Extracting data from the web

APIs and beyond

Scott Chamberlain
Karthik Ram
Garrett Grolemund

June 2016

USER 2016 CONFERENCE

Outline

Part 1 - Collecting data from an API

Part 2 - Wrapping an API with R

Part 3 - Scraping data without an API

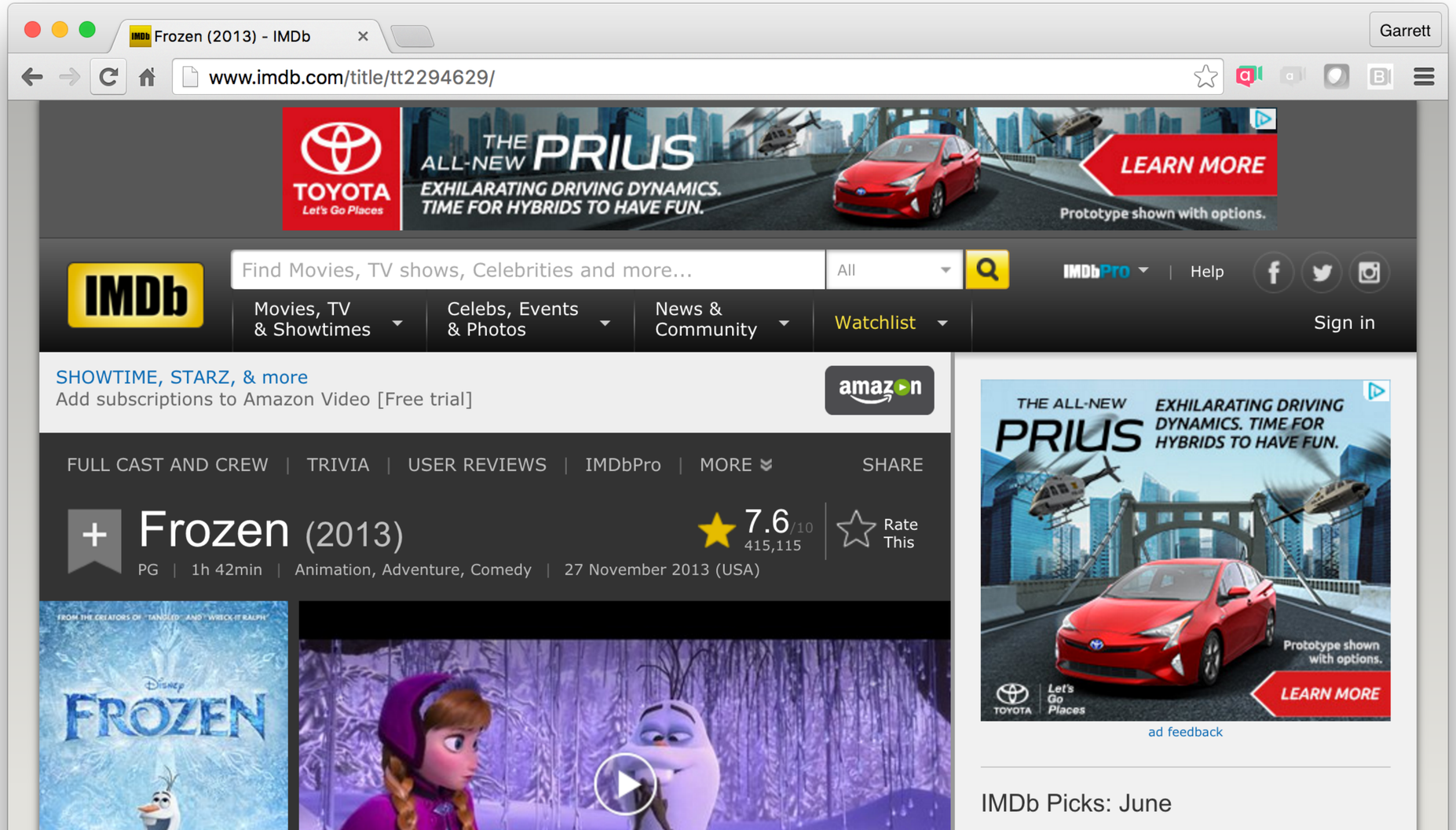
Garrett Grolemund
Master Instructor and
Data Scientist
RStudio®

Web Scraping


HELLO
my name is
Garrett

What if data is on a web page but there is no API?

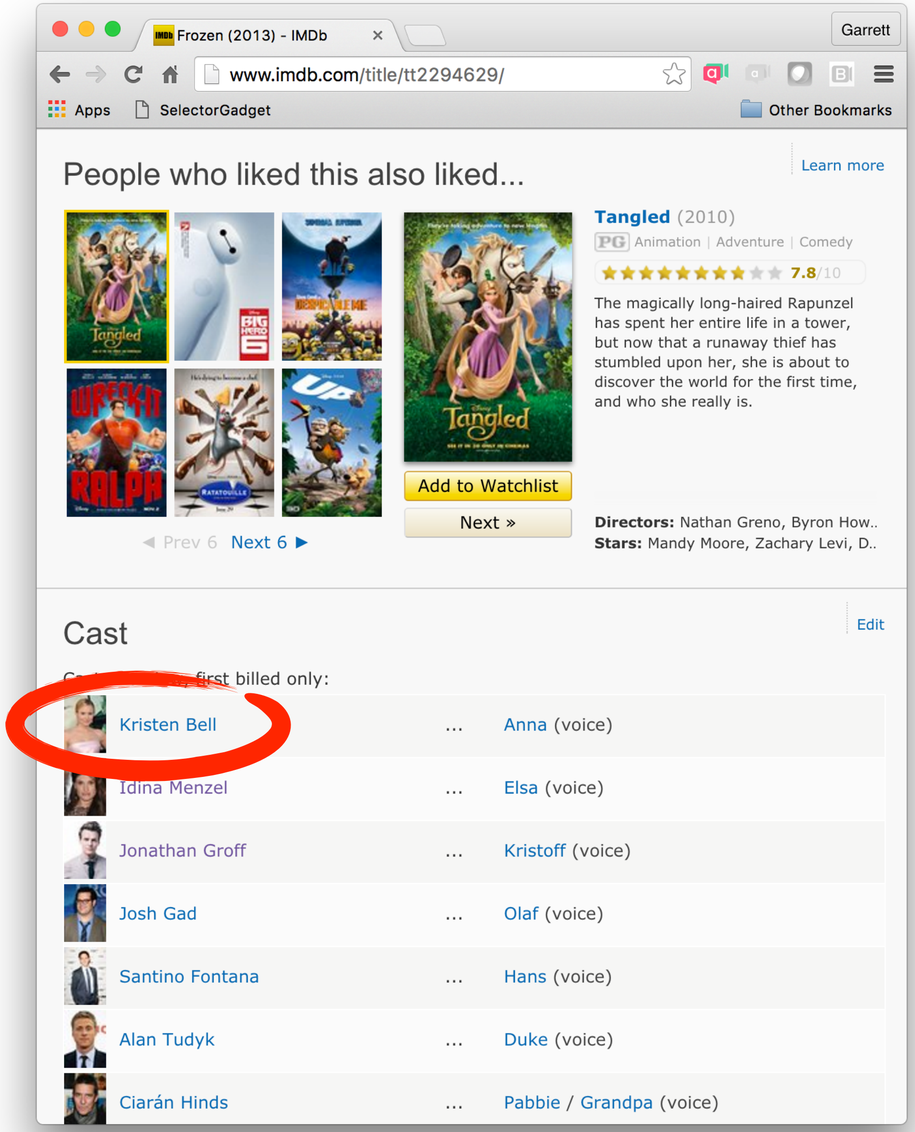
www.imdb.com/title/tt2294629/

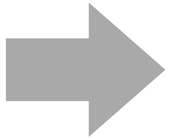


Outline

1. **How to scrape** a web page
2. **rvest**
3. **selectorGadget**
4. Practice with **tables**

Strategy


Frozen (2013) - IMDb
Garrett
www.imdb.com/title/tt2294629/
Apps
SelectorGadget
Other Bookmarks
People who liked this also liked...
Learn more
Tangled (2010)
PG Animation | Adventure | Comedy
7.8/10
The magically long-haired Rapunzel has spent her entire life in a tower, but now that a runaway thief has stumbled upon her, she is about to discover the world for the first time, and who she really is.
Add to Watchlist
Next »
Directors: Nathan Greno, Byron How..
Stars: Mandy Moore, Zachary Levi, D..
◄ Prev 6 Next 6 ►
Cast
Edit
first billed only:
Kristen Bell ... Anna (voice)
Idina Menzel ... Elsa (voice)
Jonathan Groff ... Kristoff (voice)
Josh Gad ... Olaf (voice)
Santino Fontana ... Hans (voice)
Alan Tudyk ... Duke (voice)
Ciarán Hinds ... Pabbie / Grandpa (voice)

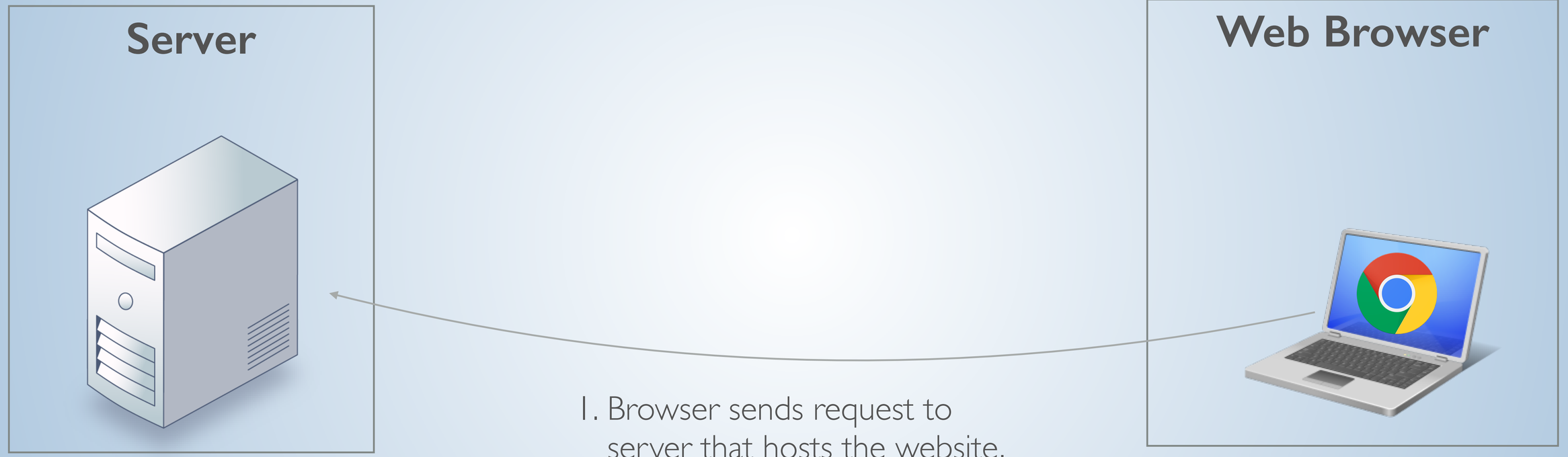



Frozen (2013) - IMDb
view-source:www.imdb.com
Garrett
view-source:www.imdb.com/title/tt2294629/
Apps
SelectorGadget
Other Bookmarks
<table class="cast_list">
<tr><td colspan="4" class="castlist_label">Cast overview, first billed only:</td></tr>
<tr class="odd">
<td class="primary_photo">
<a href="/name/nm0068338/?ref_=tt_cl_i1"
><img height="44" width="32" alt="Kristen Bell"
title="Kristen Bell"src="http://ia.media-imdb.com/images/G/01/imdb/images/nopicture/32x44/name-2138558783._CB271193362_.png"class="loadlate hidden " loadlate="http://ia.media-imdb.com/images/M/MV5BMjE2MjkzOTIxMV5BMl5BanBnXkFtZTgwNzM3MTkzMTE@._V1_UX32_CR0,0,32,44_AL_.jpg"
/>
</td>
<td class="itemprop" itemprop="actor" itemscope itemtype="http://schema.org/Person">
<a href="/name/nm0068338/?ref_=tt_cl_t1"
itemprop='url'><span class="itemprop"
itemprop="name">Kristen Bell

</td>
<td class="ellipsis">
...
</td>
<td class="character">

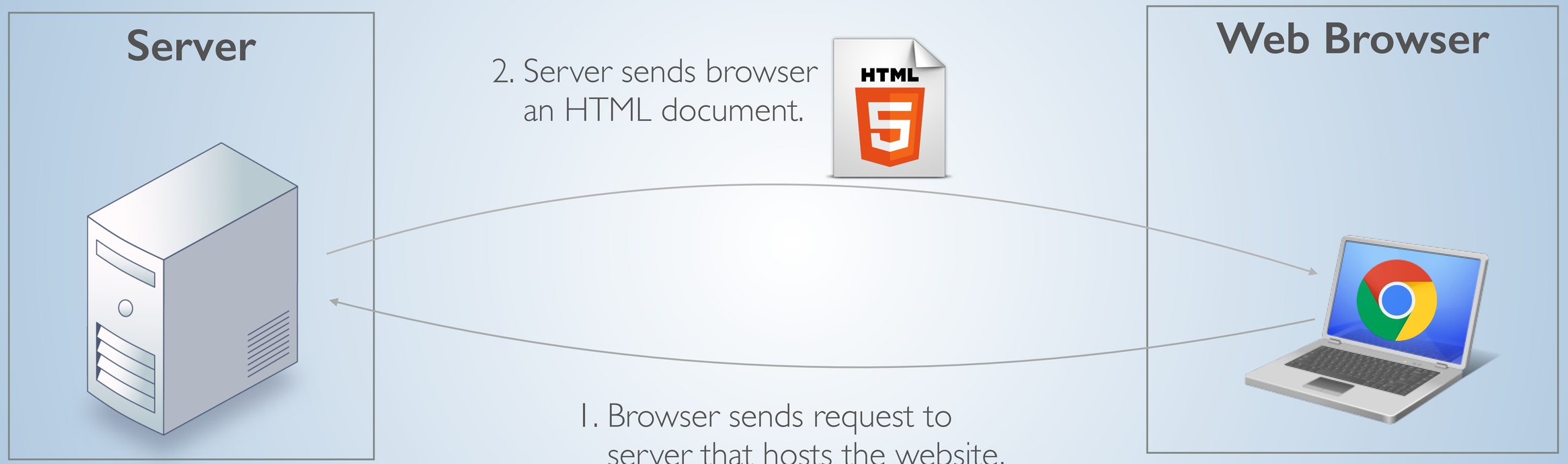
Anna
(voice)

HTML (Review)

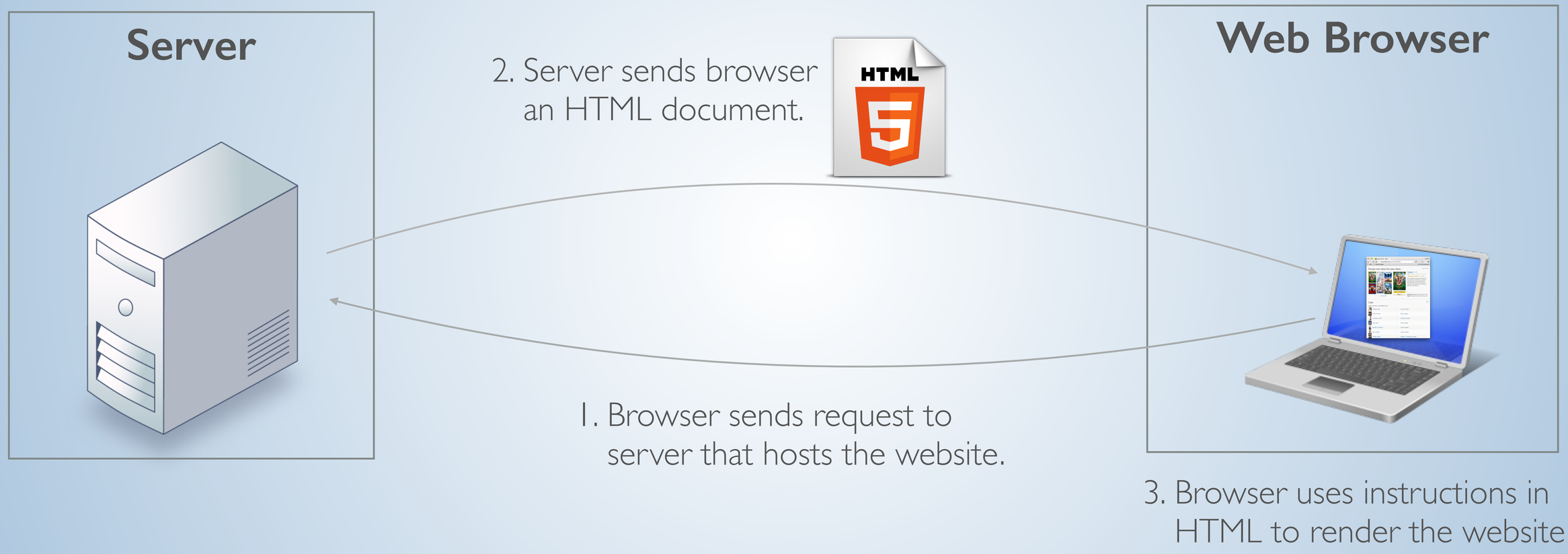


1. Browser sends request to server that hosts the website.

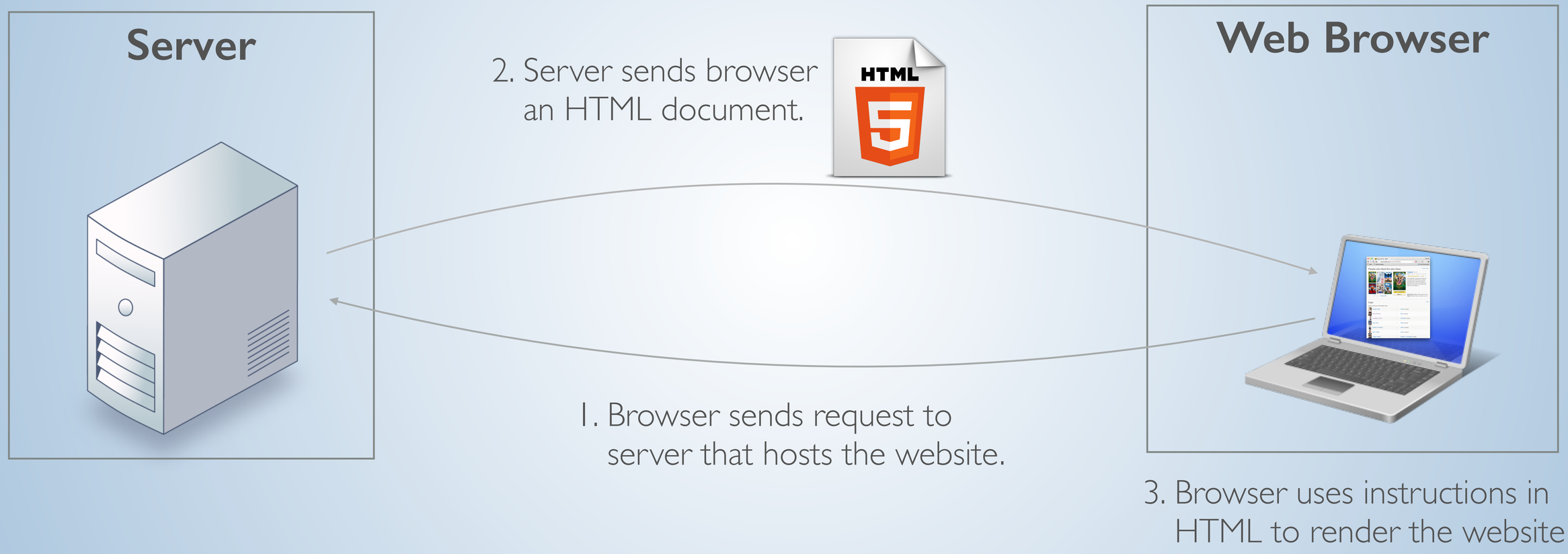
HTML (Review)


Server
2. Server sends browser an HTML document.
HTML
Web Browser
1. Browser sends request to server that hosts the website.

HTML (Review)

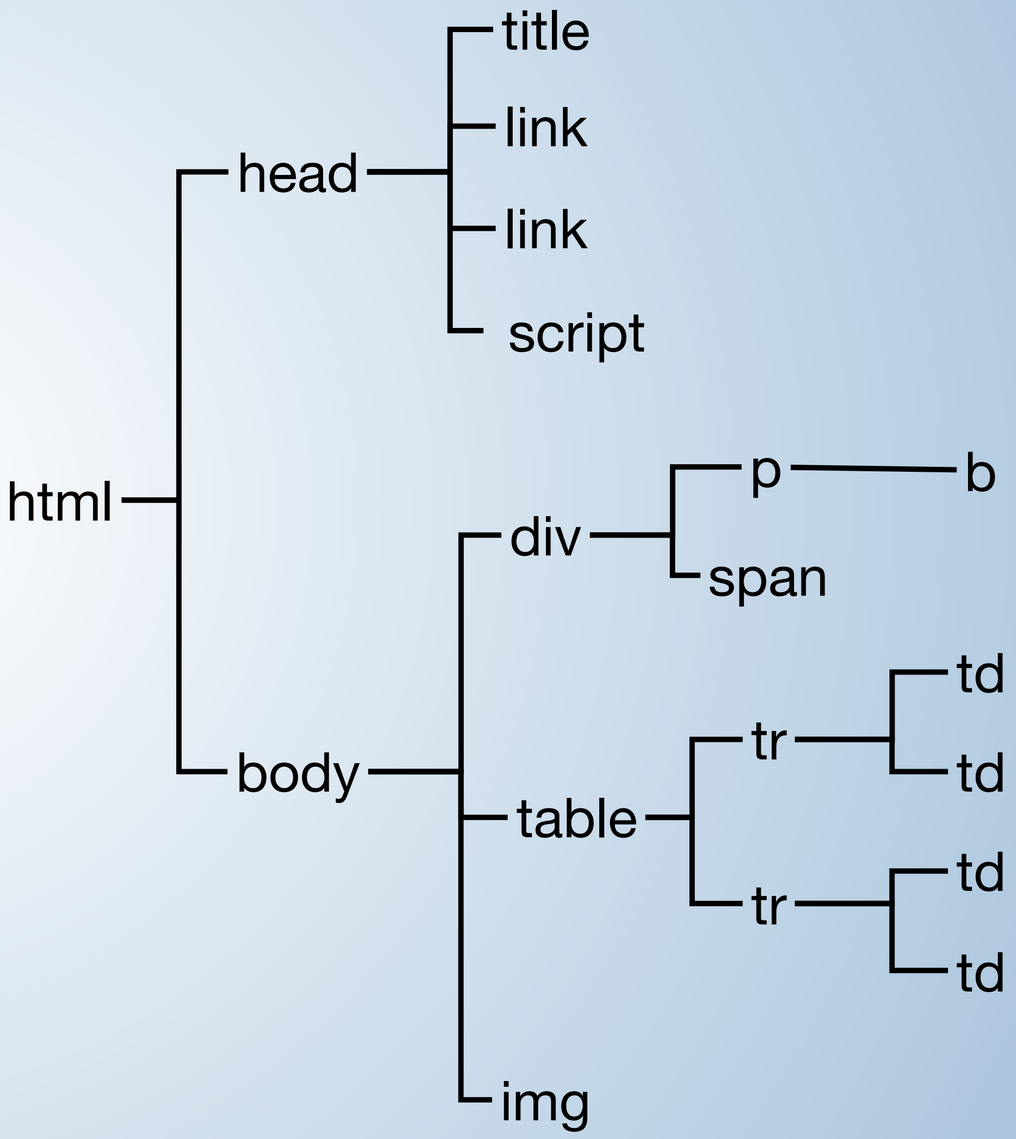

Server
2. Server sends browser an HTML document.
HTML
Web Browser
1. Browser sends request to server that hosts the website.
3. Browser uses instructions in HTML to render the website

HTML (Review)


Server
2. Server sends browser an HTML document.
HTML
Web Browser
1. Browser sends request to server that hosts the website.
3. Browser uses instructions in HTML to render the website

HTML (Review)

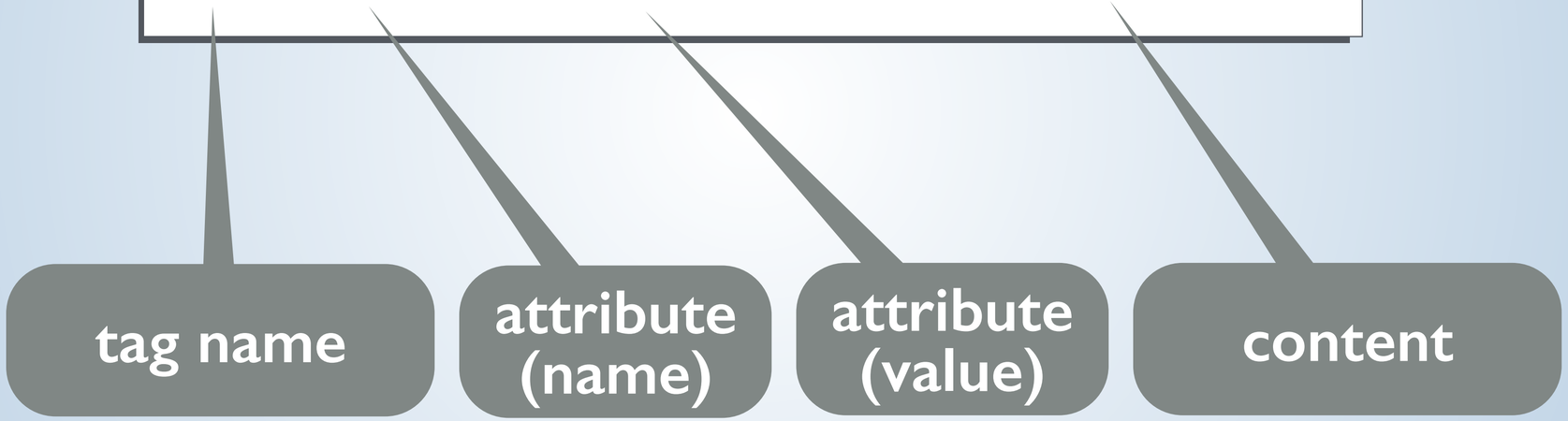

```
<html>
  <head>
    <title>Title</title>
    <link rel="icon" type="icon" href="http://a" />
    <link rel="icon" type="icon" href="http://b" />
    <script type="text/javascript">
      var ue_t0=window.ue_t0||+new Date();
    </script>
  </head>
  <body>
    <div>
      <p>Click <b>here</b> now.</p>
      <span>Frozen</span>
    </div>
    <table style="width:100%">
      <tr>
        <td>Kristen</td>
        <td>Bell</td>
      </tr>
      <tr>
        <td>Idina</td>
        <td>Menzel</td>
      </tr>
    </table>
    <img src="http://ia.media-imdb.com/images.png"/>
  </body>
</html>
```



HTML (Review)

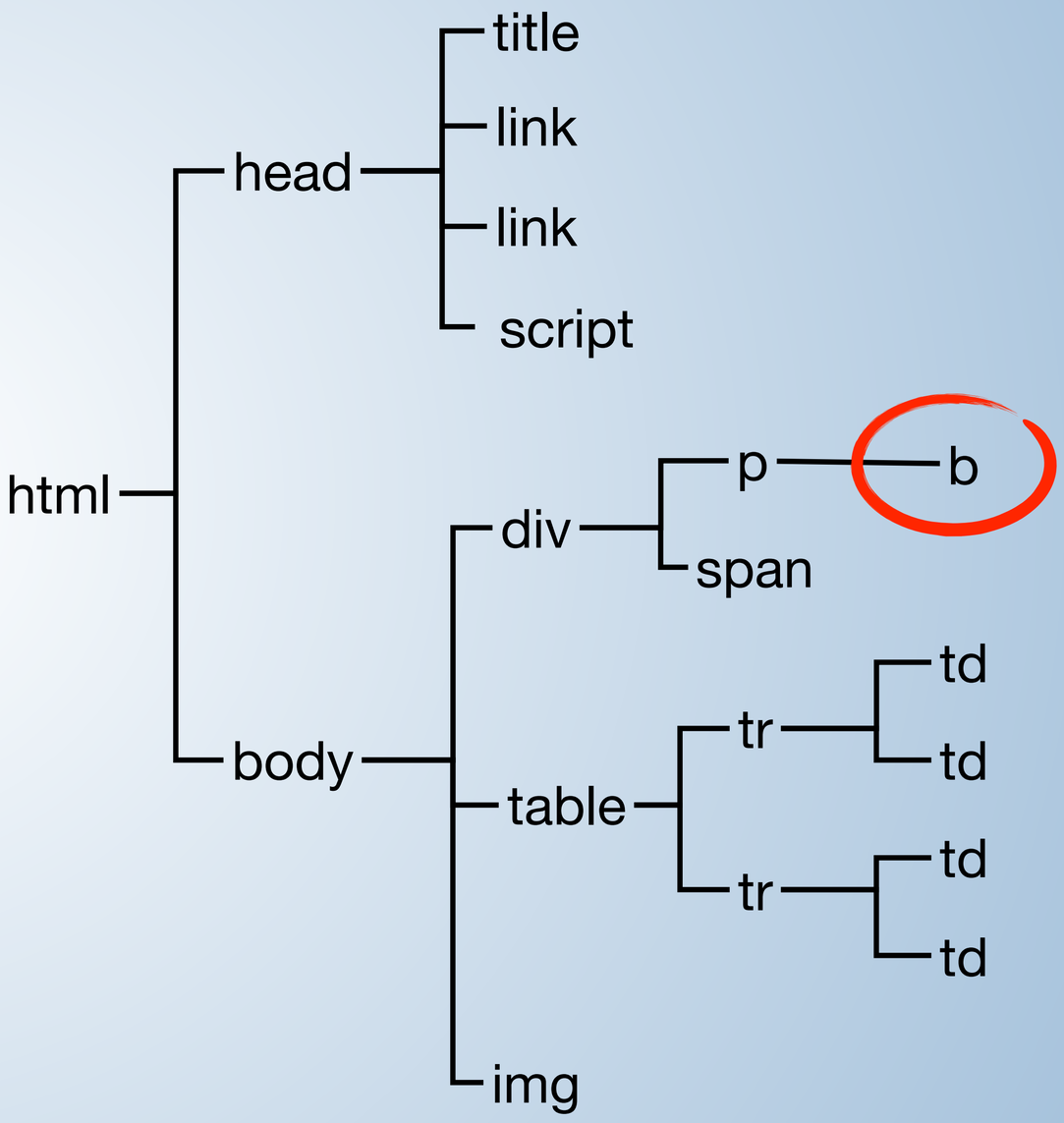
Each element in the page is created by a tag.

```
<a href="http://github.com">GitHub</a>
```



HTML (Review)


```
<html>
  <head>
    <title>Title</title>
    <link rel="icon" type="icon" href="http://a" />
    <link rel="icon" type="icon" href="http://b" />
    <script type="text/javascript">
      var ue_t0=window.ue_t0||+new Date();
    </script>
  </head>
  <body>
    <div>
      <p>Click <b>here</b> now.</p>
      <span>Frozen</span>
    </div>
    <table style="width:100%">
      <tr>
        <td>Kristen</td>
        <td>Bell</td>
      </tr>
      <tr>
        <td>Idina</td>
        <td>Menzel</td>
      </tr>
    </table>
    <img src="http://ia.media-imdb.com/images.png"/>
  </body>
</html>
```



Your Turn

Find the source code for the Frozen IMDB page

Goto:
1. View
   a. Developer
      i. View Source

1. Right click on page
   a. Click View Source

Goto:
1. Firefox
   a. Web Developer
      i. Page Source

Goto:
1. Safari
   a. Preferences
      i. Advanced
         a) Check "Show Develop menu in menu bar"
2. Develop
   a. Show Page Source

Your Turn

Navigate to the IMDB page for Frozen and open the source code.

Locate the piece of HTML that inserts Kristen Bell into the Cast section.

Which HTML tag surrounds her name?

Which HTML tag surrounds her name?

span

Which HTML tag surrounds her name?

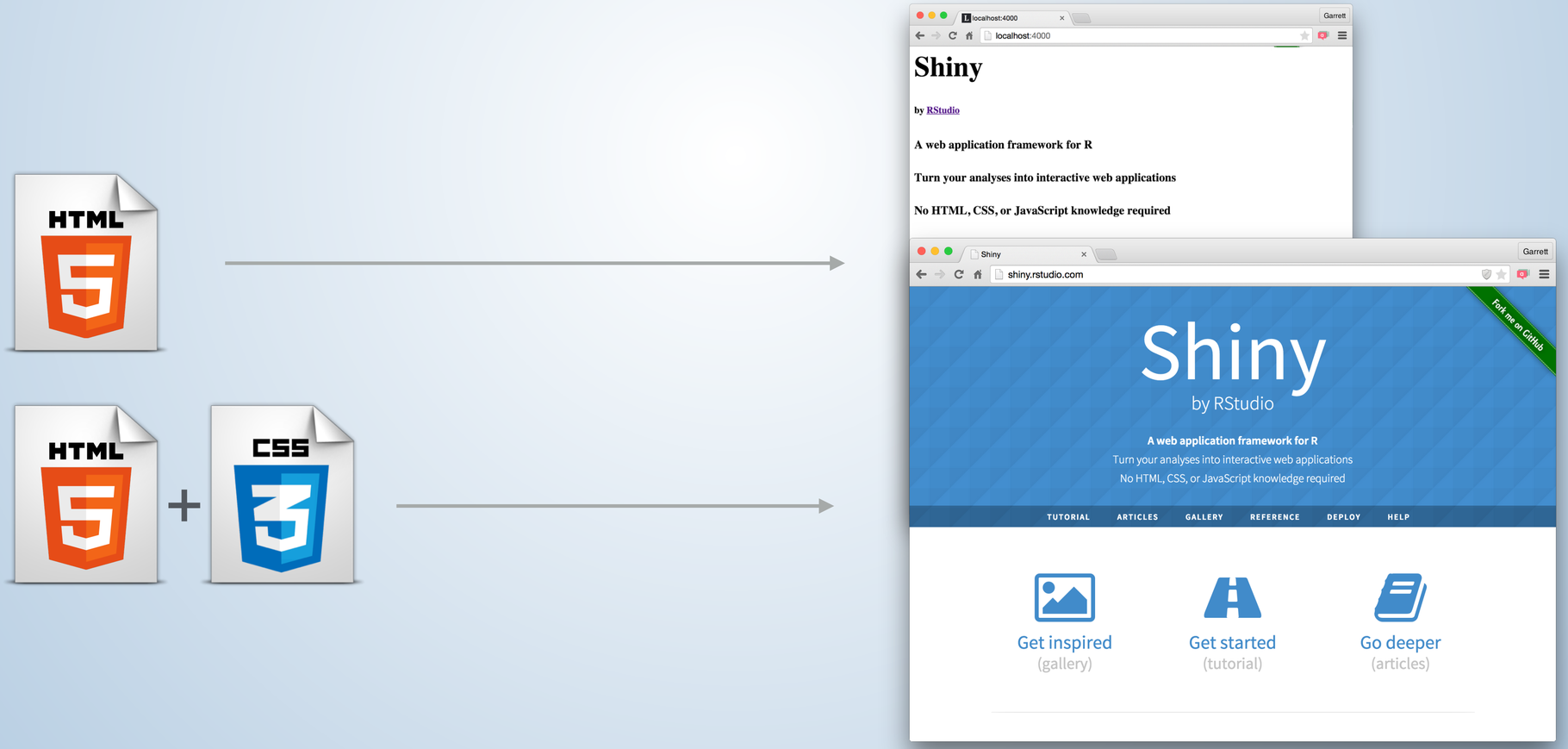
span

But how many tags are there?

css
selectors

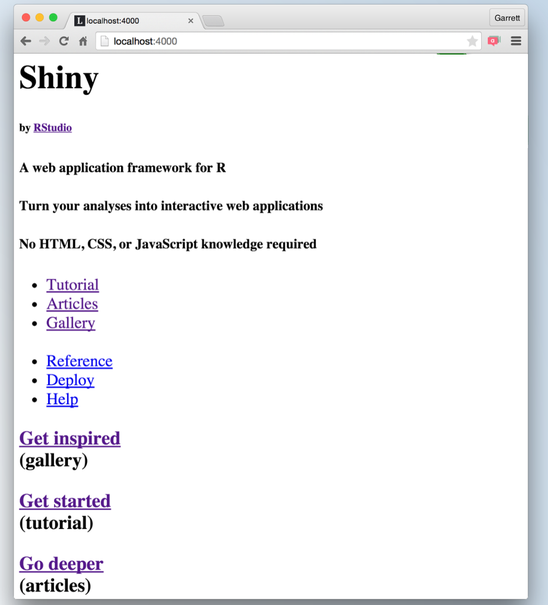
CSS (Review)

Cascading Style Sheets (CSS) are a framework for customizing the appearance of elements in a web page.

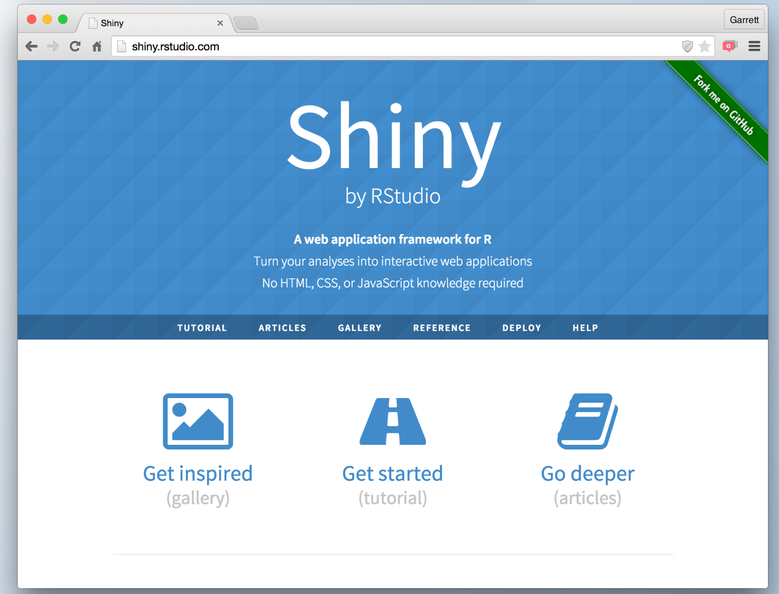


CSS (Review)


HTML


localhost:4000
Garrett
localhost:4000
Shiny
by RStudio
A web application framework for R
Turn your analyses into interactive web applications
No HTML, CSS, or JavaScript knowledge required
Tutorial
Articles
Gallery
Reference
Deploy
Help
Get inspired
(gallery)
Get started
(tutorial)
Go deeper
(articles)


HTML
CSS


Shiny
Garrett
shiny.rstudio.com
Fork me on GitHub
Shiny
by RStudio
A web application framework for R
Turn your analyses into interactive web applications
No HTML, CSS, or JavaScript knowledge required
TUTORIAL
ARTICLES
GALLERY
REFERENCE
DEPLOY
HELP
Get inspired
(gallery)
Get started
(tutorial)
Go deeper
(articles)

CSS (Review)


```
span {
  color: #ffffff;
}

.num {
  color: #a8660d;
}

table.data {
  width: auto;
}

#firstname {
  background-color: yellow;
}
```

selector
styling

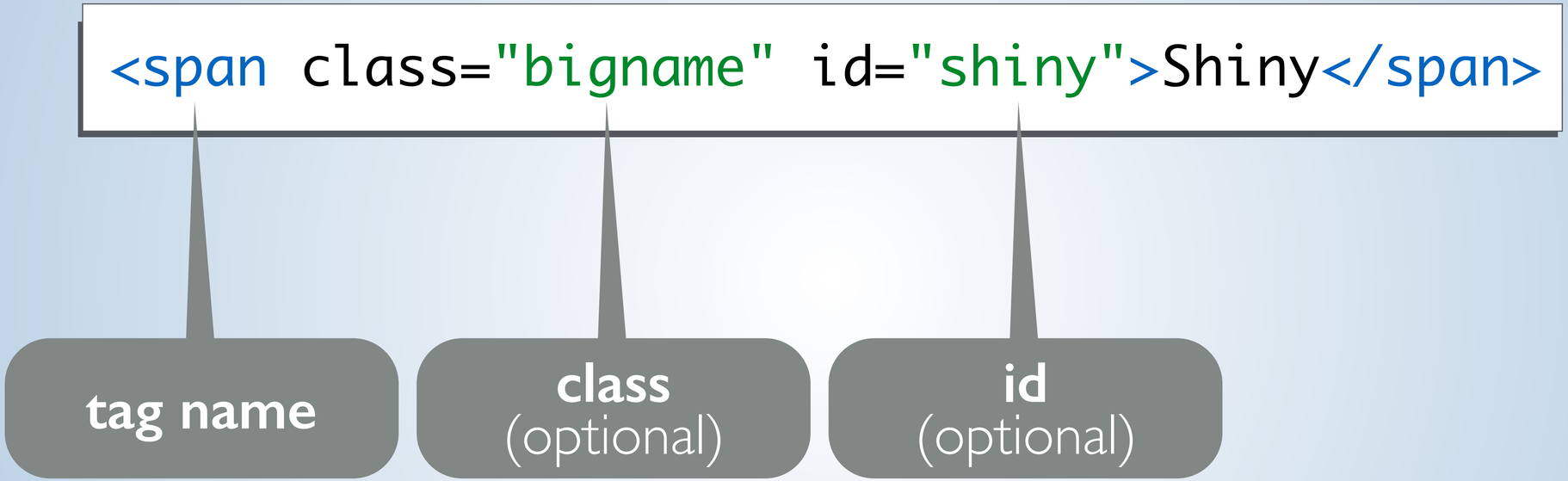
selector
styling

selector
styling

selector
styling

CSS (Review)

A CSS script describes an element by its tag, class, and/or ID.



CSS (Review)

A CSS script describes an element by its tag, class, and/or ID.

```
<span class="bigname" id="shiny">Shiny</span>
```

```
span
```

CSS selector for **ALL** elements with:

- the **span tag**

CSS (Review)

A CSS script describes an element by its tag, class, and/or ID.

```
<span class="bigname" id="shiny">Shiny</span>
```

```
.bigname
```

CSS selector for **ALL** elements with:

- the **bigname class**

CSS (Review)

A CSS script describes an element by its tag, class, and/or ID.

```
<span class="bigname" id="shiny">Shiny</span>
```

```
span.bigname
```

CSS selector for **ALL** elements with:

- the **span tag**

AND

- the **bigname class**

CSS (Review)

A CSS script describes an element by its tag, class, and/or ID.

```
<span class="bigname" id="shiny">Shiny</span>
```

```
#shiny
```

CSS selector for **ALL** elements with:

- the **shiny id**

CSS (Review)

Prefix	Matches
none	tag
.	class
#	id

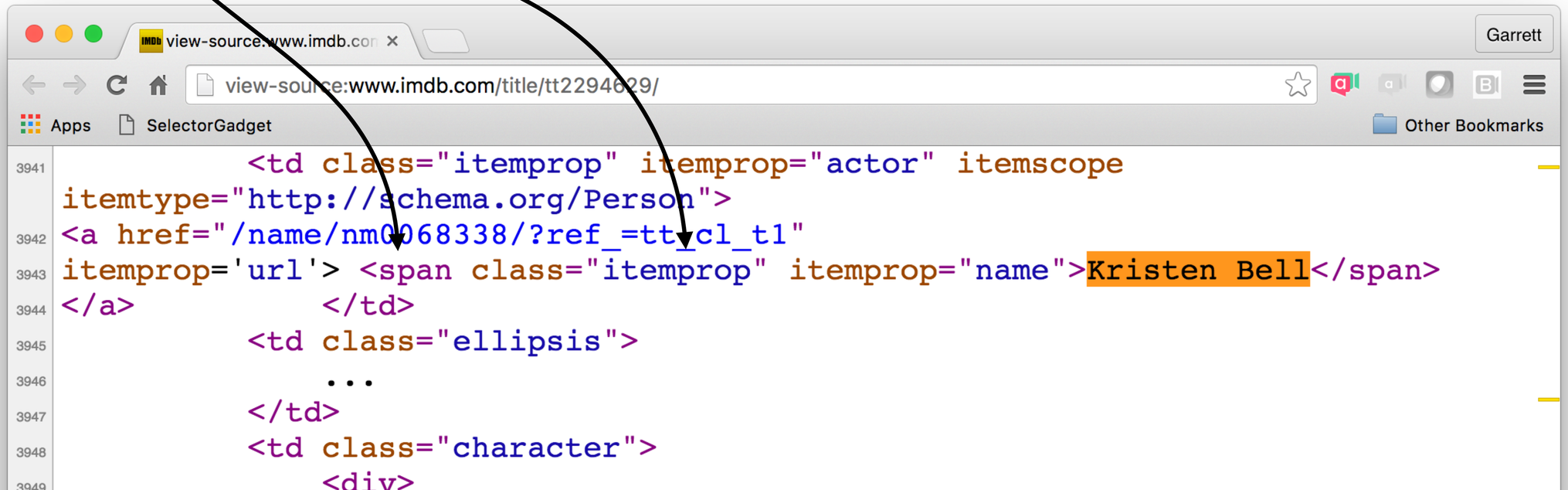
Your Turn

Which CSS identifiers are associated with Kristen Bell's name in the Frozen page? Write a CSS selector that targets them.

Which CSS identifiers are associated with Kristen Bell's name?

span (the element)

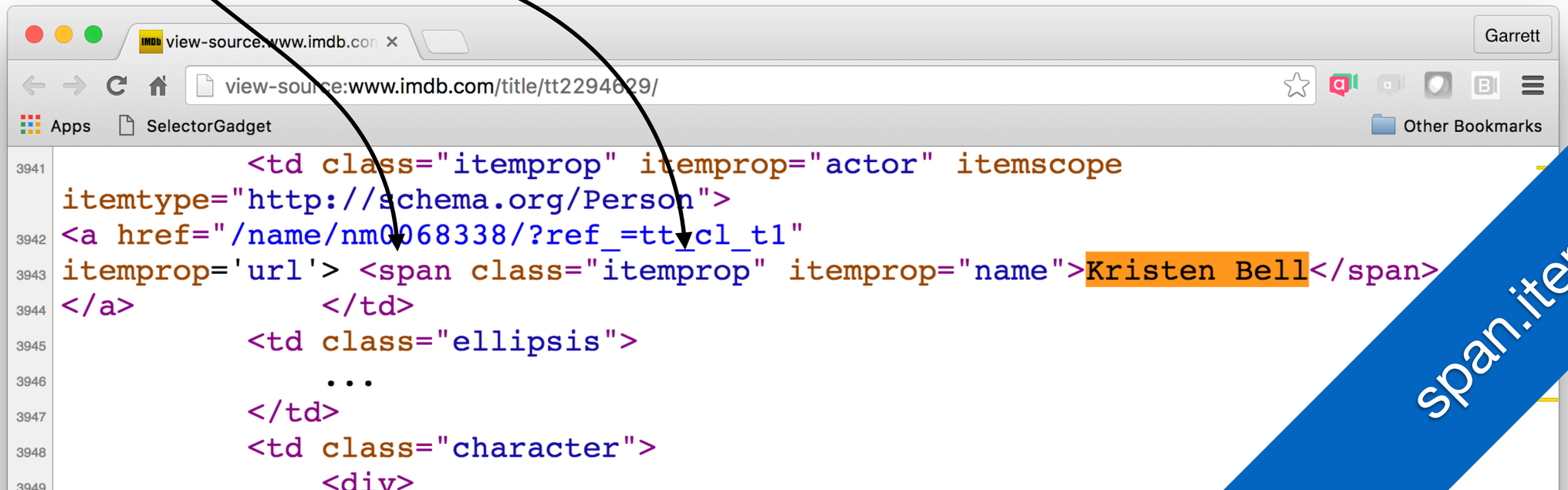
itemprop (the class)



Which CSS identifiers are associated with Kristen Bell's name?

span (the element)

itemprop (the class)



Recap

Extract information from the HTML document.
Identify information to extract with CSS selectors.

rvest

rvest

A package that makes it easy to extract info from a webpage.

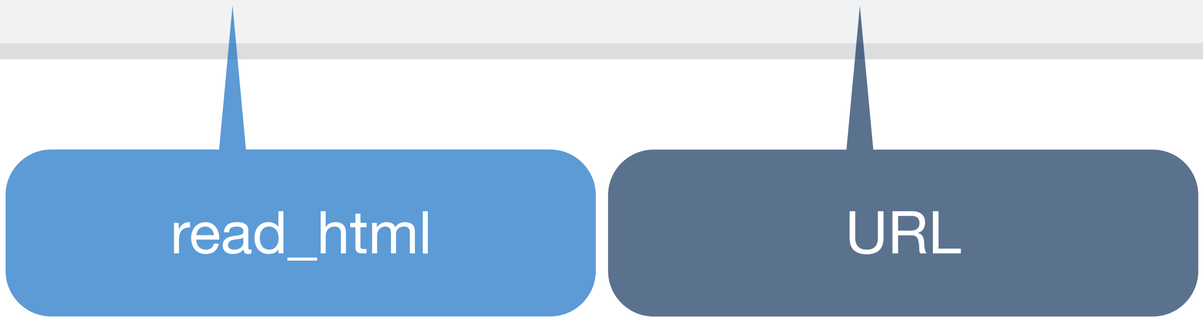
```
install.packages("rvest")
```

** This will also install xml2, a package that rvest relies on.*

Basic Workflow

1. Download the HTML and turn it into an XML file with **read_html()**

```
library(rvest)
frozen <- read_html("http://www.imdb.com/title/tt2294629/")
```



** read_html() comes in the xml2 package*

To examine contents

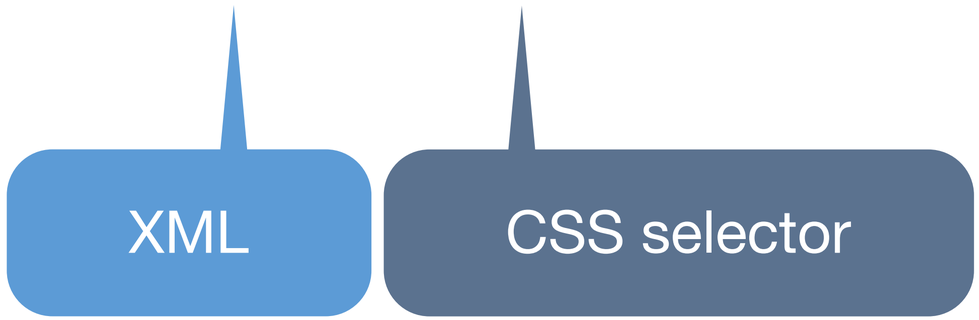
```
frozen
html_structure(frozen)
as_list(frozen)


xml_children(frozen)
xml_children(frozen)[[2]]
xml_contents(xml_children(frozen)[[2]])
```

Basic Workflow

1. Download the HTML and turn it into an XML file with **read_html()**

2. Extract specific nodes with **html_nodes()**

```
itals <- html_nodes(frozen, "em")
```



Basic Workflow

1. Download the HTML and turn it into an XML file with **read_html()**

2. Extract specific nodes with **html_nodes()**

3. Extract content from nodes with **html_text(), html_name(), html_attrs(), html_children(), html_table()**

```
itals
## {xml_nodeset (1)}
## [1] <em class="nobr">Written by\n<a href="/search/title?plot_author=DeAlan%2 ...

html_text(itals)
## [1] "Written by\nDeAlan Wilson for ComedyE.com"

html_name(itals)
## [1] "em"

html_children(itals)
## {xml_nodeset (1)}
## [1] <a href="/search/title?plot_author=DeAlan%20Wilson%20for%20ComedyE.com&a ...

html_attr(itals, "class")
## [1] "nobr"

html_attrs(itals)
## [[1]]
##  class
## "nobr"
```

Recap

1. Download the HTML and turn it into an XML file with read_html()

2. Extract specific nodes with html_nodes()

3. Extract content from nodes with html_text(), html_name(), html_attrs(), html_children(), html_table()

Your Turn

1. Read in the Frozen HTML.
2. Select the nodes that are both **spans** and class = "**itemprop**".
3. Extract the text from the nodes.

Do you collect the cast names and only the cast names?

```
library(rvest)
frozen <- read_html("http://www.imdb.com/title/tt2294629/")
cast <- html_nodes(frozen, "span.itemprop")
html_text(cast)
```

```
##  [1] "Animation"                    "Adventure"
##  [3] "Comedy"                       "Chris Buck"
##  [5] "Jennifer Lee"                 "Jennifer Lee"
##  [7] "Hans Christian Andersen"      "Kristen Bell"
##  [9] "Idina Menzel"                 "Jonathan Groff"
## [11] "Kristen Bell"                 "Idina Menzel"
## [13] "Jonathan Groff"               "Josh Gad"
## [15] "Santino Fontana"              "Alan Tudyk"
## [17] "Ciarán Hinds"                 "Chris Williams"
## [19] "Stephen J. Anderson"          "Maia Wilson"
## [21] "Edie McClurg"                 "Robert Pine"
## [23] "Maurice LaMarche"             "Livvy Stubenrauch"
## [25] "Eva Bella"                    "snowman"
## [27] "sister love"                  "sister sister relationship"
## [29] "magic"                        "snow"
## [31] "Walt Disney Animation Studios" "Walt Disney Pictures"
```

But we've scraped too much info

selectorGadget

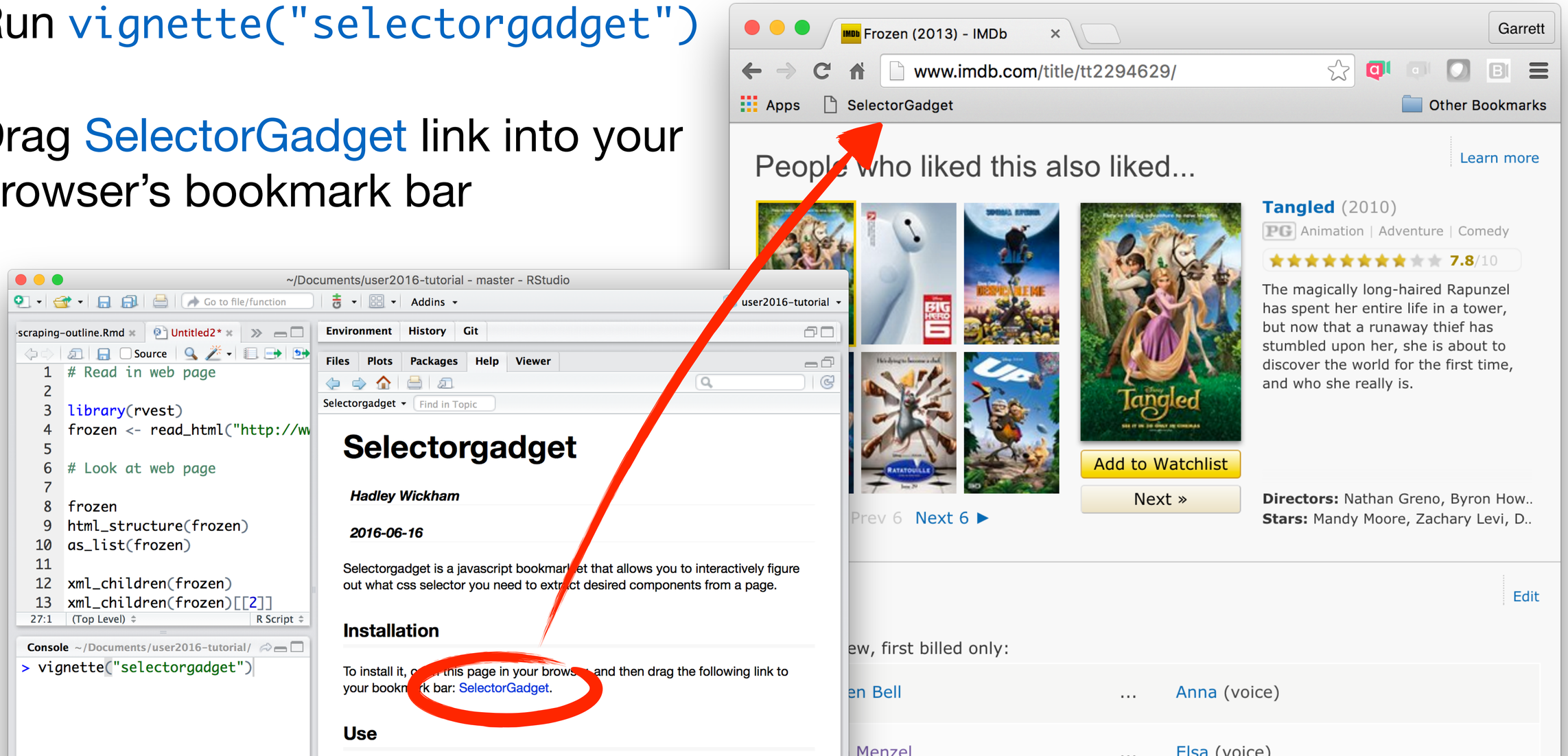
selectorGadget

A GUI tool to identify CSS selector combinations

To Install

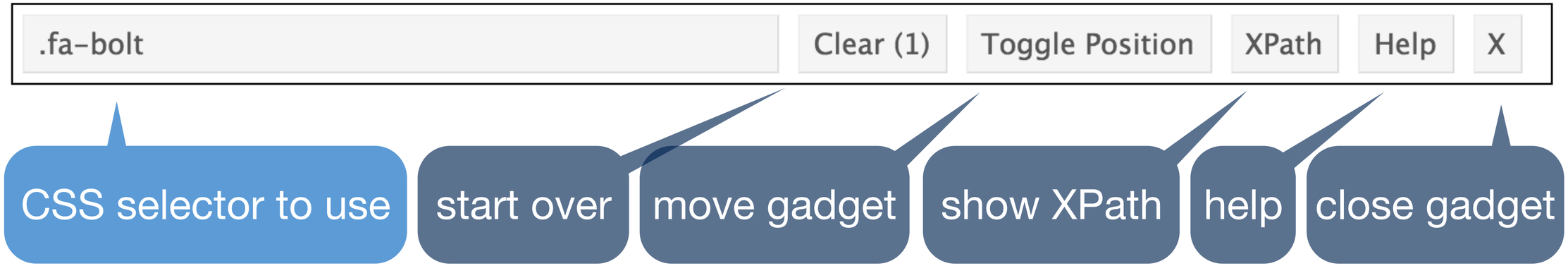
1. Run `vignette("selectorgadget")`

2. Drag SelectorGadget link into your browser's bookmark bar



To Use

1. **Navigate** to a webpage
2. **Open** the SelectorGadget bookmark
3. **Click** on item to scrape
4. **Click** on yellow items You do not want to scrape
5. **Click** on additional items that you do want to scrape
6. **Copy** selector to use with html_nodes()


.fa-bolt
Clear (1)
Toggle Position
XPath
Help
X
CSS selector to use
start over
move gadget
show XPath
help
close gadget

Your Turn

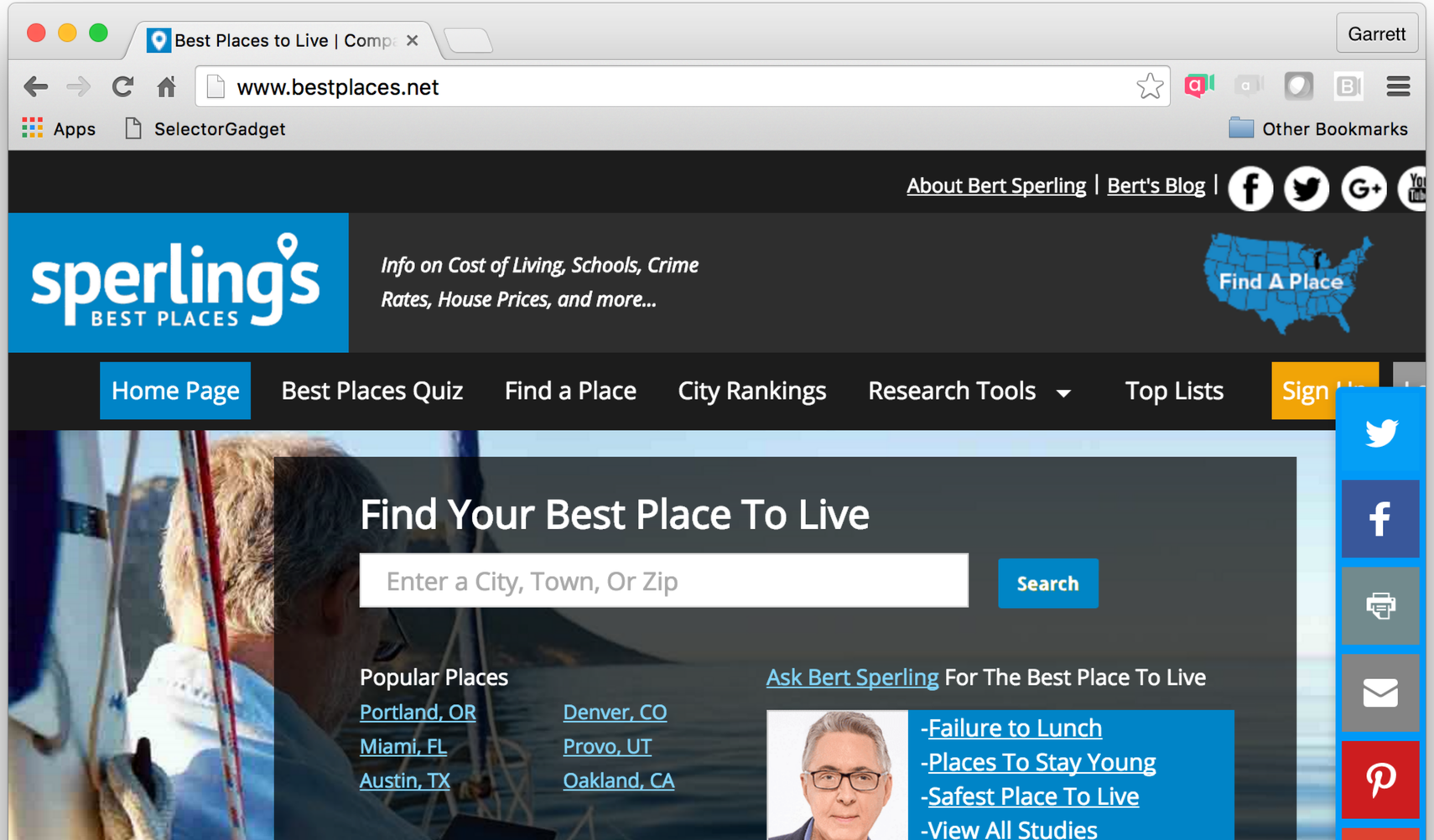
Install SelectorGadget in your browser.

Then use selectorGadget to find a CSS selector combination that identifies just the cast member names.

```
cast2 <- html_nodes(frozen, "#titleCast span.itemprop")
html_text(cast2)

cast3 <- html_nodes(frozen, ".itemprop .itemprop")
html_text(cast3)
```

Bringing it home

http://www.bestplaces.net/



Your Turn

Look up the cost of living for your hometown on http://www.bestplaces.net/. Then extract it with html_nodes() and html_text().

```
kw <- read_html("http://www.bestplaces.net/cost_of_living/
city/florida/key_west")

col <- html_nodes(kw, css = "#mainContent_dgCostOfLiving
tr:nth-child(2) td:nth-child(2)")
html_text(col)

kw %>%
  html_nodes(css = "#mainContent_dgCostOfLiving tr:nth-
child(2) td:nth-child(2)") %>%
  html_text()
```

tables

Tables

Use html_table() to scrape whole tables of data as a data frame.

```
tables <- html_nodes(kw, css = "table")
html_table(tables, header = TRUE)[[2]]
```

Your Turn

Visit the Climate tab for your home town. Extract the climate statistics of your hometown as a data frame with useful column names.

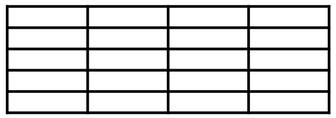
```
kw2 <- read_html("http://www.bestplaces.net/climate/city/
florida/key_west")

climate <- html_nodes(kw2, css = "table")
html_table(climate, header = TRUE)[[2]]

kw2 %>%
  html_nodes(css = "table") %>%
  html_table(header = TRUE) %>%
  .[[2]]
```

Recap

Pull from HTML, use HTML/CSS structure

read_html() ? **html_nodes()** ? **html_text()**,

selectorGadget for finding useful selector combinations



html_table() for tables

thank you