

Web Scraping

HELLO

my name is

Garrett



@StatGarrett

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

www.imdb.com/title/tt2294629/

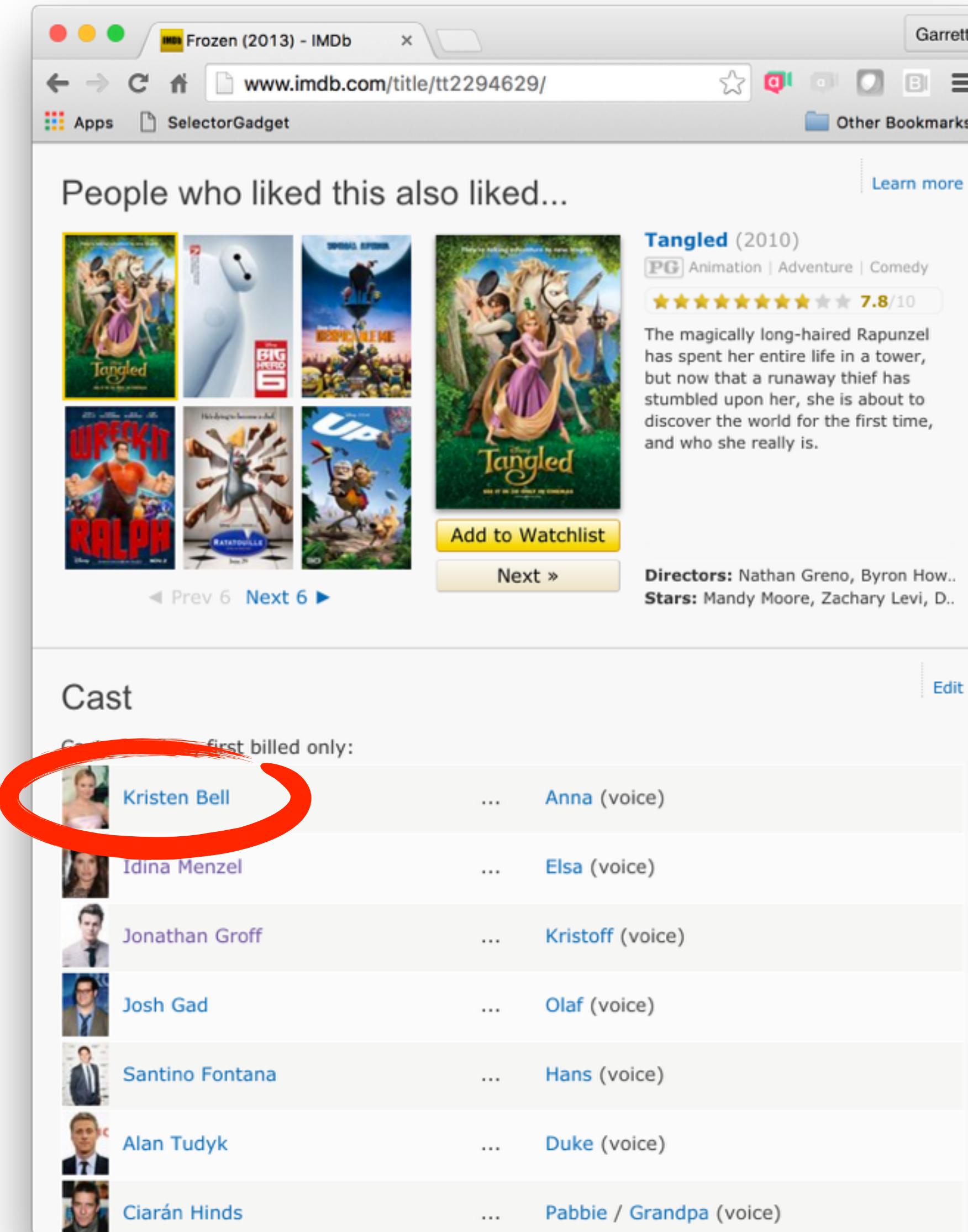
The screenshot shows a web browser window displaying the IMDb movie page for "Frozen (2013)". The page includes the movie's title, rating (7.6/10), and a thumbnail image. An advertisement for the "ALL-NEW PRIUS" is overlaid on the page, featuring a red Prius driving across a bridge with helicopters in the background. The ad text reads: "EXHILARATING DRIVING DYNAMICS. TIME FOR HYBRIDS TO HAVE FUN." A "LEARN MORE" button and a note stating "Prototype shown with options." are also visible. The browser interface shows the URL "www.imdb.com/title/tt2294629/" in the address bar and various browser extensions.

IMDb Picks: June

Outline

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

Strategy



People who liked this also liked...

Tangled (2010) PG Animation | Adventure | Comedy 7.8/10

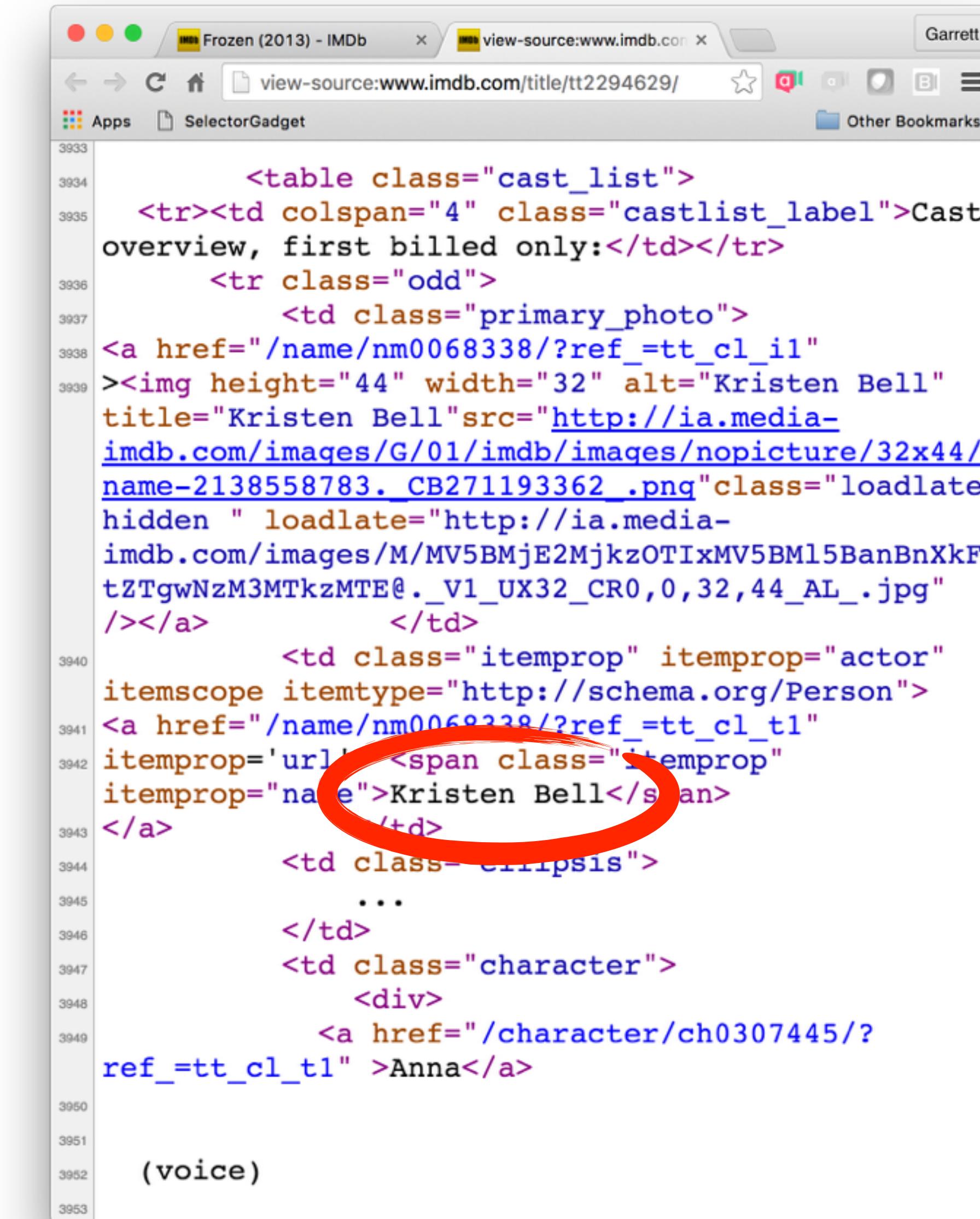
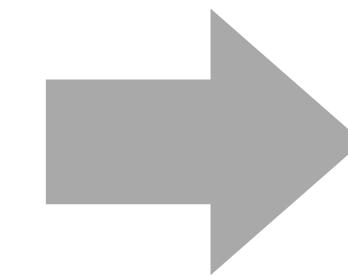
The magically long-haired Rapunzel has spent her entire life in a tower, but now that a runaway thief has stumbled upon her, she is about to discover the world for the first time, and who she really is.

Add to Watchlist Next »

◀ Prev 6 Next 6 ▶

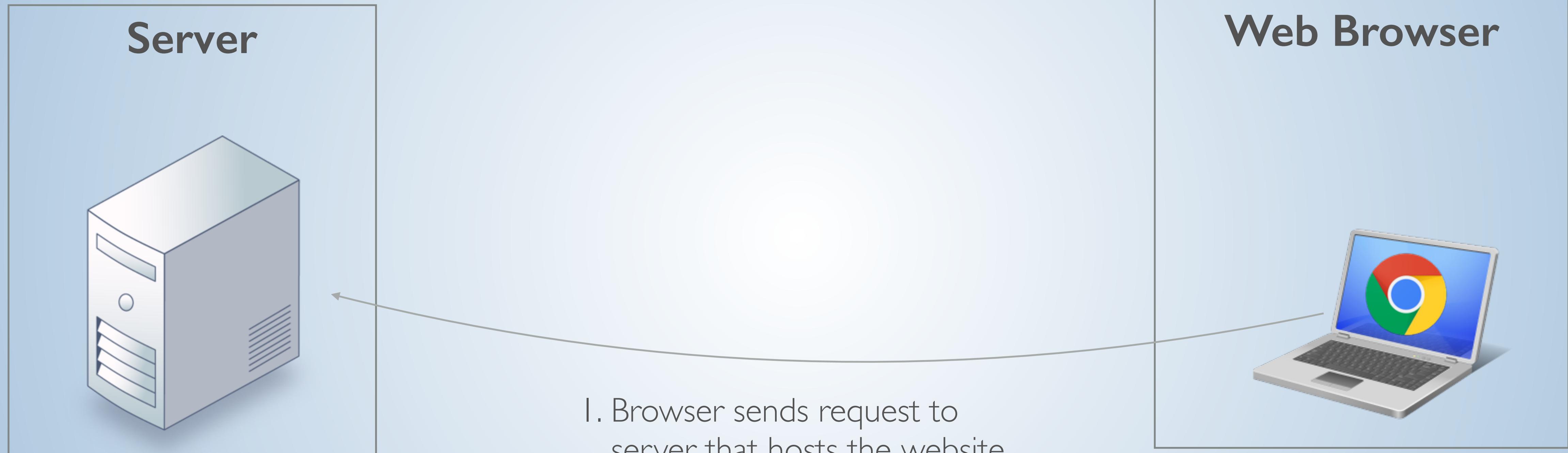
Cast

	Cast	Character
	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)



```
3933 <table class="cast_list">
3934 <tr><td colspan="4" class="castlist_label">Cast
3935 overview, first billed only:</td></tr>
3936 <tr class="odd">
3937 <td class="primary_photo">
3938 <a href="/name/nm0068338/?ref_=tt_cl_i1"
3939 ></a> </td>
3940 <td class="itemprop" itemprop="actor"
3941 itemscope itemtype="http://schema.org/Person">
3942 <a href="/name/nm0068338/?ref_=tt_cl_t1"
3943 itemprop='url'><span class="itemprop"
3944 itemprop="name">Kristen Bell</span>
3945 </a> </td>
3946 <td class="ellipsis">
3947 ...
3948 </td>
3949 <td class="character">
3950 <div>
3951 <a href="/character/ch0307445/?ref_=tt_cl_t1" >Anna</a>
3952
3953 (voice)
```

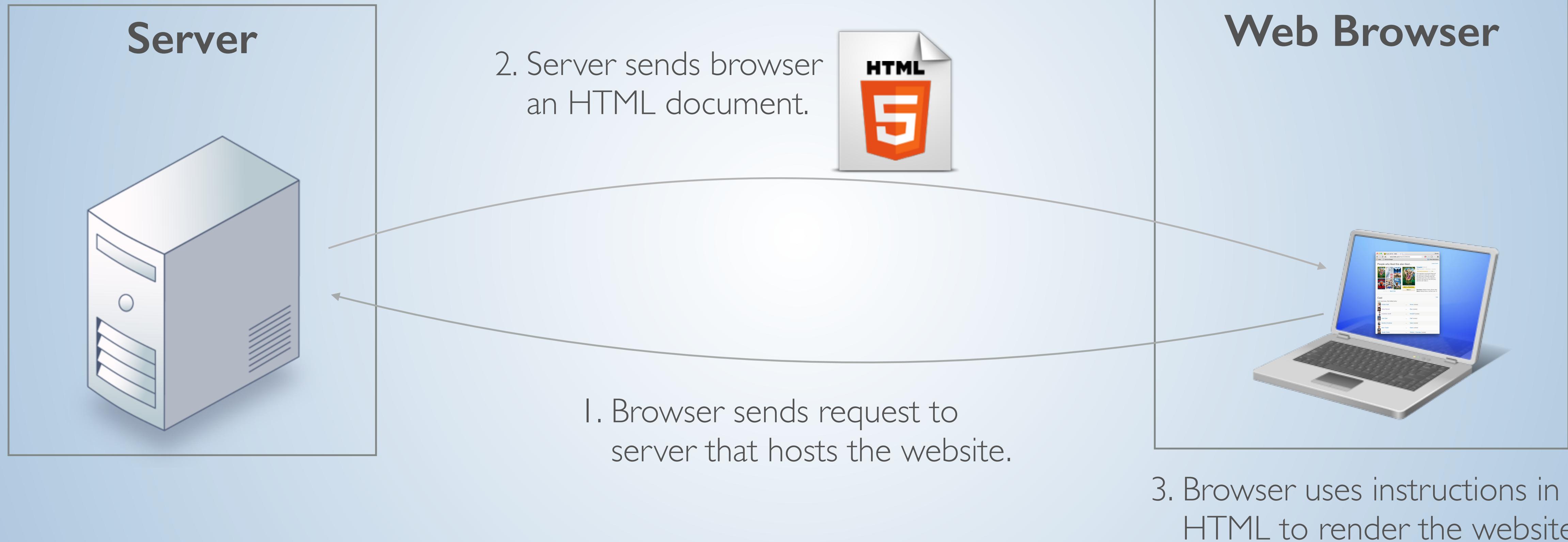
HTML (Review)



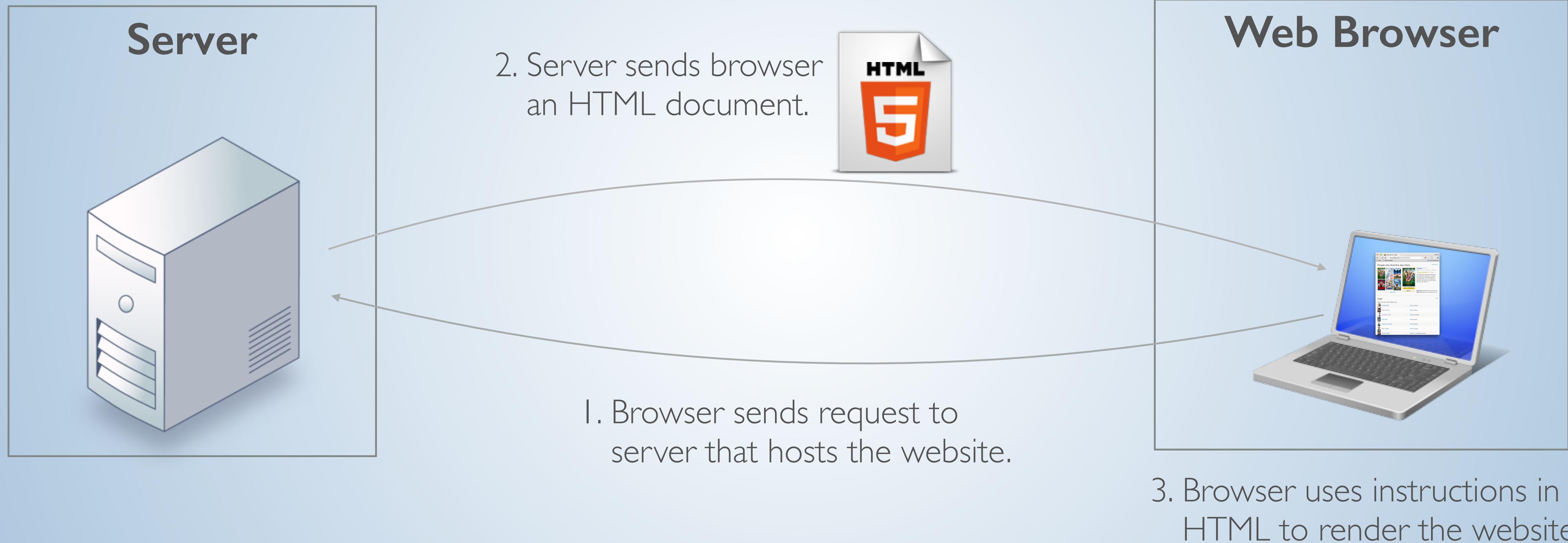
HTML (Review)



HTML (Review)



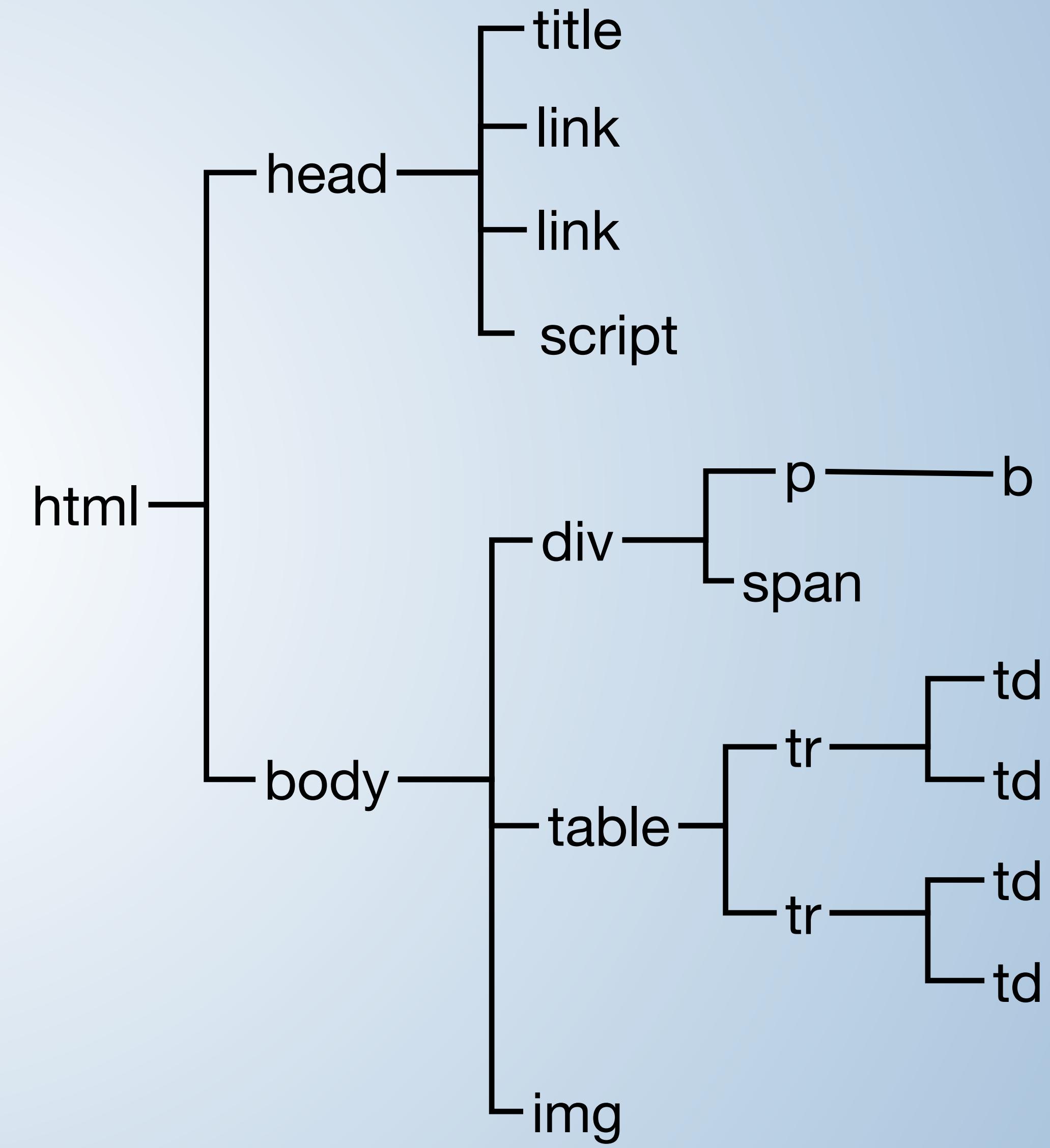
HTML (Review)



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>
    
  </body>
</html>
```



HTML (Review)

Each element in the page is created by a tag.

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

tag name

attribute
(name)

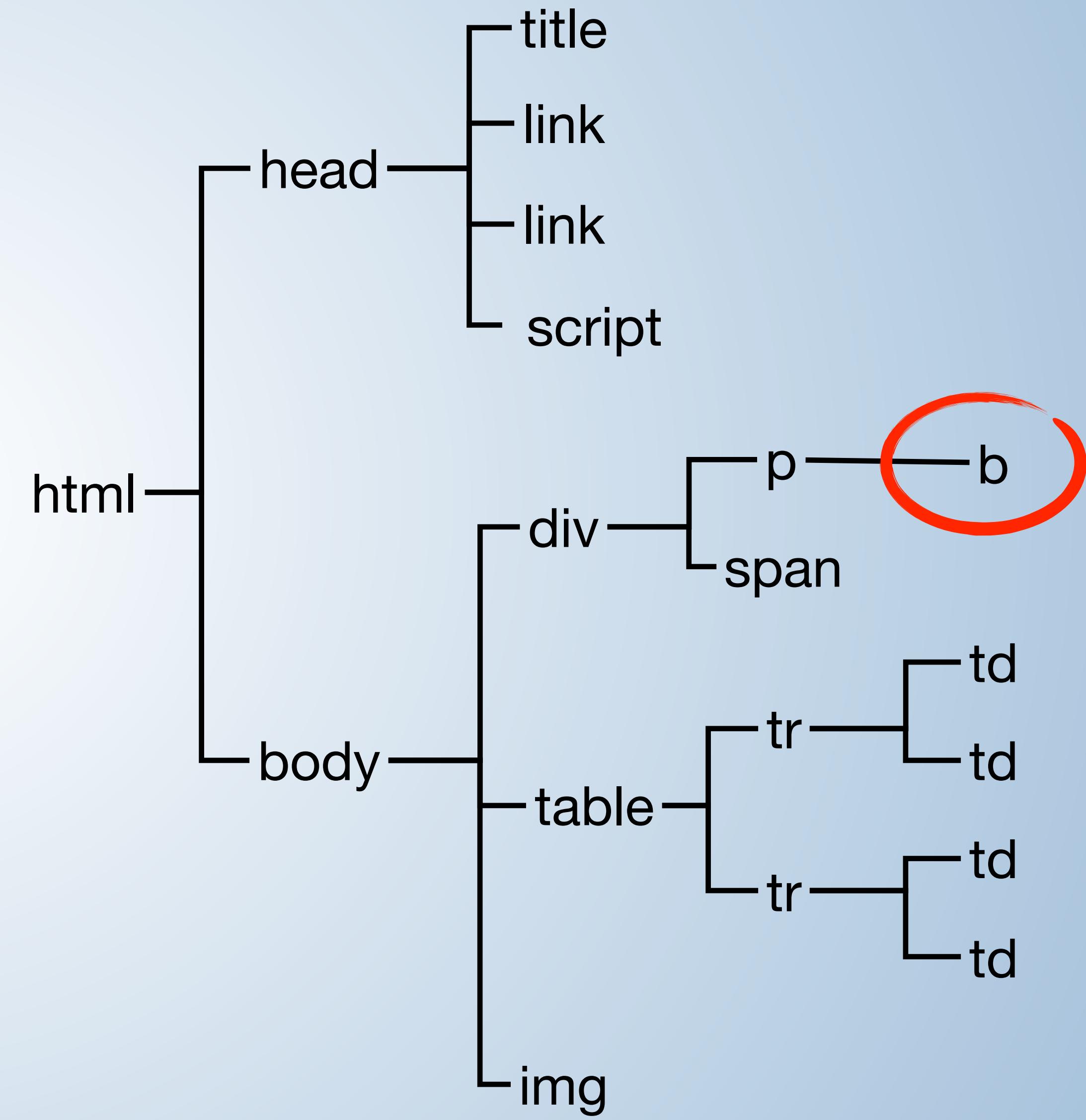
attribute
(value)

content

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>
    
  </body>
</html>
```



Your Turn

Find the source code for the Frozen IMDB page



Chrome

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

03 : 00



Explorer

1. Right click
 on page
 a. Click View
 Source



Firefox

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



Safari

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.

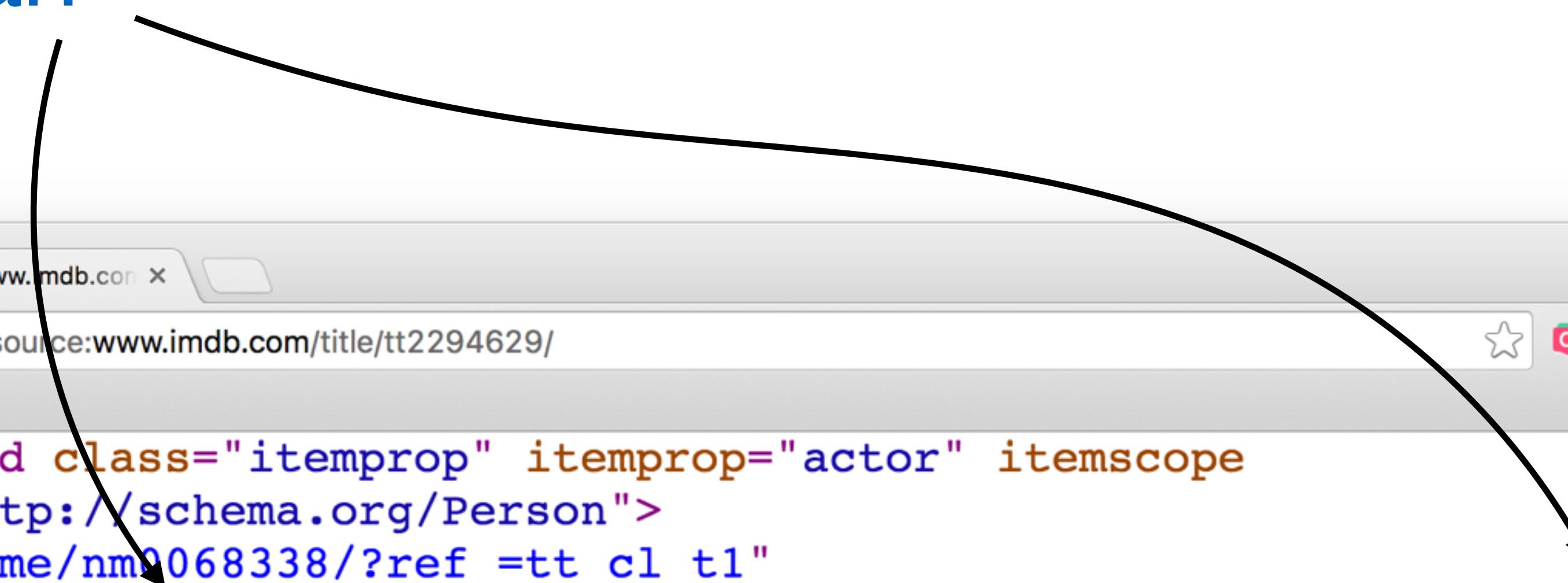
Locate the piece of HTML that inserts Kristen Bell into the Cast section.

Which HTML tag surrounds her name?

01:00

Which HTML tag surrounds her name?

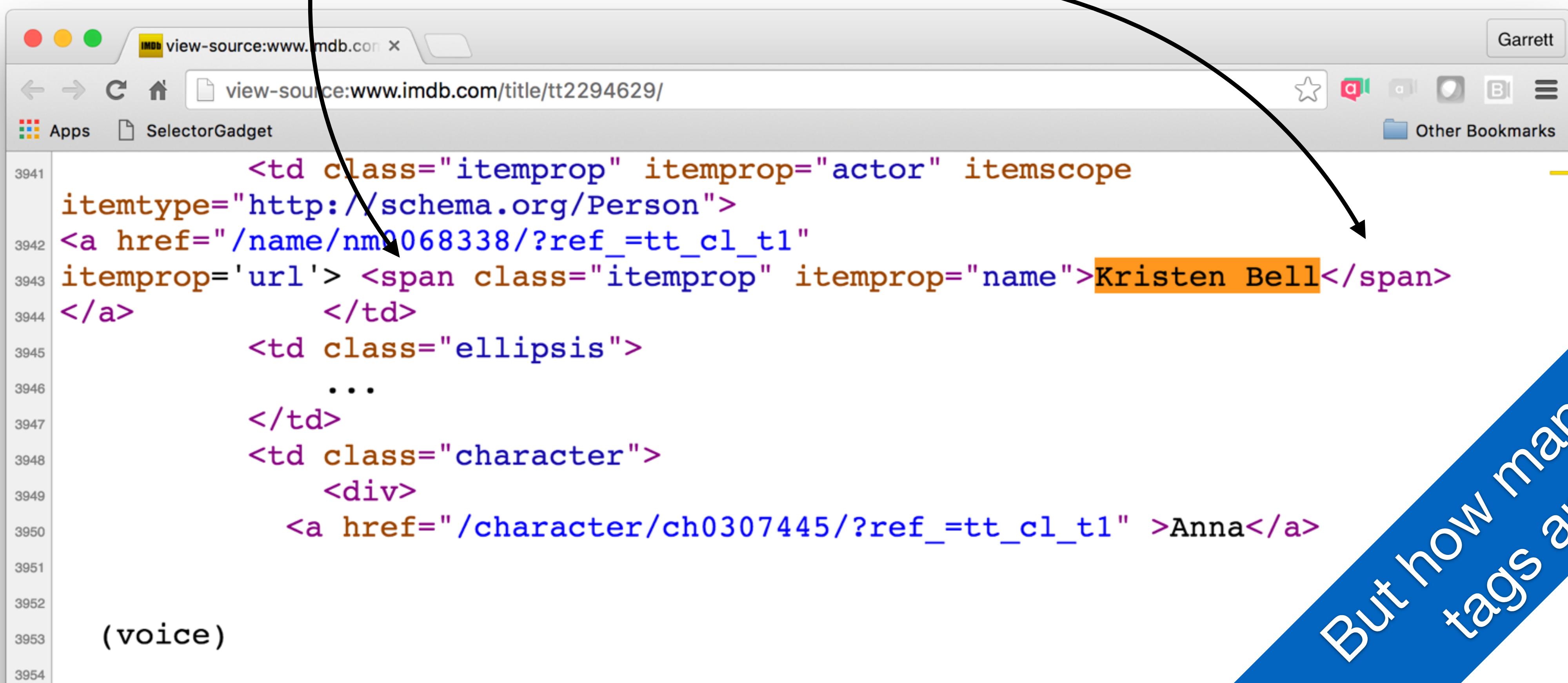
span



```
view-source:www.imdb.com x Garrett
view-source:www.imdb.com/title/tt2294629/
Apps SelectorGadget Other Bookmarks
3941 <td class="itemprop" itemprop="actor" itemscope
itemtype="http://schema.org/Person">
3942 <a href="/name/nm0068338/?ref_=tt_cl_t1"
3943 itemprop='url'> <span class="itemprop" itemprop="name">Kristen Bell</span>
3944 </a>
3945 <td class="ellipsis">
3946 ...
3947 </td>
3948 <td class="character">
3949 <div>
3950 <a href="/character/ch0307445/?ref_=tt_cl_t1" >Anna</a>
3951
3952
3953 (voice)
3954
```

Which HTML tag surrounds her name?

span



view-source:www.imdb.com

view-source:www.imdb.com/title/tt2294629/

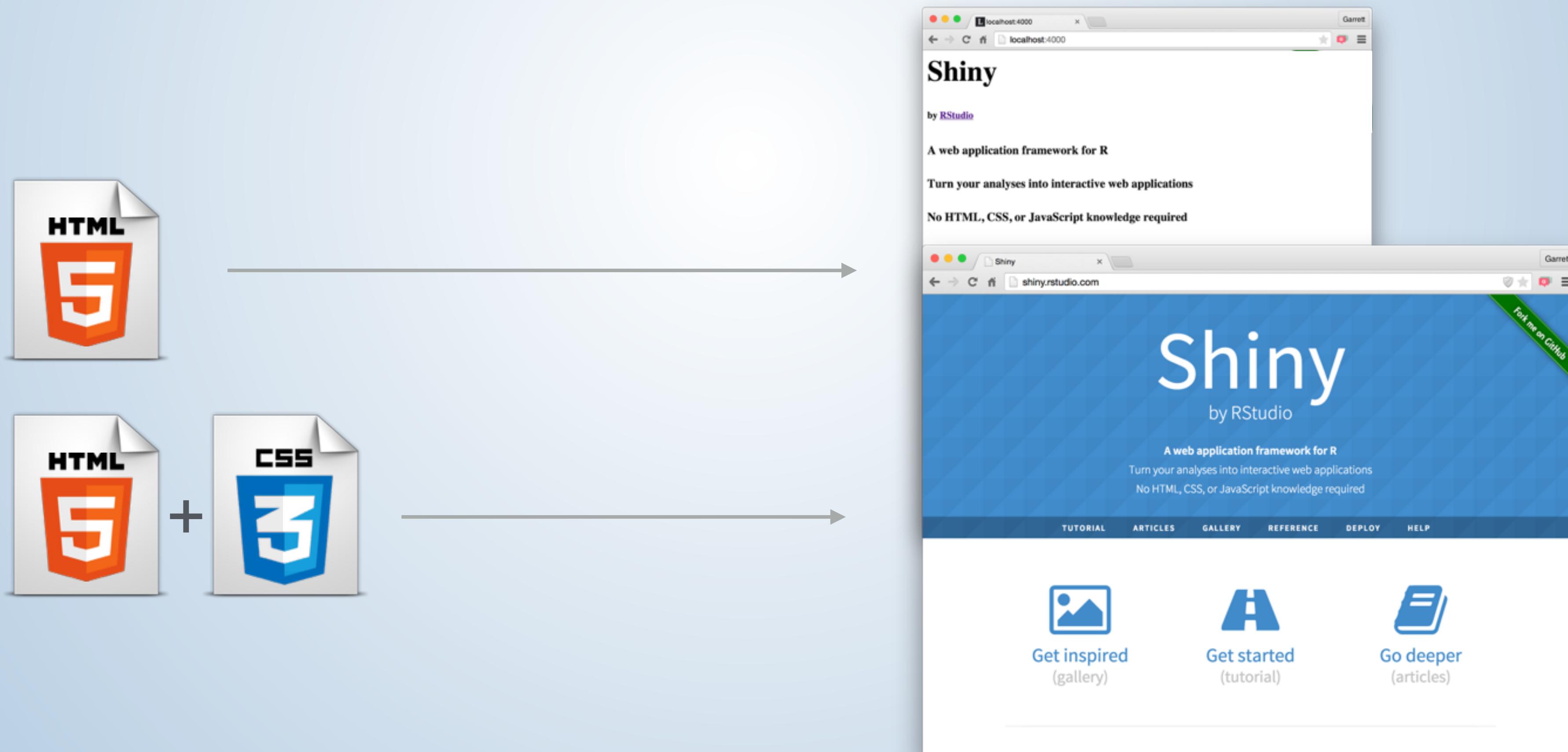
```
3941     <td class="itemprop" itemprop="actor" itemscope
3942 itemtype="http://schema.org/Person">
3943 <a href="/name/nm0068338/?ref_=tt_cl_t1"
3944 itemprop='url'> <span class="itemprop" itemprop="name">Kristen Bell</span>
3945             </td>
3946             <td class="ellipsis">
3947               ...
3948             </td>
3949             <td class="character">
3950               <div>
3951                 <a href="/character/ch0307445/?ref_=tt_cl_t1" >Anna</a>
3952             </div>
3953           (voice)
```

But how many tags are there?

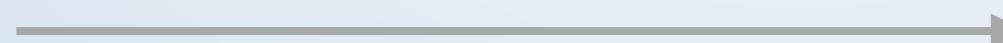
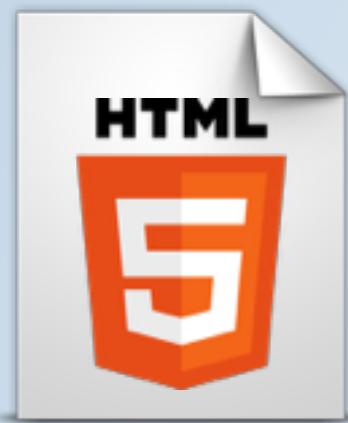
css selectors

CSS (Review)

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



CSS (Review)



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.

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

tag name

class
(optional)

id
(optional)

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

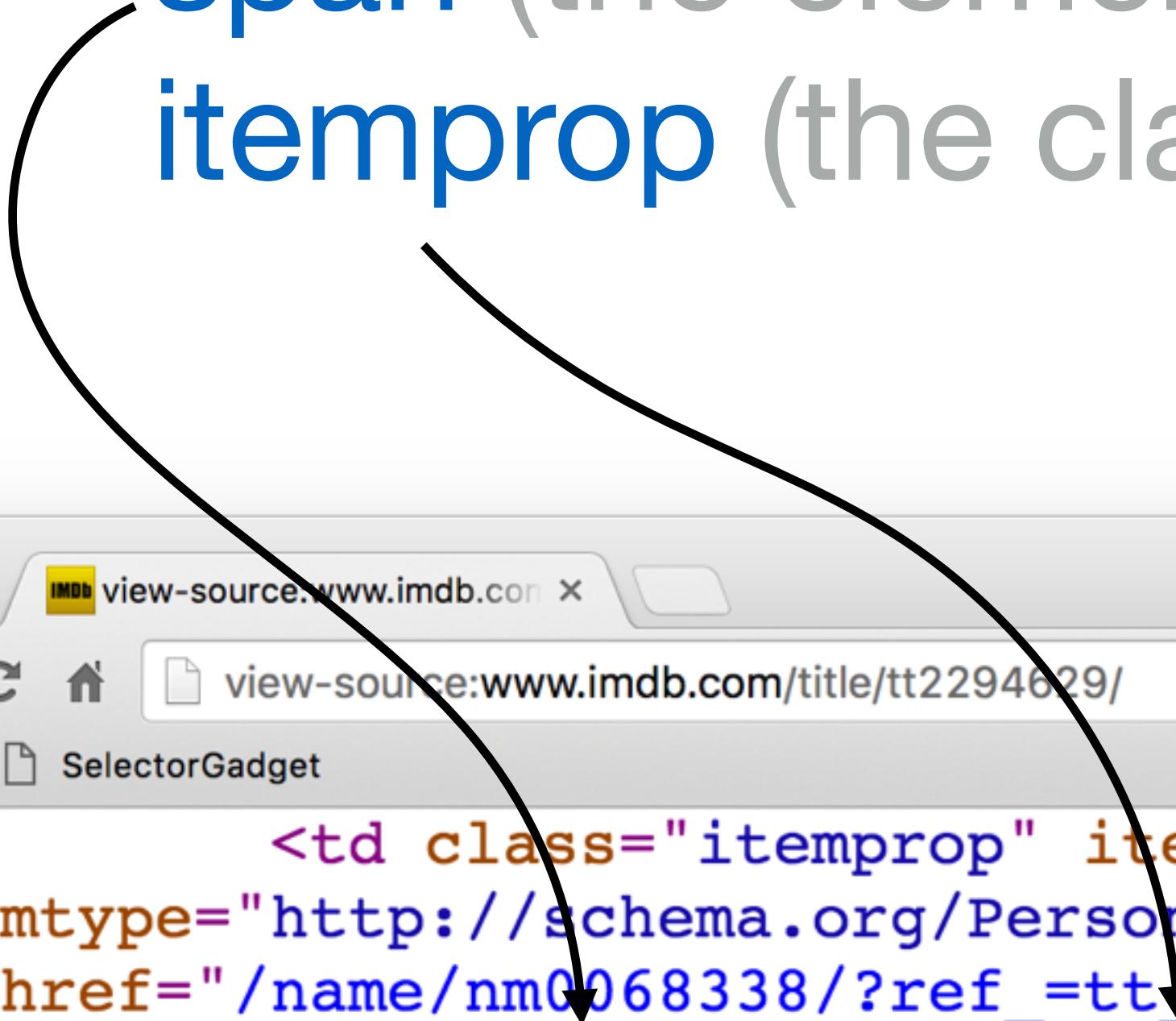
Which CSS identifiers are associated with Kristen Bell's name in the Frozen page? Write a CSS selector that targets them.



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

span (the element)

itemprop (the class)



```
view-source:www.imdb.com
view-source:www.imdb.com/title/tt2294629/
Apps SelectorGadget Other Bookmarks
3941     <td class="itemprop" itemprop="actor" itemscope
3942     itemtype="http://schema.org/Person">
3943     <a href="/name/nm0068338/?ref_=tt_cl_t1"
3944         itemprop='url'> <span class="itemprop" itemprop="name">Kristen Bell</span>
3945     </a>
3946     <td class="ellipsis">
3947         ...
3948     </td>
3949     <td class="character">
3950         <div>
```

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

span (the element)

itemprop (the class)

```
3941      <td class="itemprop" itemprop="actor" itemscope
3942      itemtype="http://schema.org/Person">
3943      <a href="/name/nm0068338/?ref_=tt_cl_t1"
3944          itemprop='url'> <span class="itemprop" itemprop="name">Kristen Bell</span>
3945      </a>          </td>
3946      <td class="ellipsis">
3947          ...
3948      </td>
3949      <td class="character">
3950          <div>
```

Span.itemprop

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

URL

* `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")
```

XML

CSS selector

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<br><a href="/search/title?plot_author=DeAlan%2 ...
```

```
html_text(itals)
```

```
## [1] "Written by<br>DeAlan 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

The image shows a composite of two screenshots. On the left, the RStudio interface displays an R script named 'scraping-outline.Rmd' with the following code:

```
1 # Read in web page
2
3 library(rvest)
4 frozen <- read_html("http://www.imdb.com/title/tt2294629/")
5
6 # Look at web page
7
8 frozen
9 html_structure(frozen)
10 as_list(frozen)
11
12 xml_children(frozen)
13 xml_children(frozen)[[2]]
```

In the R console below, the command `> vignette("selectorgadget")` is run.

On the right, a screenshot of a web browser (Safari) shows the IMDb page for the movie 'Frozen (2013)'. A red arrow points from the 'Selectorgadget' link in the RStudio vignette to the 'SelectorGadget' bookmark in the browser's bookmark bar. The browser window also shows the URL `www.imdb.com/title/tt2294629/` and the title 'IMDb Frozen (2013) - IMDb'.

The browser content includes a section titled 'People who liked this also liked...' with movie posters for 'Despicable Me', 'Big Hero 6', 'Up', 'Ratatouille', and 'Tangled'. Below this is a 'Learn more' button. To the right, there is a detailed movie summary for 'Tangled (2010)' with a rating of 7.8/10, directors Nathan Greno and Byron Howard, and stars Mandy Moore and Zachary Levi.

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.

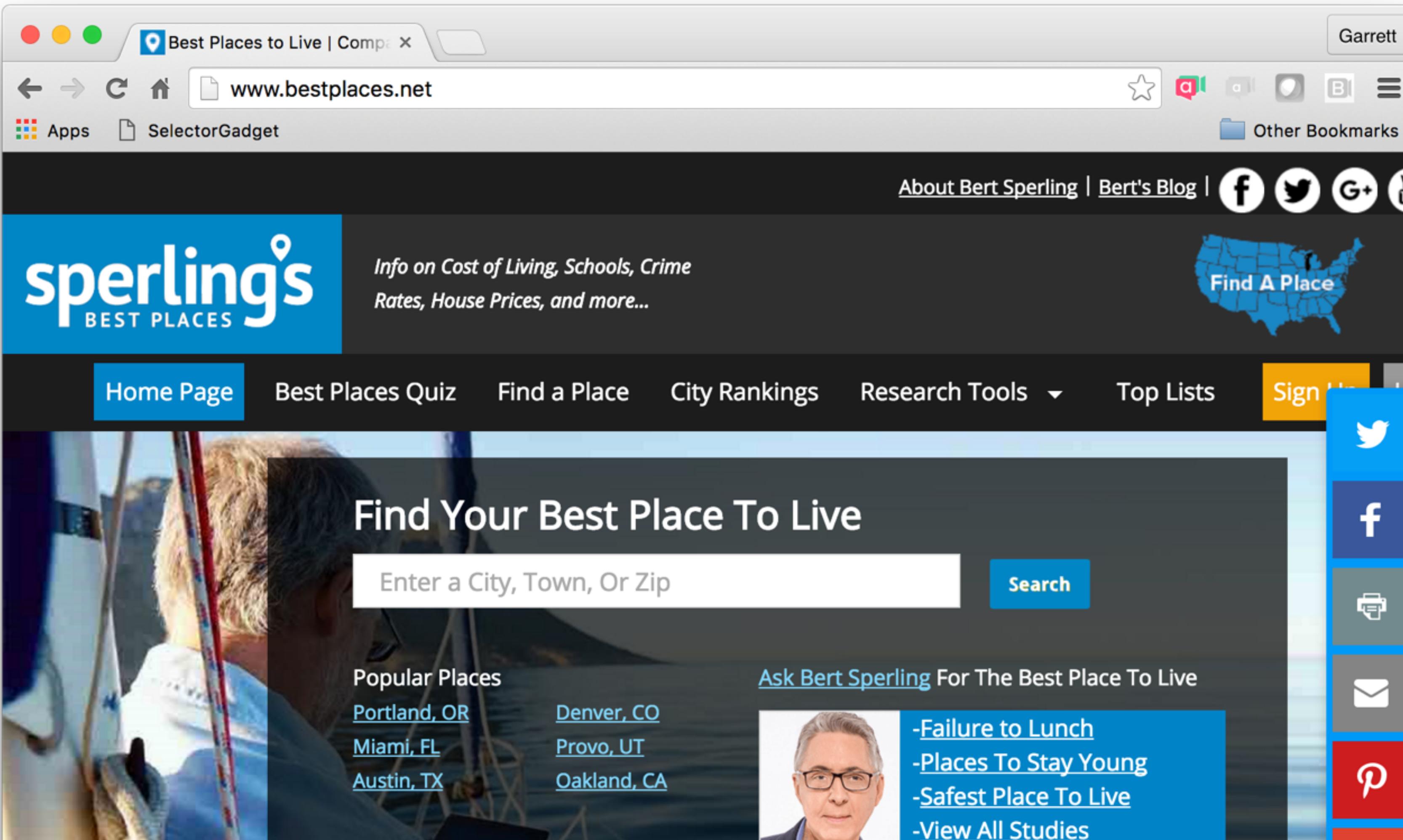


```
cast2 <- html_nodes(frozen, "#titleCast span.itemprop")
html_text(cast2)

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.



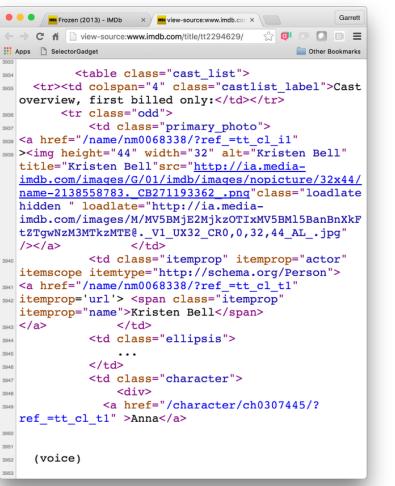
```
kw2 <- read_html("http://www.bestplaces.net/climate/city/  
florida/key_west")
```

```
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()**, etc.

selectorGadget for finding useful selector combinations



html_table() for tables

thank you