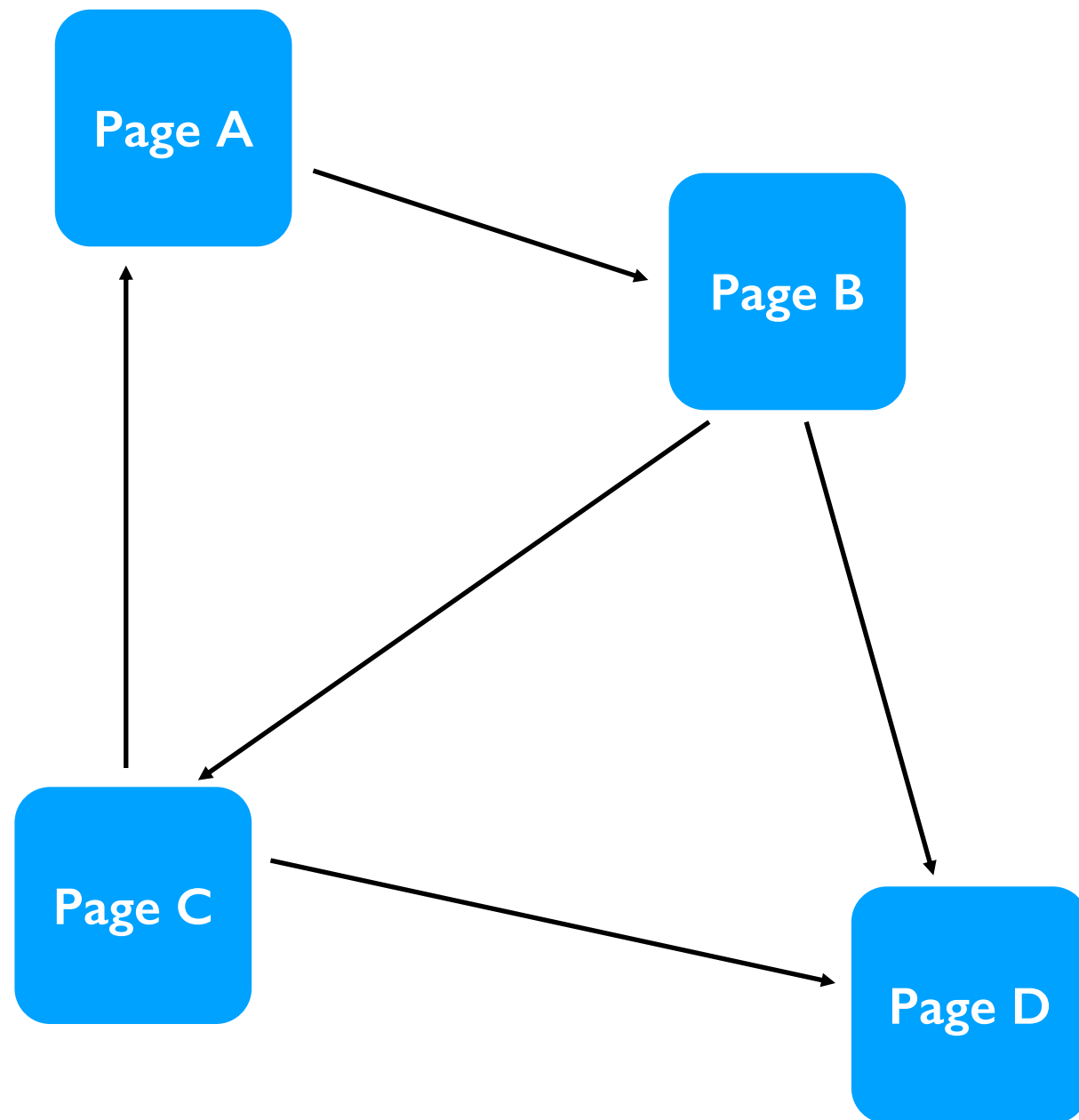


[320] Web I: Selenium

Meenakshi Syamkumar

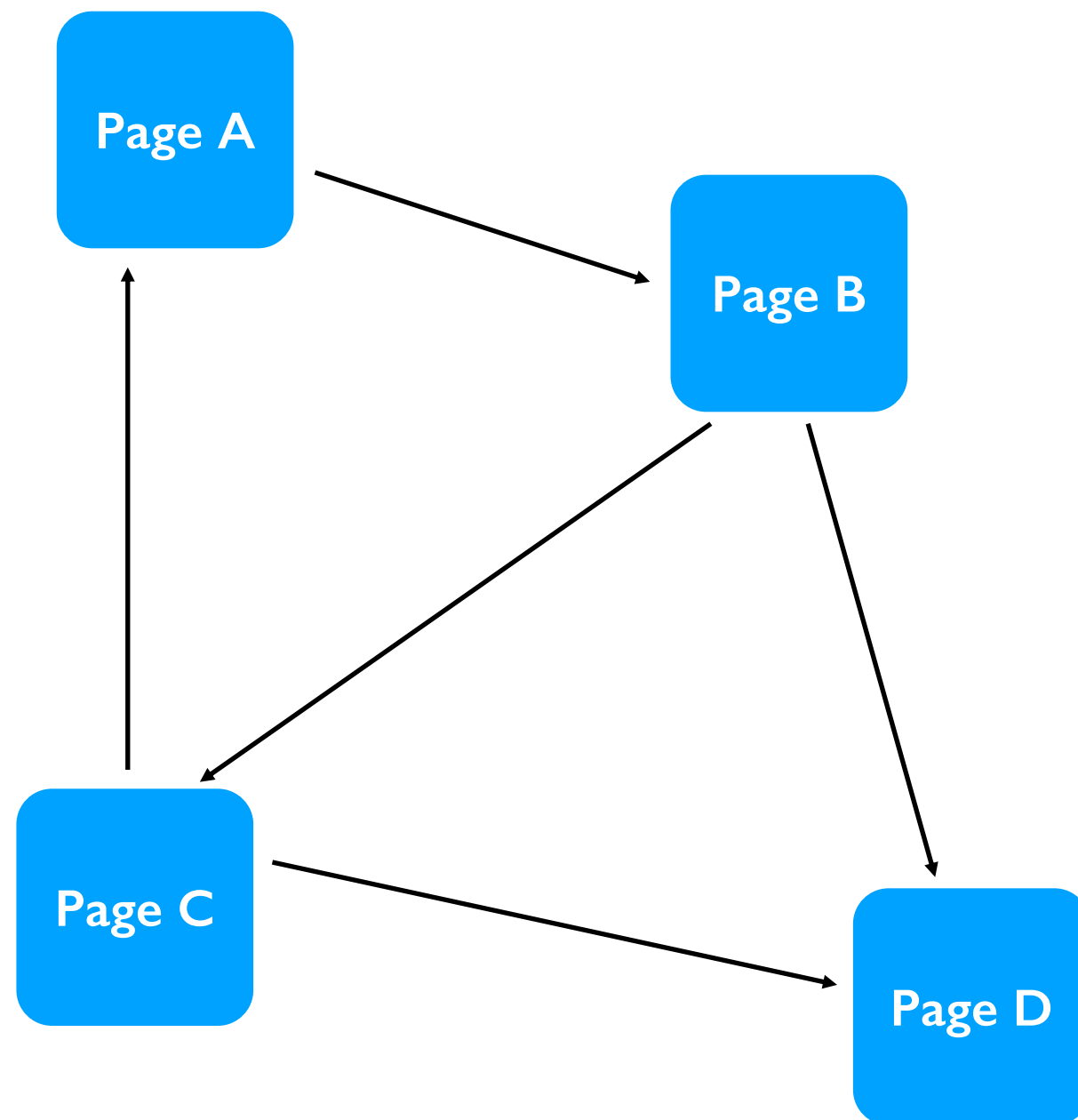
how to scrape a webpage graph?

how to scrape a complicated page?



how to scrape a webpage graph?

how to scrape a complicated page?
 → requests module (220)
 → **selenium** module (320)



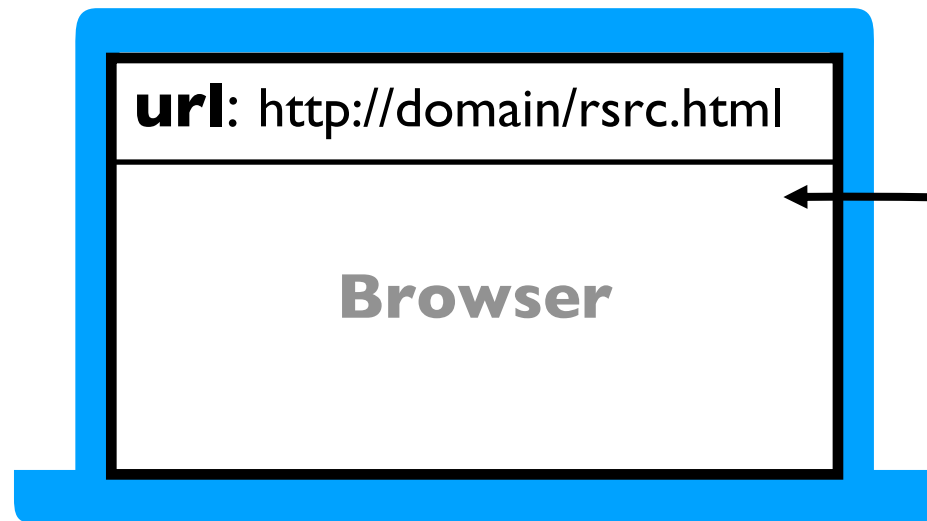
Document Object Model:

Every Webpage is a Tree

What does a web browser do when it gets some **HTML** in an **HTTP** response?

(hyper-text
markup language)

(hyper-text
transfer protocol)



HTTP Response

HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74

```
<html>
  <head><script>...</script></head>
  <body>
    <h1>Welcome</h1>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </body>
</html>
```

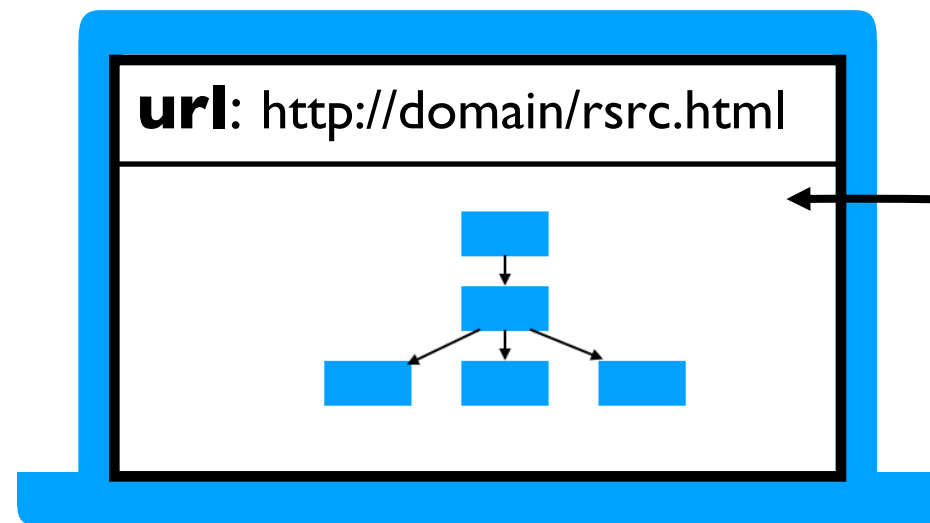
url: http://domain/rsrc.html

```
<html>
  <body>
    <h1>Welcome</h1>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </body>
</html>
```

HTTP Response

HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74

```
<html>
  <head><script>...</script></head>
  <body>
    <h1>Welcome</h1>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </body>
</html>
```



HTTP Response

HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74

```
<html>  
  <head><script>...</script></head>  
  <body>  
    <h1>Welcome</h1>  
    <a href="about.html">About</a>  
    <a href="contact.html">Contact</a>  
  </body>  
</html>
```

before displaying a page, the browser
uses HTML to generate a
Document Object Model (DOM Tree)

url: http://domain/rsrc.html

HTTP Response

HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74

```
<html>  
  <head><script>...</script></head>  
  <body>  
    <h1>Welcome</h1>  
    <a href="about.html">About</a>  
    <a href="contact.html">Contact</a>  
  </body>  
</html>
```

html

body

h1

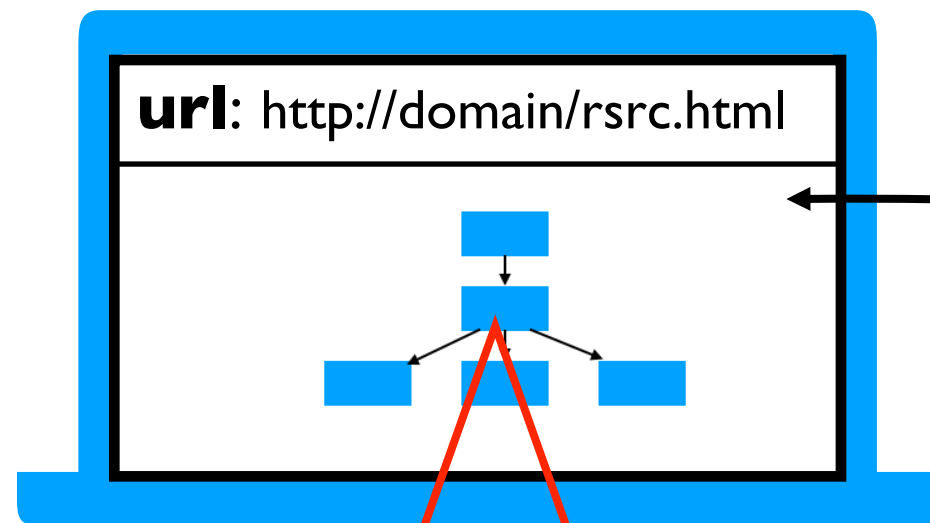
a

a

vocab: elements

Elements may contain

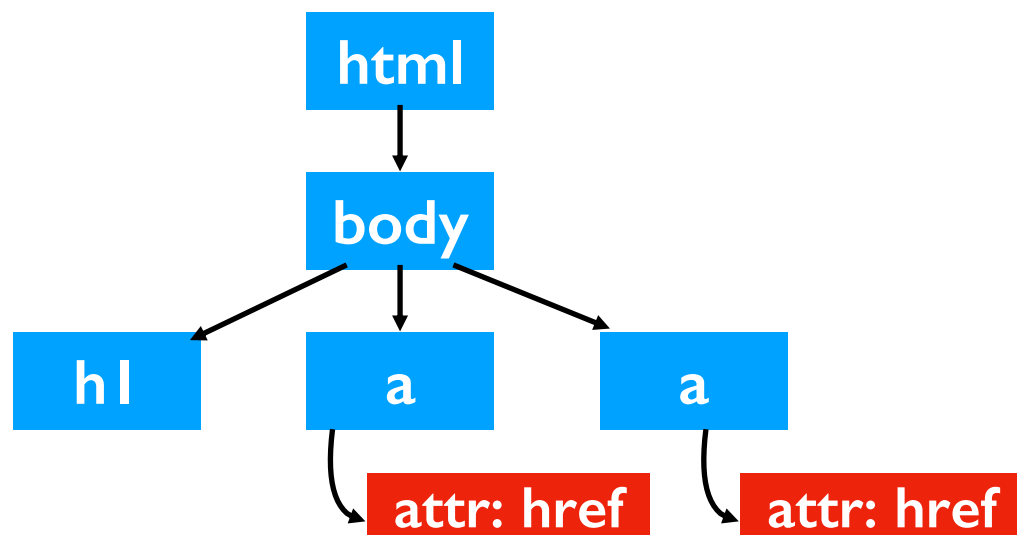
- attributes



HTTP Response

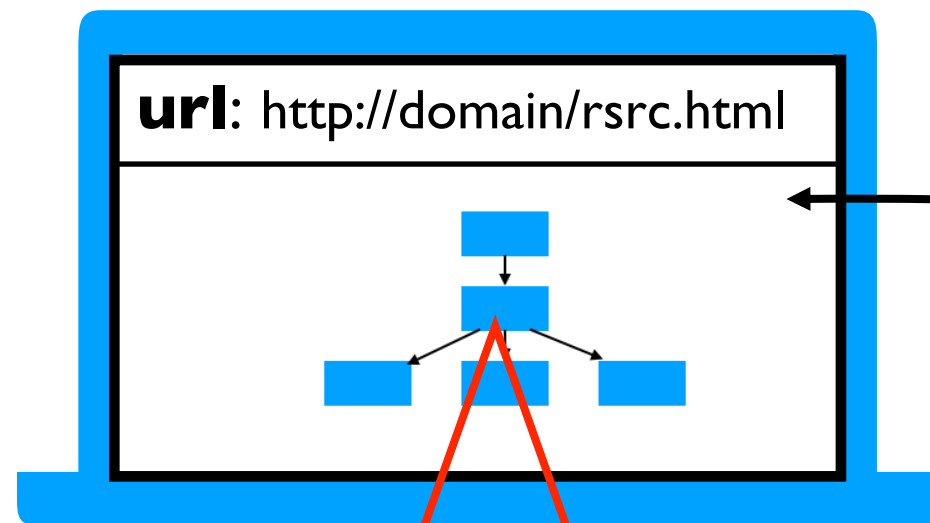
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74

```
<html>  
  <head><script>...</script></head>  
  <body>  
    <h1>Welcome</h1>  
    <a href="about.html">About</a>  
    <a href="contact.html">Contact</a>  
  </body>  
</html>
```



Elements may contain

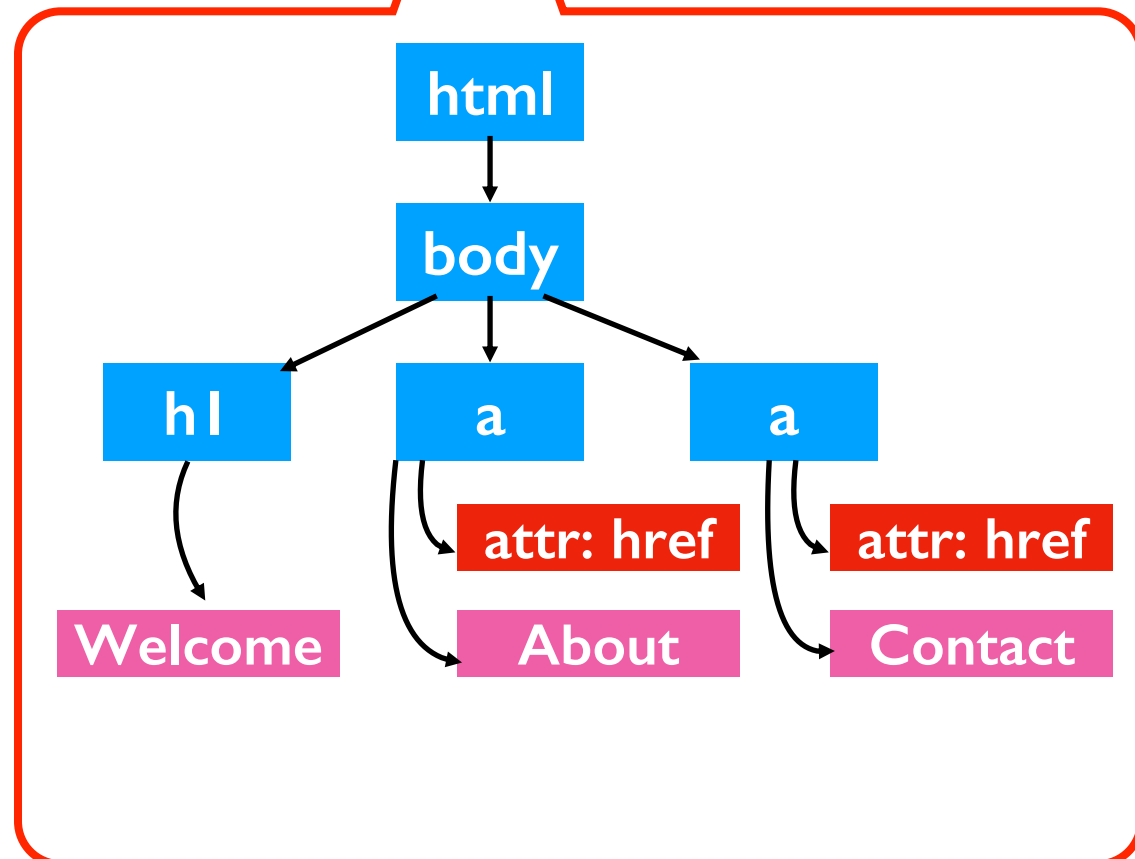
- attributes
- text



HTTP Response

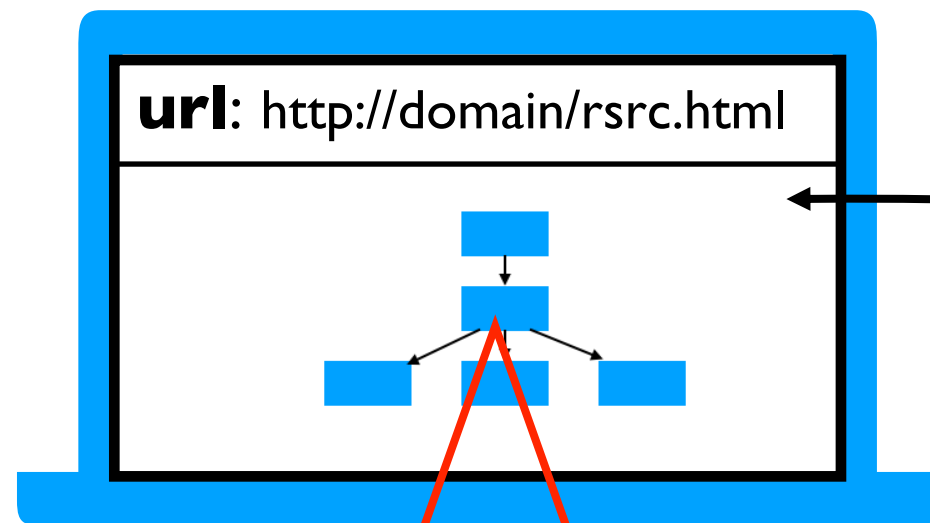
```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74
```

```
<html>
  <head><script>...</script></head>
  <body>
    <h1>Welcome</h1>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </body>
</html>
```



Elements may contain

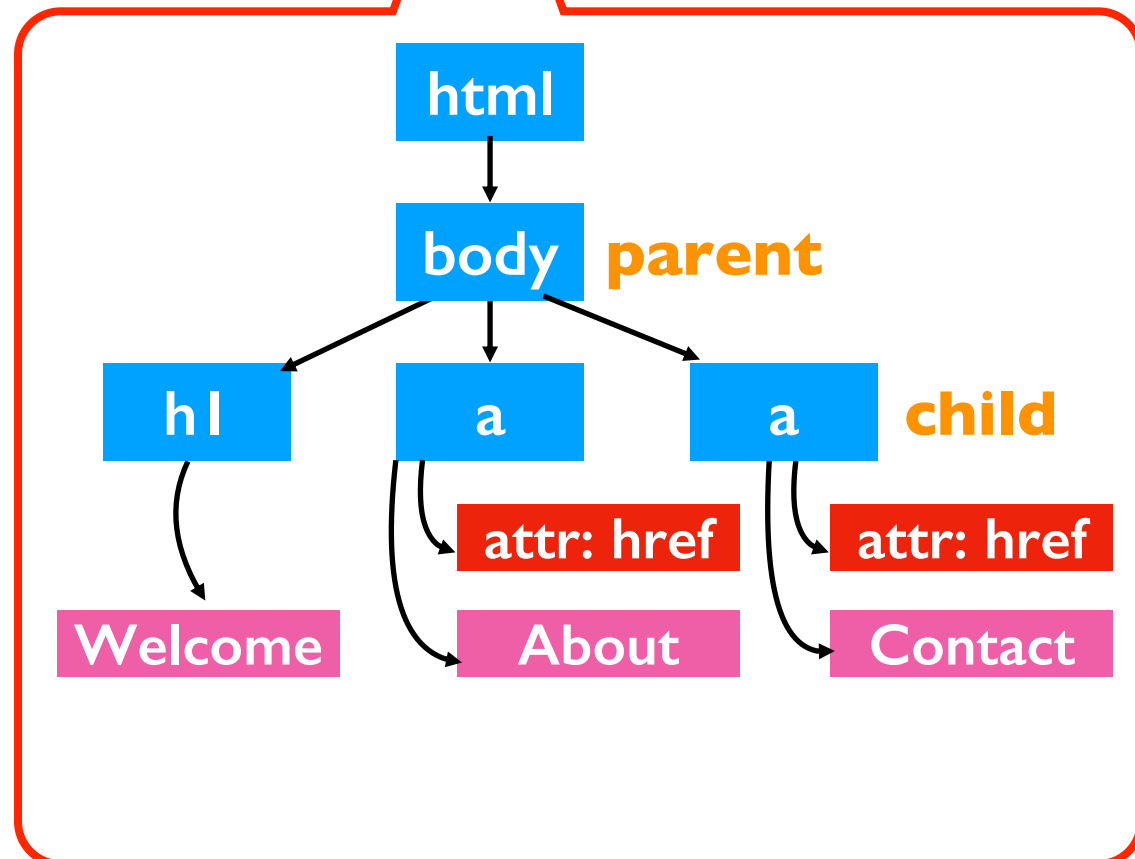
- attributes
- text
- other elements



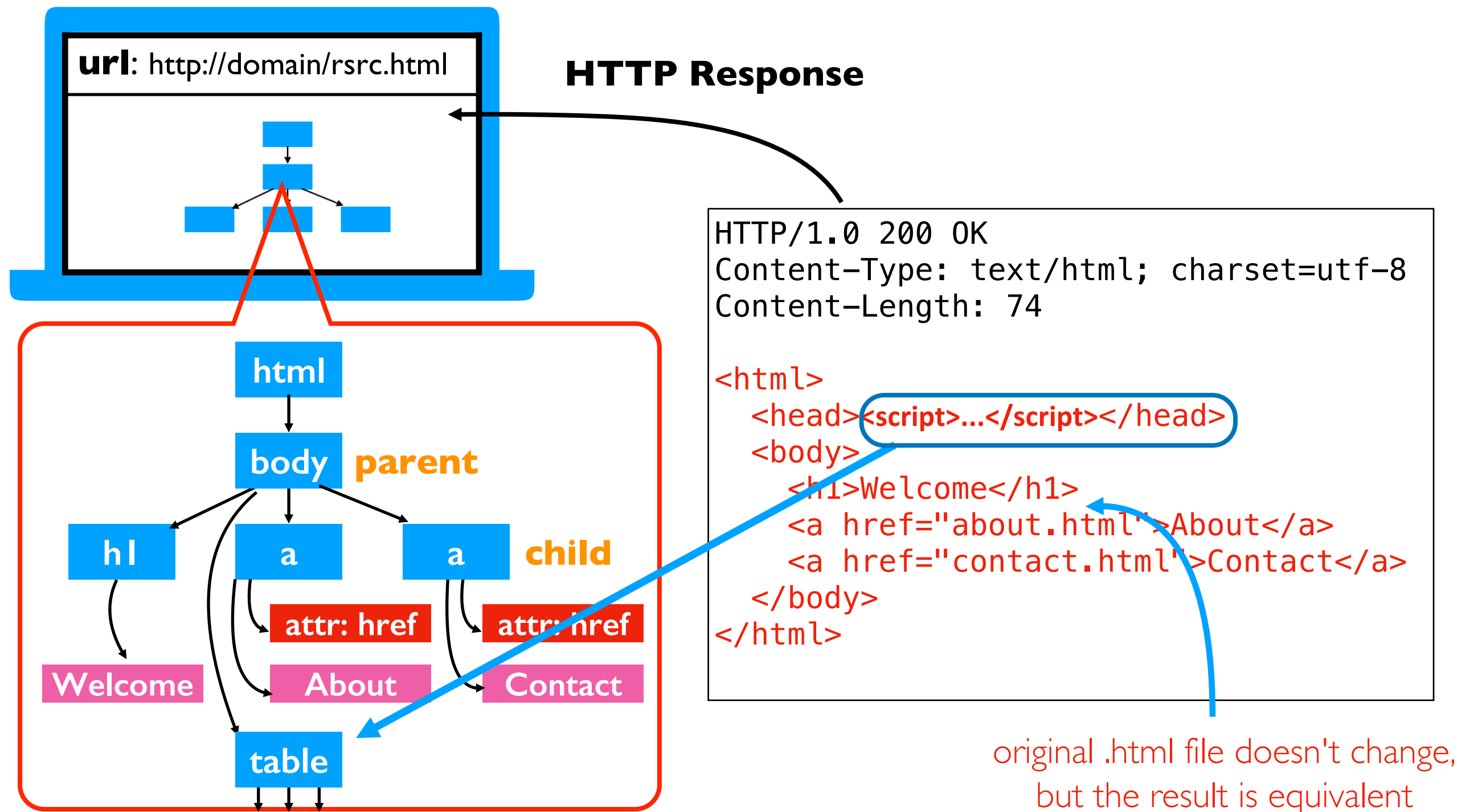
HTTP Response

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74
```

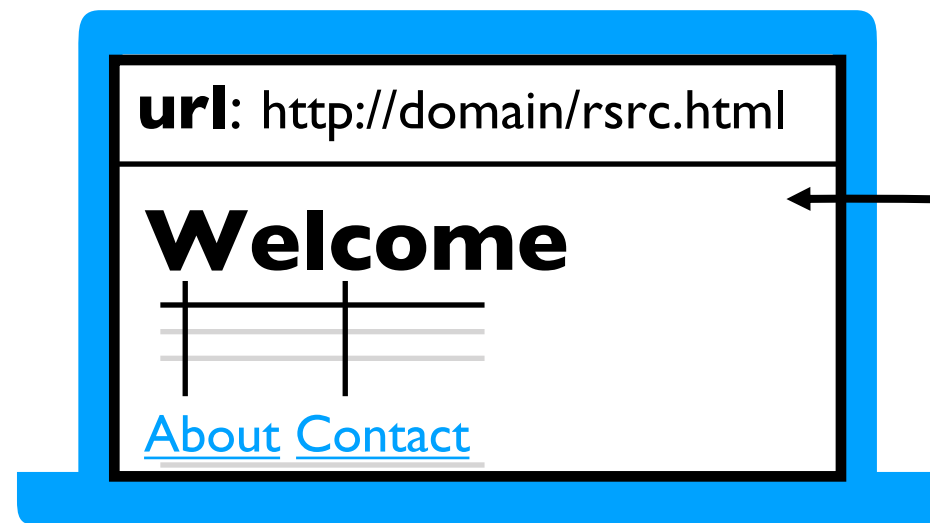
```
<html>
  <head><script>...</script></head>
  <body>
    <h1>Welcome</h1>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </body>
</html>
```



JavaScript (if there's an engine to execute it) may directly edit the DOM!



we need a JavaScript engine so we
can scrape the generated table



HTTP Response

HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 74

```
<html>
  <head><script>...</script></head>
  <body>
    <h1>Welcome</h1>
    <a href="about.html">About</a>
    <a href="contact.html">Contact</a>
  </body>
</html>
```

browser renders (displays) the
DOM tree, based on original file
and any JavaScript changes

Web Scraping: Simple and Complicated

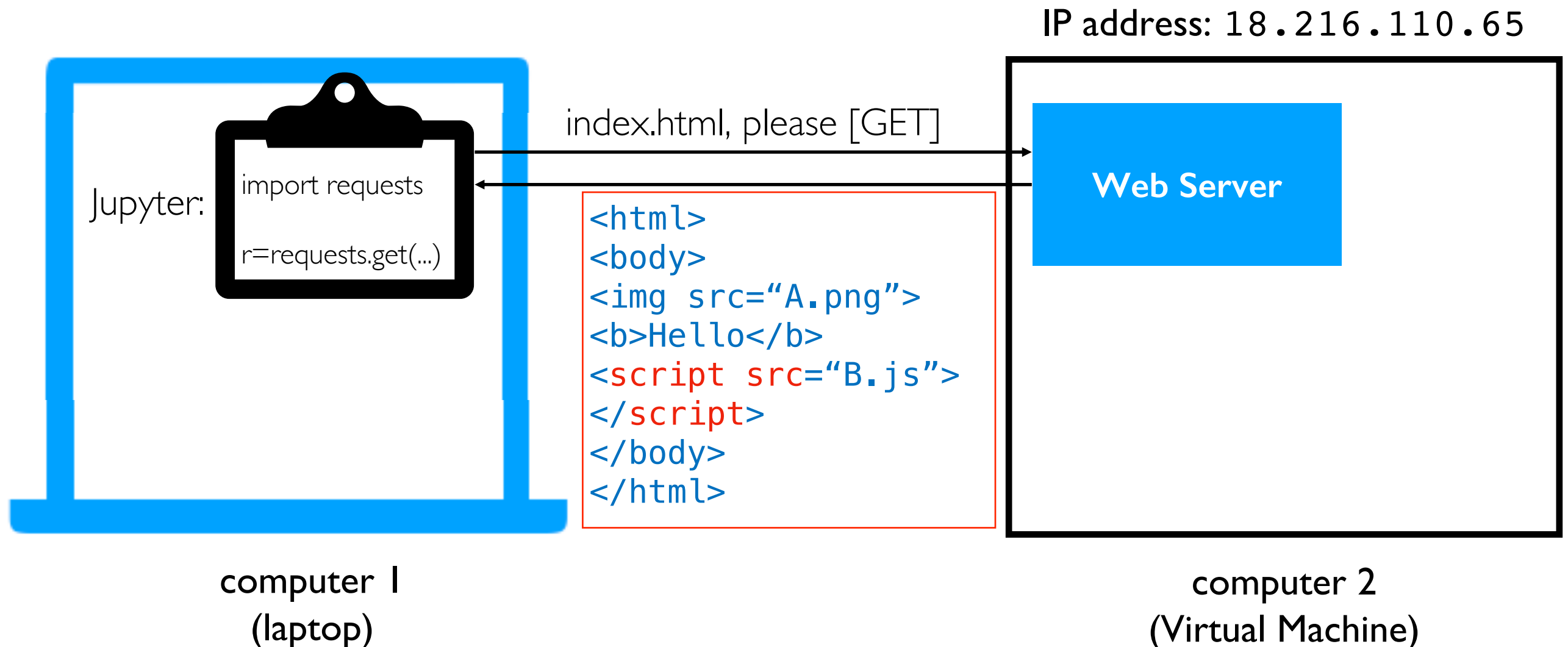
requests vs. Selenium

requests module (FAST!)

- **can** fetch .html, .js, .etc file

Selenium

- **can** fetch .html, .js, .etc file
- **can** run a .js file in browser
- **can** grab HTML version of DOM after JavaScript has modified it



requests vs. Selenium

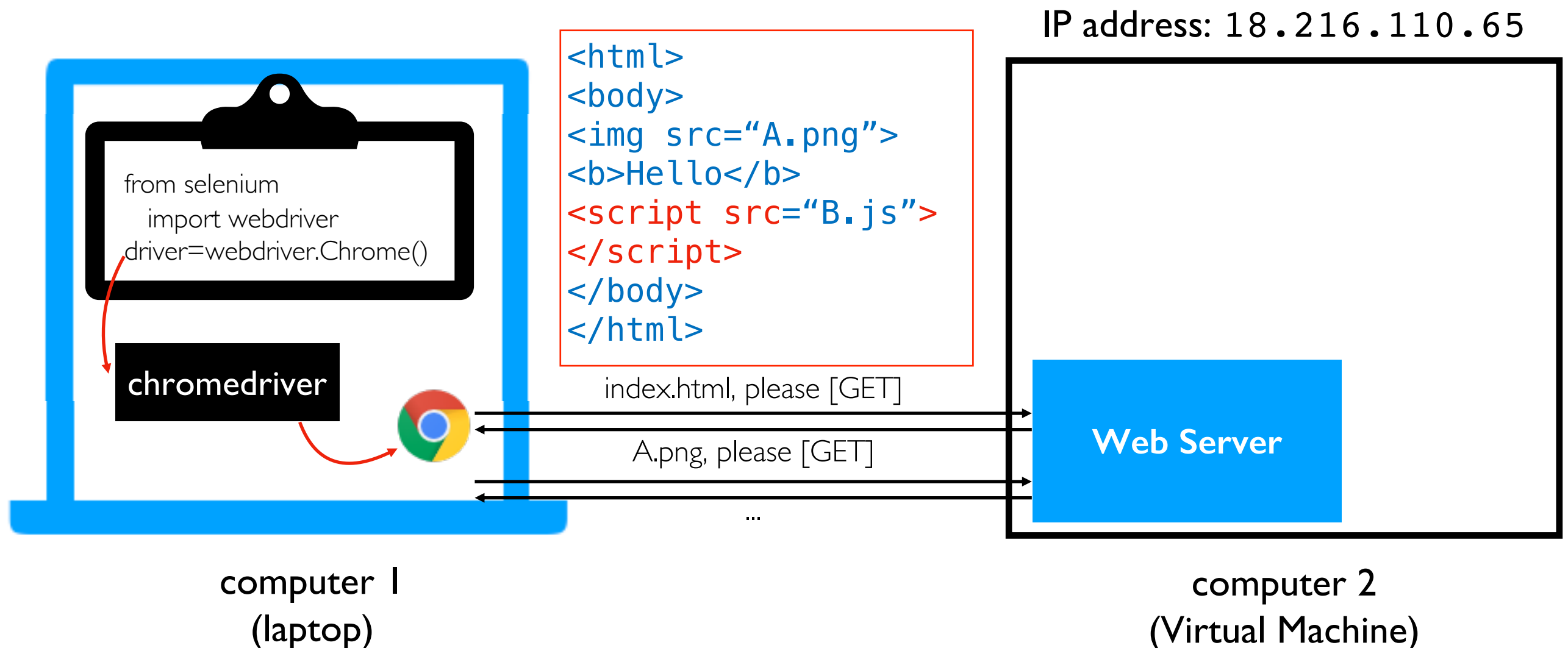
requests module (FAST!)

- **can** fetch .html, .js, .etc file

Selenium

- **can** fetch .html, .js, .etc file
- **can** run a .js file in browser
- **can** grab HTML version of DOM after JavaScript has modified it

note: Selenium is most commonly used for testing websites, but it works great for tricky scraping too

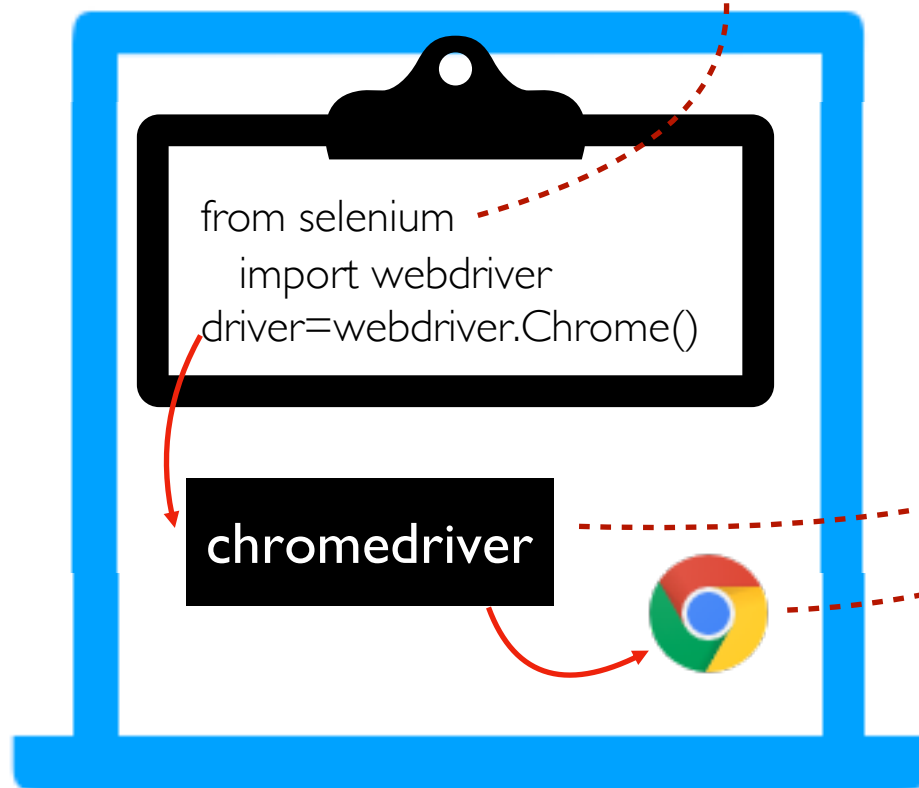


Installing: Selenium, Chrome, Driver

Selenium Install (Ubuntu 20.04)

<https://github.com/msyamkumar/cs320-s23-projects/tree/main/p3#part-3-web-crawling-websearcher>

pip3 install selenium



computer I
(laptop)

on some systems, chromedriver is installed separately

<https://chromedriver.chromium.org/downloads>

Current Releases

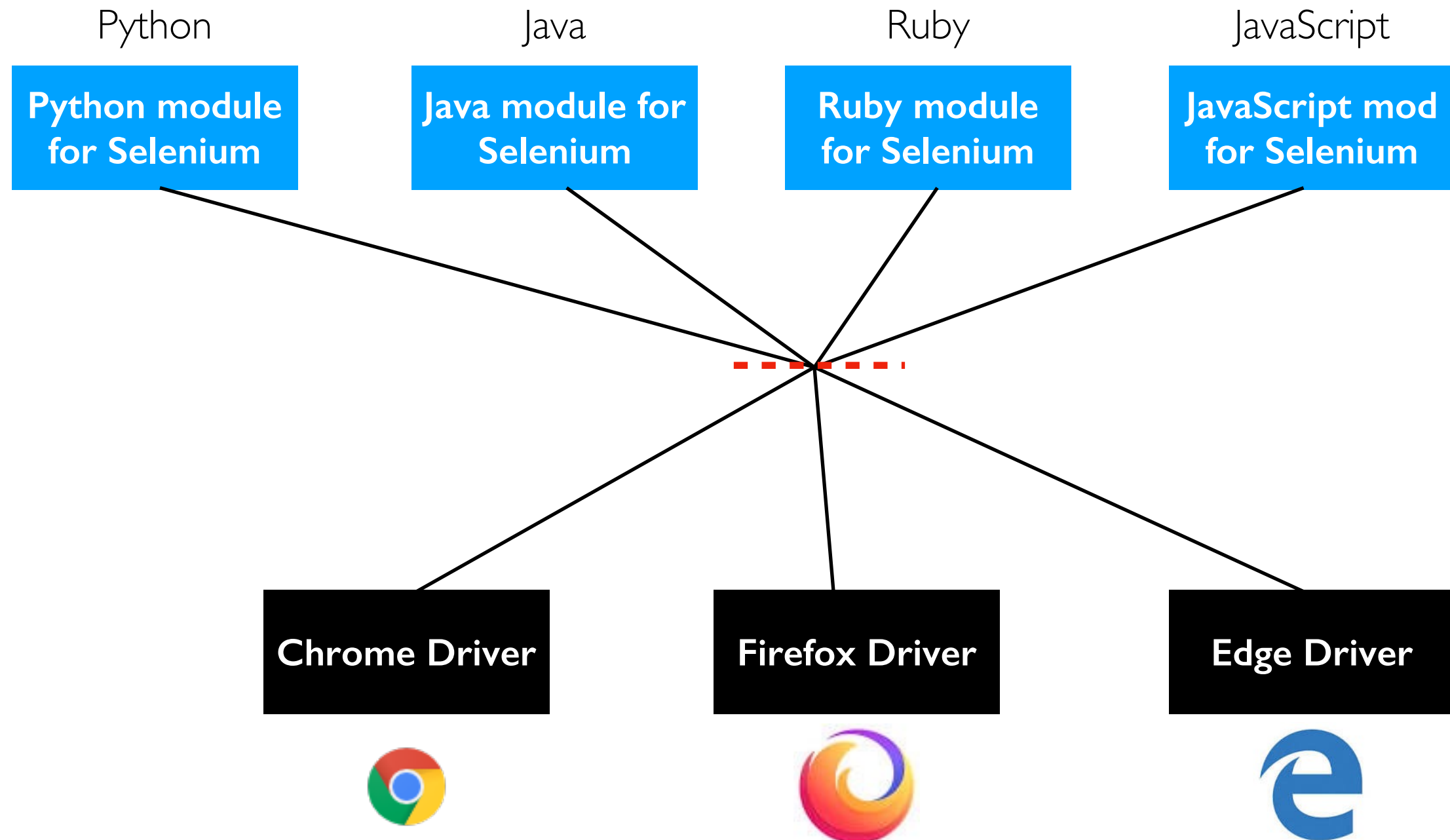
- If you are using Chrome version 86, please download [ChromeDriver 86.0.4240.22](#)
- If you are using Chrome version 85, please download [ChromeDriver 85.0.4183.87](#)
- If you are using Chrome version 84, please download [ChromeDriver 84.0.4147.30](#)
- For older version of Chrome, please see below for the version of ChromeDriver that supports it.

sudo apt -y install chromium-browser

Check...

```
trh@instance-1:~$ chromium-browser --version
/usr/bin/chromium-browser: 12: xdg-settings: not found
Chromium 94.0.4606.81 snap
trh@instance-1:~$ chromium.chromedriver --version
ChromeDriver 94.0.4606.81 (5a03c5f1033171d5ee1671d219a...
```

Why Drivers?



Examples

Example 1a: Late Loading Table (page1.html)

Welcome

Here's a table

A	B	C
1	2	3
4	5	6

And another one...

x	y
0	1
2	3
4	5
6	7
8	9
10	11
12	13
14	15
16	17
18	19

← added after 1 second

Example 1b: Headless Mode and Screenshots

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.common.exceptions import NoSuchElementException

options = Options()
options.headless = True
b = webdriver.Chrome(options=options)

b.get('????')

from IPython.core.display import Image
b.save_screenshot("out.png")
Image("out.png")

b.close()
```

Example 2: Auto-Clicking Buttons

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.common.exceptions import NoSuchElementException
```

```
options = Options()
options.headless = True
b = webdriver.Chrome(options=options)
```

```
b.get('????')
```

```
btn = b.find_element_by_id("BTN_ID")
btn.click()
```

```
b.close()
```

Keep clicking...

name	formed	dissipated	mph	damage	deaths
Baker	08/18/1950	09/01/1950	105	2.55M	38
Camille	08/14/1969	08/22/1969	175	1.42B	259
Eloise	09/13/1975	09/24/1975	125	560M	80
Frederic	08/29/1979	09/15/1979	130	1.77B	12
Elena	08/28/1985	09/04/1985	125	1.3B	9
Opal	09/27/1995	10/06/1995	150	4.7B	63
Danny	07/16/1997	07/27/1997	80	100M	4
Ivan	09/02/2004	09/25/2004	165	26.1B	92
Dennis	07/04/2005	07/18/2005	150	3.98B	76

Show More!

auto click



Example 3: Entering Passwords

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.common.exceptions import NoSuchElementException

options = Options()
options.headless = True
b = webdriver.Chrome(options=options)

b.get('????')

pw = b.find_element_by_id("pw")
pw.send_keys("fido")

b.close()
```

Sign In to View Table

Password:

Table...

name	formed	dissipated	mph	damage	deaths
Baker	08/18/1950	09/01/1950	105	2.55M	38
Camille	08/14/1969	08/22/1969	175	1.42B	259
Eloise	09/13/1975	09/24/1975	125	560M	80
Frederic	08/29/1979	09/15/1979	130	1.77B	12

Example 4: Many Queries

Give Me a Year

Year:

Table...

name	formed	dissipated	mph	damage	deaths
Baker	08/18/1950	09/01/1950	105	2.55M	38
Easy	09/01/1950	09/09/1950	125	3.3M	2
King	10/13/1950	10/20/1950	130	32M	11

