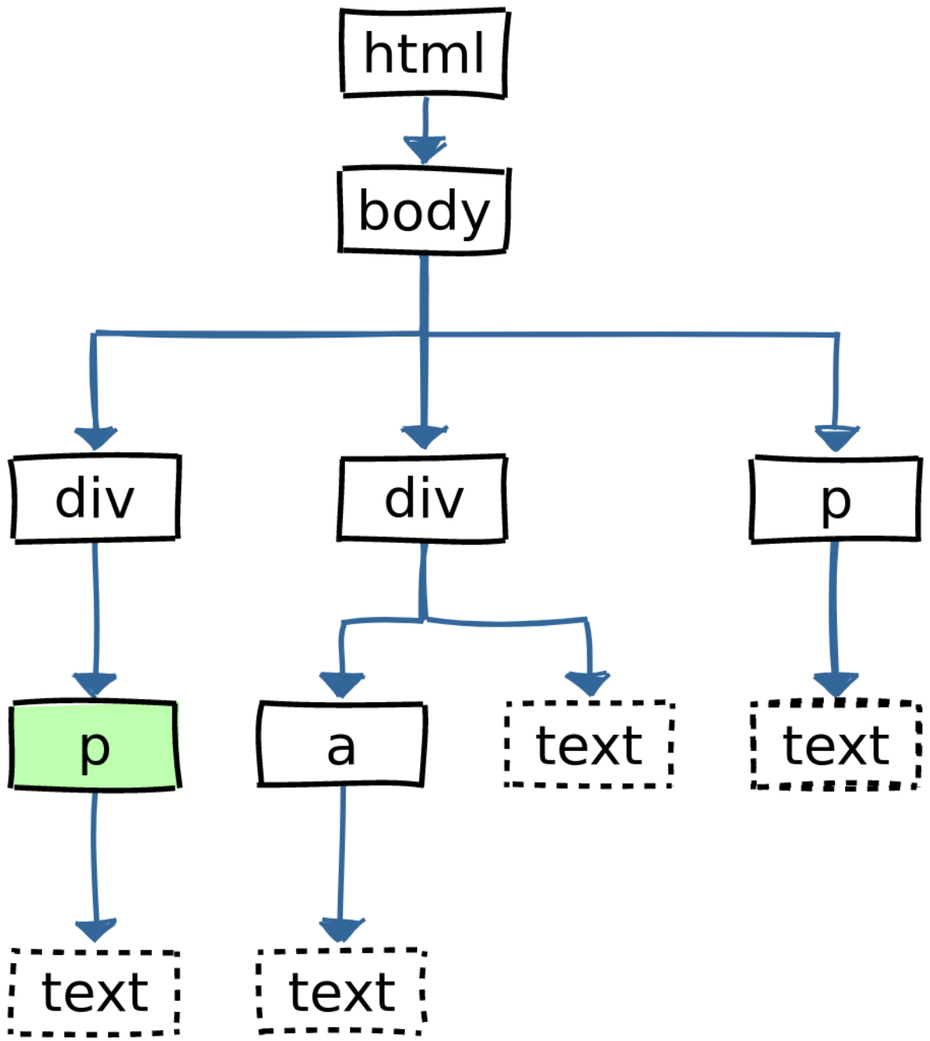
- A path through an HTML tree is formulated, e.g. `//div/p[@class = "blue"]` (equivalent to `div > p.blue`)
- Select nodes on properties of other nodes
- More advanced and customized selections possible
- For example: Select elements based on properties of their children, e.g. select only `div` elements that contain `a` nodes with a `special` class



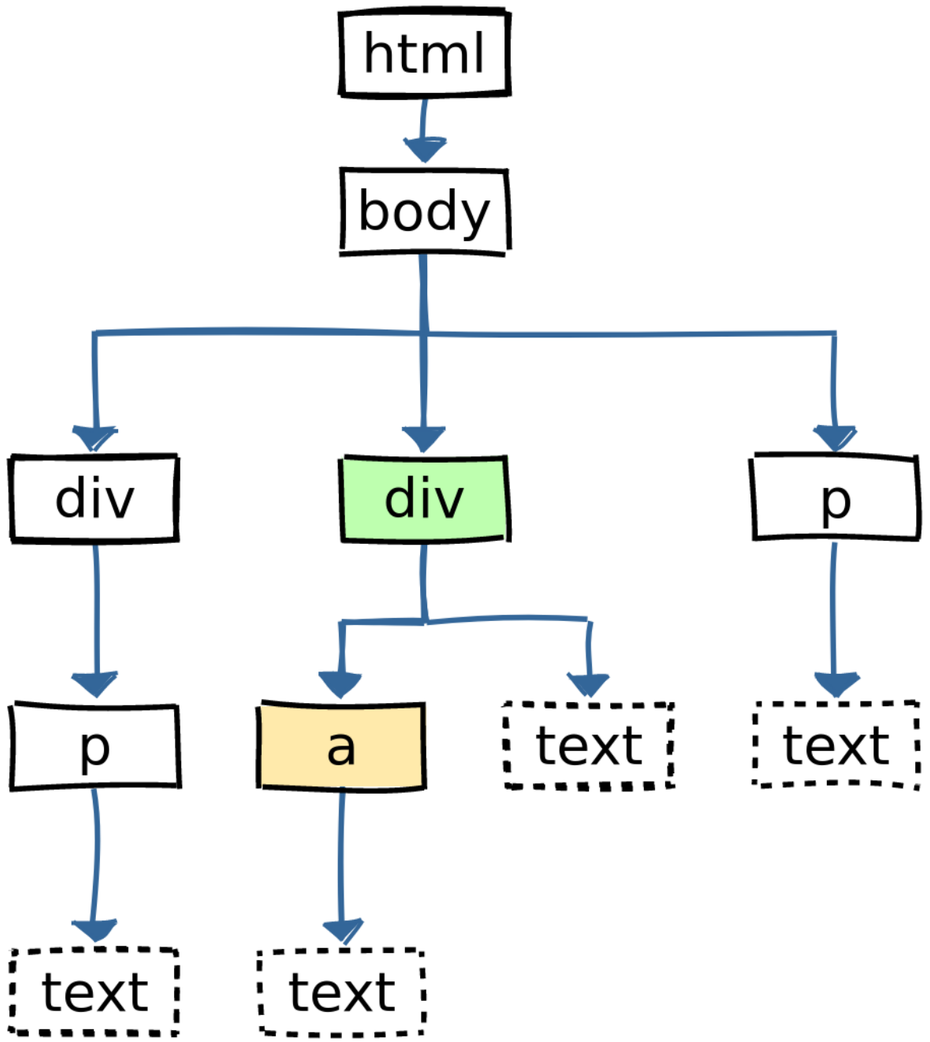

```
html %>%
    html_nodes(xpath = '//p')
# CSS selector equivalent: p
```

```
html %>%
    html_nodes(xpath = '//body//p')
# CSS selector equivalent: body p
```

```
html %>%
    html_nodes(xpath = '/html/body//p')
# CSS selector equivalent: html > body p
```




```
html %>%
    html_nodes(xpath = '//div/p')
# CSS selector equivalent: div > p
```




```
html %>%
    html_nodes(xpath = '//div[a]')
# CSS selector equivalent: none
```

Syntax: axes, steps, and predicates

- Axes: `/` or `//`
- Steps: HTML types like `span` and `a`
- Predicates: `[...]`
- Example: `//span/a[@class = "external"]` (CSS: `span > a.external`)
- Example: `//*[@id = "special"]//div` (CSS: `#special div` or `*#special div`)

Let's practice!

WEB SCRAPING IN R

XPATH functions and advanced predicates

WEB SCRAPING IN R

Timo Grossenbacher
Instructor

datacamp

The position() function

```
...
<ol>
  <li>First element.</li>
  <li>Second element.</li>
  <li>Third element.</li>
  <li>Fourth element.</li>
  <li>Fifth element.</li>
</ol>
...
```

```
html %>%
  html_nodes(xpath =
               '//ol/li[position() = 2]')
# Equivalent CSS selector:
# ol > li:nth-child(2)
```

```
{xml_nodeset (1)}
[1] <li>Second element.</li>
```

More operators for the position() function

```
...
<ol>
  <li>First element.</li>
  <li>Second element.</li>
  <li>Third element.</li>
  <li>Fourth element.</li>
  <li>Fifth element.</li>
</ol>
...
```

```
html %>%
  html_nodes(xpath =
               '//ol/li[position() < 3]')
```

```
{xml_nodeset (2)}
[1] <li>First element.</li>
[2] <li>Second element.</li>
```

More operators for the position() function

```
...
<ol>
  <li>First element.</li>
  <li>Second element.</li>
  <li>Third element.</li>
  <li>Fourth element.</li>
  <li>Fifth element.</li>
</ol>
...
```

```
html %>%
  html_nodes(xpath =
               '//ol/li[position() != 3]')
```

```
{xml_nodeset (4)}
[1] <li>First element.</li>
[2] <li>Second element.</li>
[3] <li>Fourth element.</li>
[4] <li>Fifth element.</li>
```

Combining predicates

```
...
<ol>
  <li class = 'blue'>First element.</li>
  <li>Second element.</li>
  <li class = 'blue'>Third element.</li>
  <li>Fourth element.</li>
  <li class = 'blue'>Fifth element.</li>
</ol>
...
```

```
html %>%
  html_nodes(xpath =
  '//ol/li[position() != 3 and @class = "blue"]')
```

```
{xml_nodeset (2)}
[1] <li class="blue">First element.</li>
[2] <li class="blue">Fifth element.</li>
```

```
html %>%
  html_nodes(xpath =
  '//ol/li[position() != 3 or @class = "blue"]')
```

```
{xml_nodeset (5)}
...
```

The count() function

```
...
<ol>
  <li class = 'blue'>First element.</li>
  <li>Second element.</li>
  <li class = 'blue'>Third element.</li>
</ol>
<ol>
  <li class = 'red'>First element.</li>
  <li>Second element.</li>
</ol>
...
```

```
html %>%
  html_nodes(xpath = '//ol[count(li) = 2]')
```

```
{xml_nodeset (1)}
[1] <ol>\n<li class="red">...
```

```
html %>%
  html_nodes(xpath = '//ol[count(li) > 2]')
```

```
{xml_nodeset (1)}
[1] <ol>\n<li class="blue">...
```

Let's try out some functions!

WEB SCRAPING IN R

The XPATH text() function

WEB SCRAPING IN R

Timo Grossenbacher
Instructor

datacamp

```
<html>
<body>
  <table id="cast">
    <tr><td class = "actor">Arnold S.</td><td class = "role"><em>1</em> (Voice)</td></tr>
    <tr><td class = "actor">Burt R.</td><td class = "role"><em>2</em> (Choreo)</td></tr>
    <tr><td class = "actor">Charlize T.</td><td class = "role"><em>3</em> (Voice)</td></tr>
  </table>
</body>
</html>
```

```
html %>%
  html_nodes("#cast td.role") %>%
  html_text()
```

```
[1] "1 (Voice)"  "2 (Choreo)" "3 (Voice)"
```

```
<html>
<body>
  <table id="cast">
    <tr><td class = "actor">Arnold S.</td><td class = "role"><em>1</em> (Voice)</td></tr>
    <tr><td class = "actor">Burt R.</td><td class = "role"><em>2</em> (Choreo)</td></tr>
    <tr><td class = "actor">Charlize T.</td><td class = "role"><em>3</em> (Voice)</td></tr>
  </table>
</body>
</html>
```

```
html %>%
  html_nodes(xpath = '//*[@id = "cast"]//td[@class = "role"]') %>% # equal to ""#cast td.role"
  html_nodes(xpath = "./text()") %>%
  html_text(trim = T)
```

```
[1] "(Voice)"  "(Choreo)" "(Voice)"
```

```
<html>
<body>
  <table id="cast">
    <tr><td class = "actor">Arnold S.</td><td class = "role"><em>1</em> (Voice)</td></tr>
    <tr><td class = "actor">Burt R.</td><td class = "role"><em>2</em> (Choreo)</td></tr>
    <tr><td class = "actor">Charlize T.</td><td class = "role"><em>3</em> (Voice)</td></tr>
  </table>
</body>
</html>
```

```
html %>%
 html_nodes(xpath = '//*[@id = "cast"]//td[@class = "role" and text() = " (Voice)"]')
```

```
{xml_nodeset (2)}
[1] <td class="role">\n<em>1</em> (Voice)</td>
[2] <td class="role">\n<em>3</em> (Voice)</td>
```

```
<html>
<body>
  <table id="cast">
    <tr><td class = "actor">Arnold S.</td><td class = "role"><em>1</em> (Voice)</td></tr>
    <tr><td class = "actor">Burt R.</td><td class = "role"><em>2</em> (Choreo)</td></tr>
    <tr><td class = "actor">Charlize T.</td><td class = "role"><em>3</em> (Voice)</td></tr>
  </table>
</body>
</html>
```

```
html %>%
 # same as before
 html_nodes(xpath = '//*[@id = "cast"]//td[@class = "role" and text() = " (Voice)"]') %>%
 html_nodes(xpath = '..') # selects the parent (tr) of each selected td element
```

```
{xml_nodeset (2)}
[1] <tr>\n<td class="actor">Arnold S.</td>\n<td class="role">\n<em>1</em> (Voice)</td>\n ...
[2] <tr>\n<td class="actor">Burt R.</td>\n<td class="role">\n<em>3</em> (Voice)</td>\n ..
```

Let's practice!

WEB SCRAPING IN R