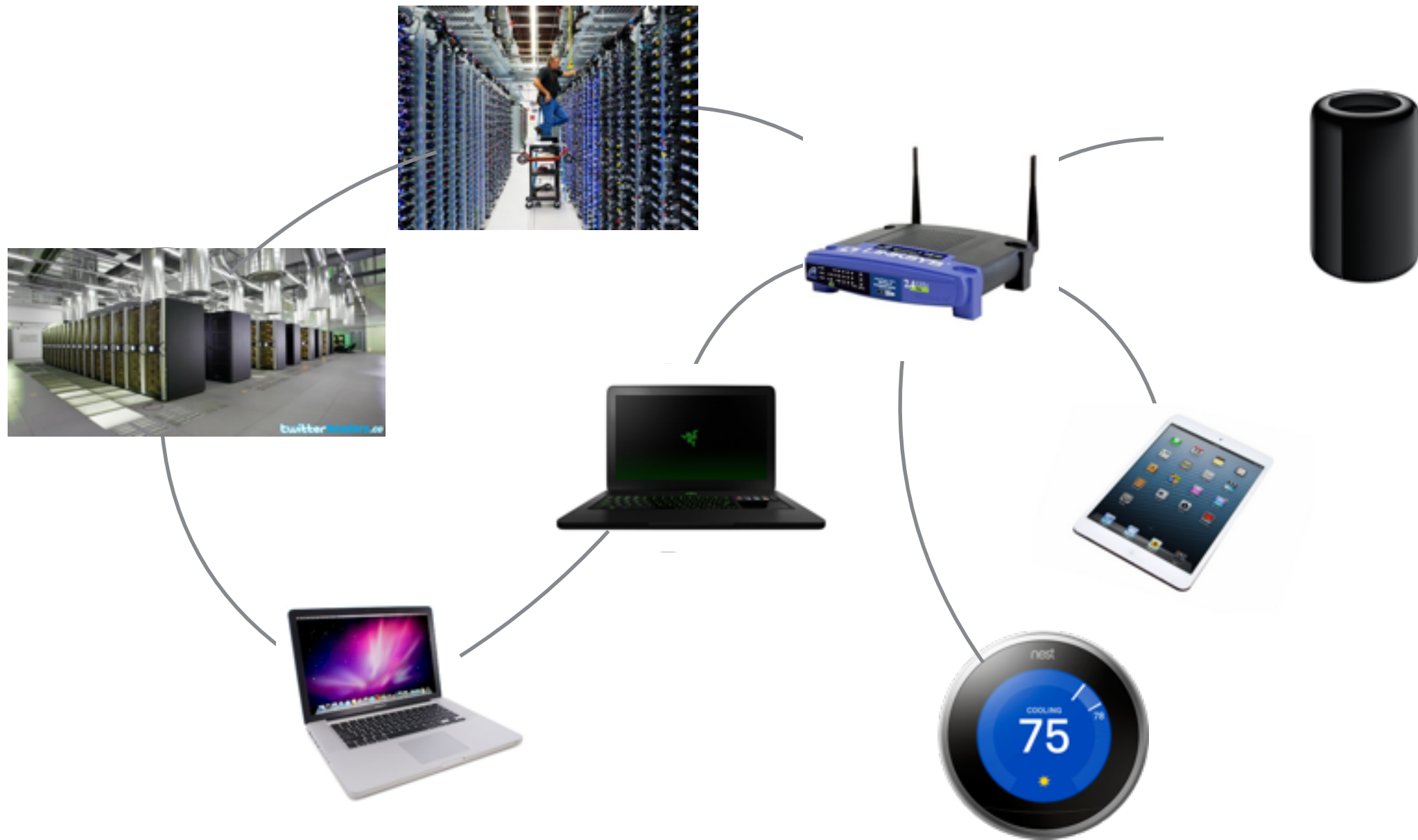
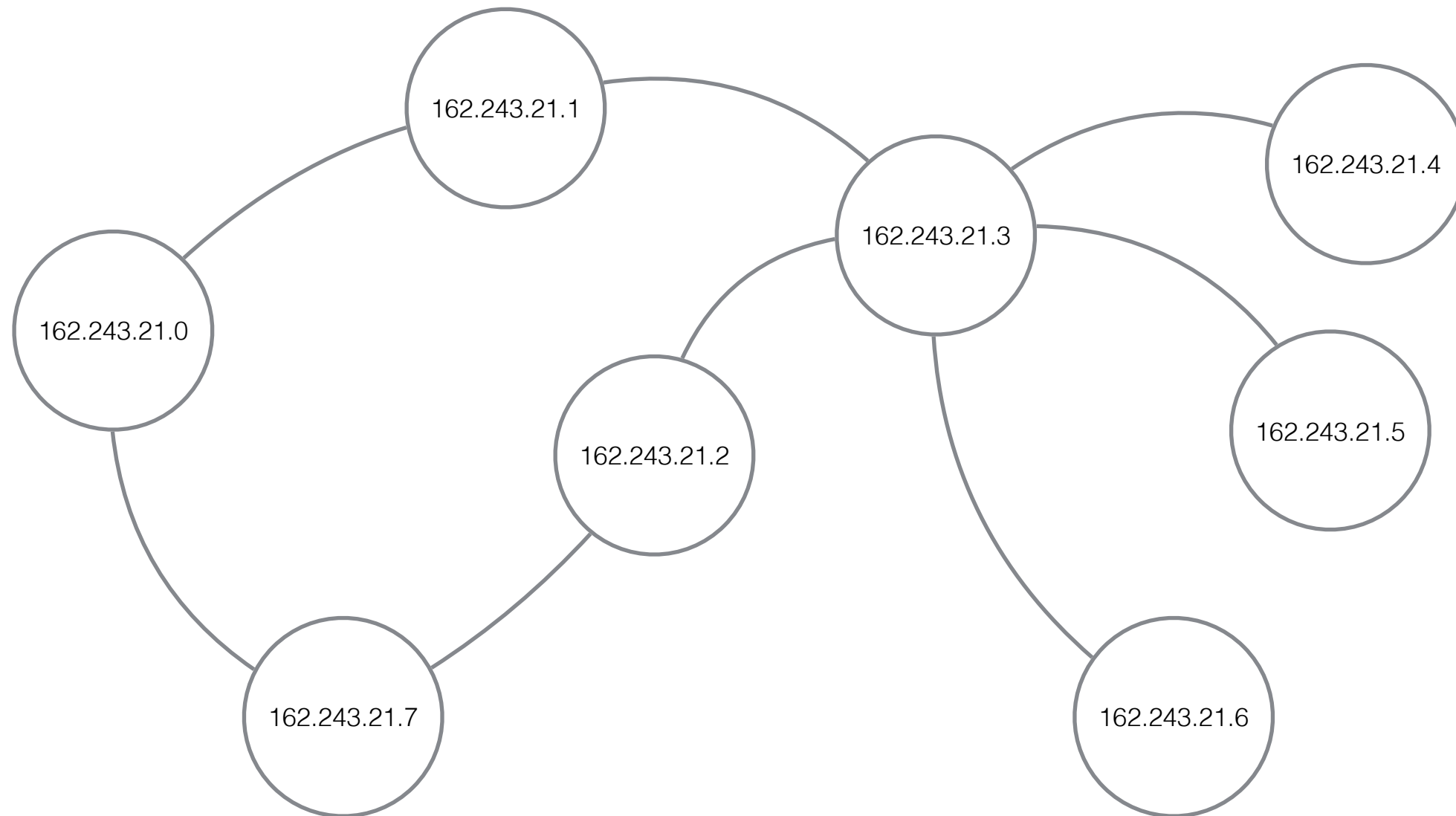


Our Web Application

How do computers talk to each other?



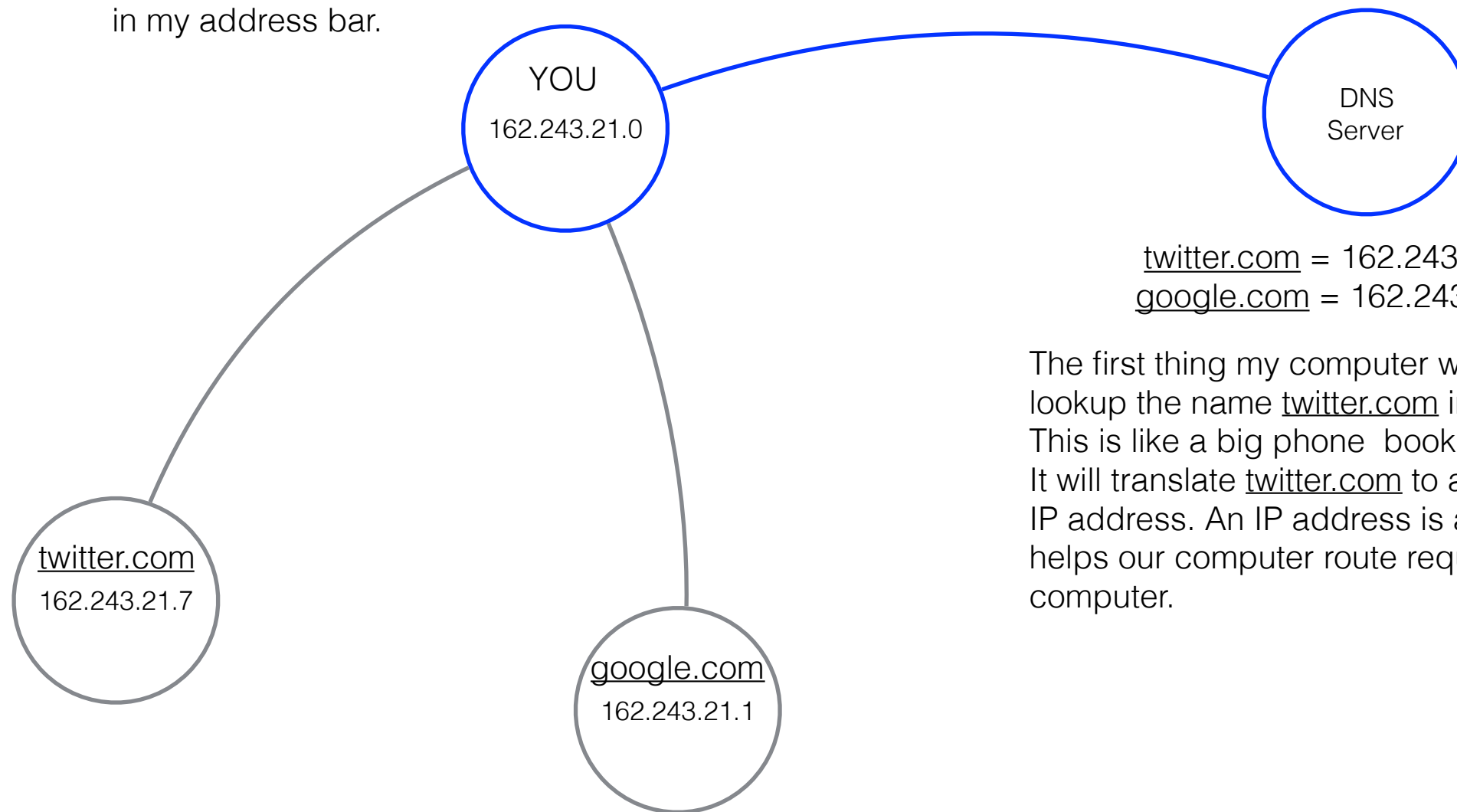
How do computers talk to each other?



https://youtu.be/C3sr7_0FyPA

How do we find other computers?

I want to go to twitter.com
So I type the URL http://twitter.com
in my address bar.

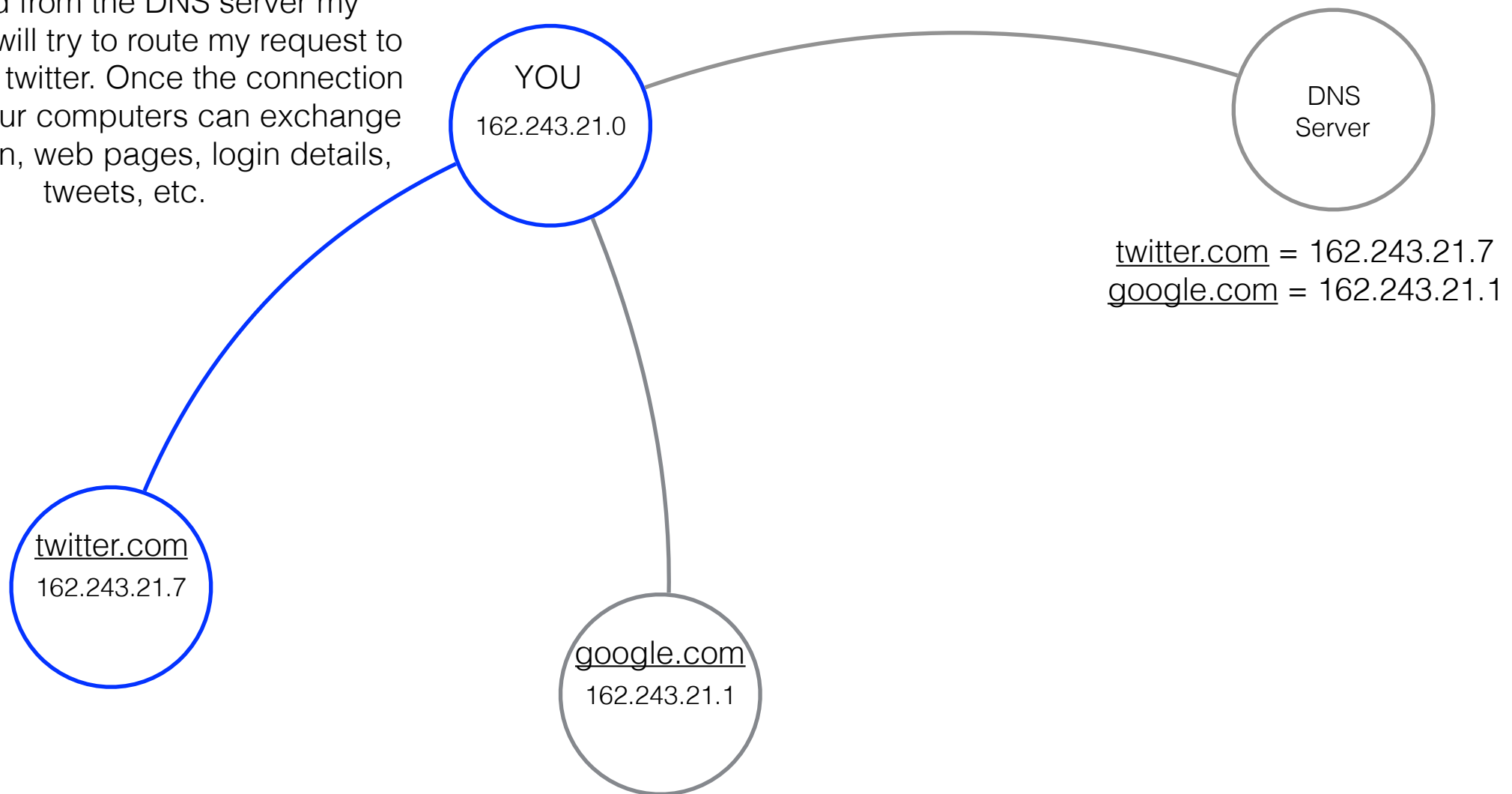


twitter.com = 162.243.21.7
google.com = 162.243.21.1

The first thing my computer will do is lookup the name twitter.com in a DNS server. This is like a big phone book of the internet. It will translate twitter.com to an IP address. An IP address is a number that helps our computer route requests to another computer.

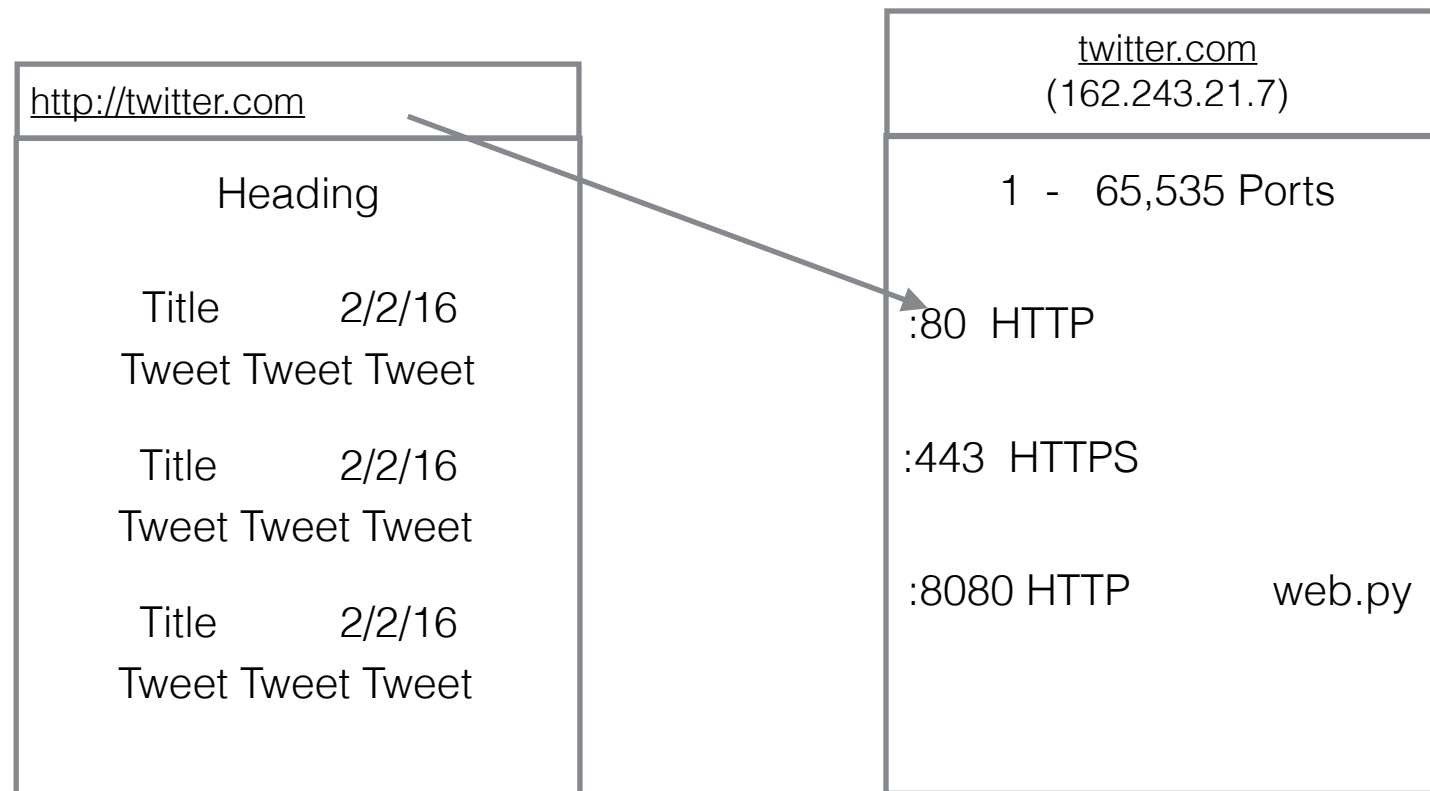
How do we find other computers?

Using the IP address 162.243.21.7 returned from the DNS server my computer will try to route my request to connect to twitter. Once the connection is made, our computers can exchange information, web pages, login details, tweets, etc.



I found and connected to a computer, now what?

How do I get a web page?



On every computer you connect to there are a set of ports that programs **listen** on for connections.

Where an IP address can be thought of as a phone number. A port can be thought of as an office extension.

Some of these ports are predefined for specific types of connections.

Port 80 is typically used for web pages.
Port 443 is used for secure web pages.

Port 22 is for SSH
/etc/services has a list of common ports

Once your browser connects to a port, the program or web application listening on that port will send you content. Content could be a webpage, images, code, and other things needed to show you a website.

You can write programs that listen on ports not already being used.

We're going to write a web application that listens in on port 8080.

Our Web Application

http://0.0.0.0:8080/

Heading

Title 2/2/16
Entry Entry Entry

Title 2/2/16
Entry Entry Entry

Title 2/2/16
Entry Entry Entry

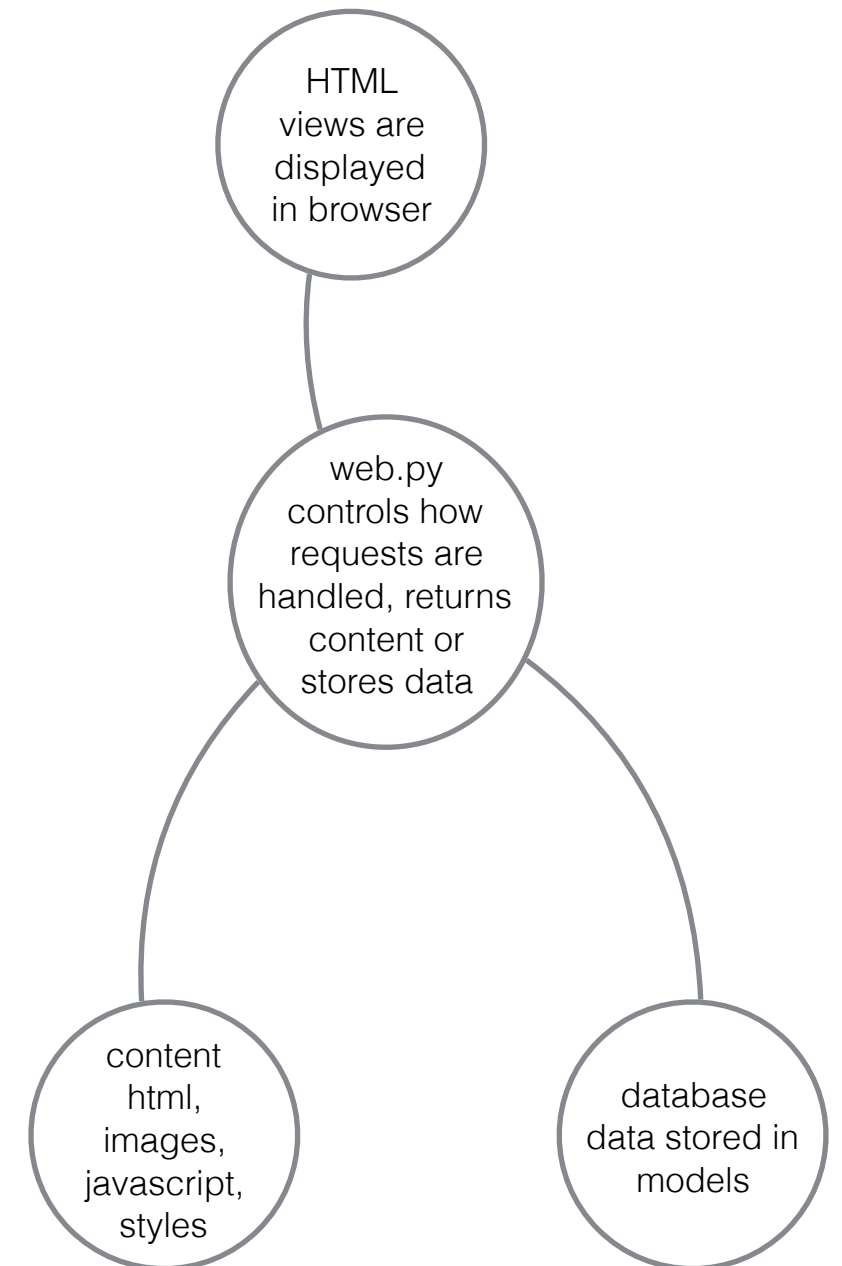
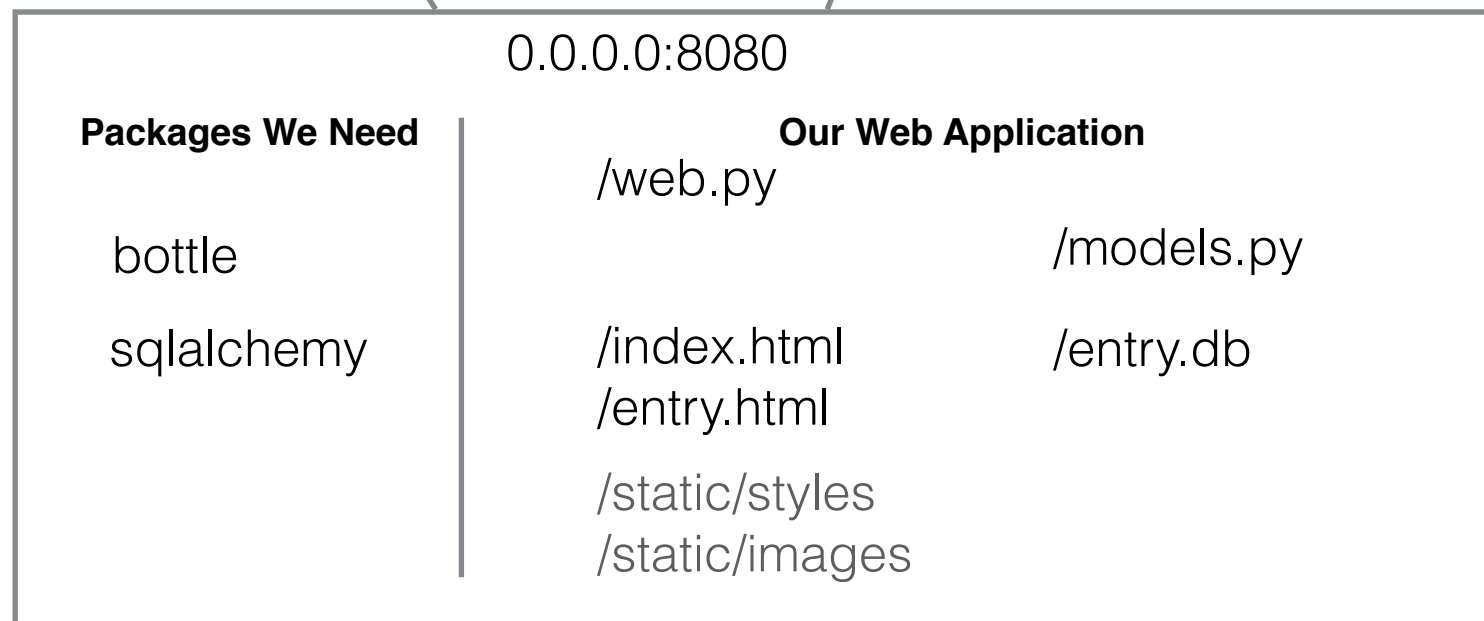
http://0.0.0.0:8080/entry

Heading

Date

Title

Entry



The steps we'll take

1. Create pages to display and collect entries

- create HTML file index.html
- web.py - three routes `/` and `/entry` and `/static`, using bottle
- style our page and add our `/static` content

2. Store entries in a database

- clean up and add to our HTML
- create models.py
- update web.py

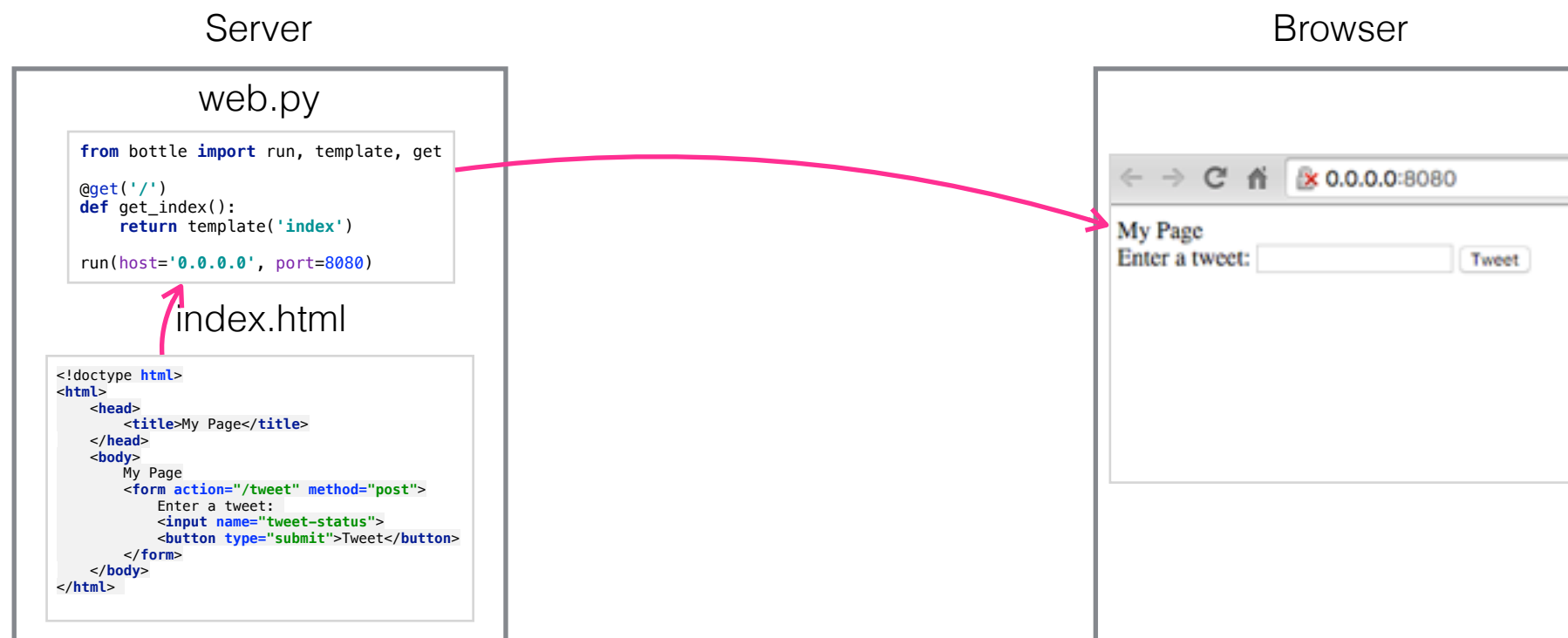
3. Display all of our entries

- update web.py
- update index.html

HTML

Hypertext Markup Language (HTML) is a language used to describe how a webpage looks. HTML is sent to a browser from a server. The browser uses the instructions in HTML to draw a page.

HTML only identifies visually how a page should look. It's descriptive. HTML does not have logic in it. To add logic you may add JavaScript to your HTML or add logic to your server.



HTML

HTML is a series of nested elements. Elements are identified by a start tag, content and an end tag. `<body> Hello! </body>`

Tags are not displayed on a page, they only describe how the content should be organized and displayed.

A few elements may be defined with only a start tag. These are called void tags. `<input>`
``

```
<!doctype html>
<html>
  <head>
    <title>My Page</title>
  </head>
  <body>
    My Page
    <form action="/tweet" method="post">
      Enter a tweet:
      <input name="tweet-status">
      <button type="submit">Tweet</button>
    </form>
  </body>
</html>
```

Elements

Document

<code><html></html></code>	Defines a document
<code><head></head></code>	Document information, not displayed
<code><body></body></code>	All displayed content

Spacing and Size

<code><p></p></code>	Defines a paragraph, creates vertical space
<code><h1></h1></code>	Heading, big text, h1(big) through h6(small)
<code>
</code>	Blank line, creates vertical space

Lists and Unformatted

<code> </code>	Unordered list, bullet points
<code> </code>	Ordered list, numbered items
<code><pre></pre></code>	Unformatted text, useful to show source code

Image, Link and Line

<code></code>	Display an image
<code><hr></code>	Horizontal line
<code> link</code>	Anchor or link

Form and Input

<code><form action="add" method="post"></form></code>	Defines form, contains form elements
<code><input type="text" name="username"></code>	Form element, capture text
<code><input type="submit"></code>	Form element, submit the form

Button and Select

<code><button type="submit"> Submit</button></code>	Form element, submit the form
<code><selection name="names"> </select></code>	Form element, defines dropdown
<code><option value="TN"> Tennessee</option></code>	Select element, defines a dropdown option

Getting Started

1. Create a new file called index.html.
2. Add a !doctype tag and the following elements, html, head, and body.
3. Inside the head element add a title element. Give your page a name inside title.

4. Inside the body element add the following:

- Page heading using an h2 element
- Add a p element, inside place instructions
- Add a form element, inside add a text input
- Also add a submit button and name it GO!

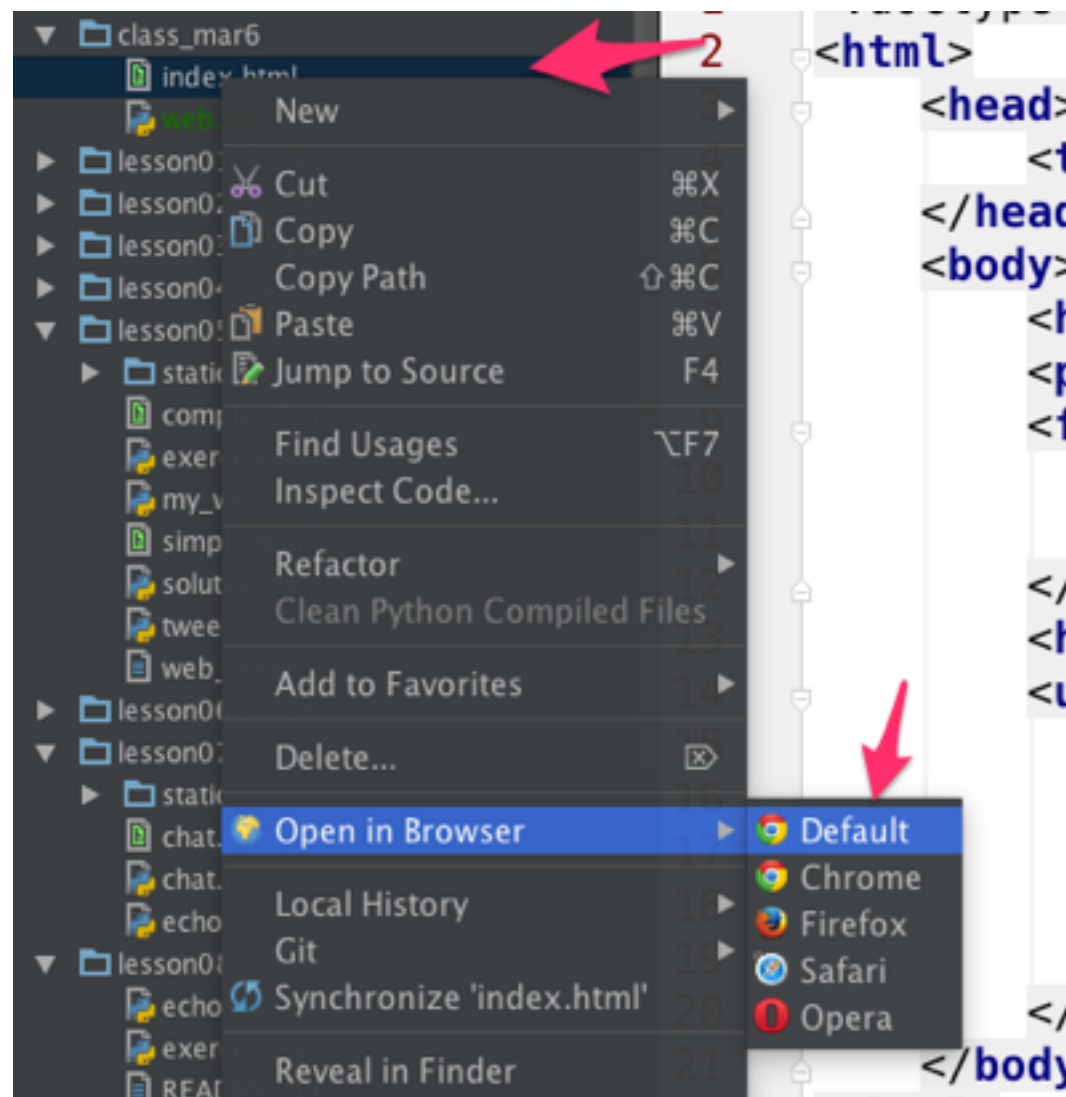
5. Back to body, add a horizontal line.

6. Add an unordered list of items. Invent some items.

```
<!doctype html>
<html>
  <head>
    <title>My Page</title>
  </head>
  <body>
    <h1>My Page</h1>
    <p>Enter your name</p>
    <form>
      <input type="text">
      <button type="submit">GO!</button>
    </form>
    <hr>
    <ul>
      <li>Scooby</li>
      <li>Shaggy</li>
      <li>Velma</li>
      <li>Daphne</li>
      <li>Fred</li>
    </ul>
  </body>
</html>
```

View Page

In PyCharm, you can view your new web page by choosing to open it in a browser.



Using Bottle.py

Bottle is small Python framework that we can use to build web applications.

1. Create a Python called web.py.

2. Import the bottle module like this `from bottle import get, template, run`

This tells Python to import the bottle module and that you would like to use the `get`, `template`, and `run` elements in your code.

3. Add a decorator, `@get('/')`

This decorator tells bottle to send all web requests for `/` to the next function.

4. Create a function and call it `get_index()`.

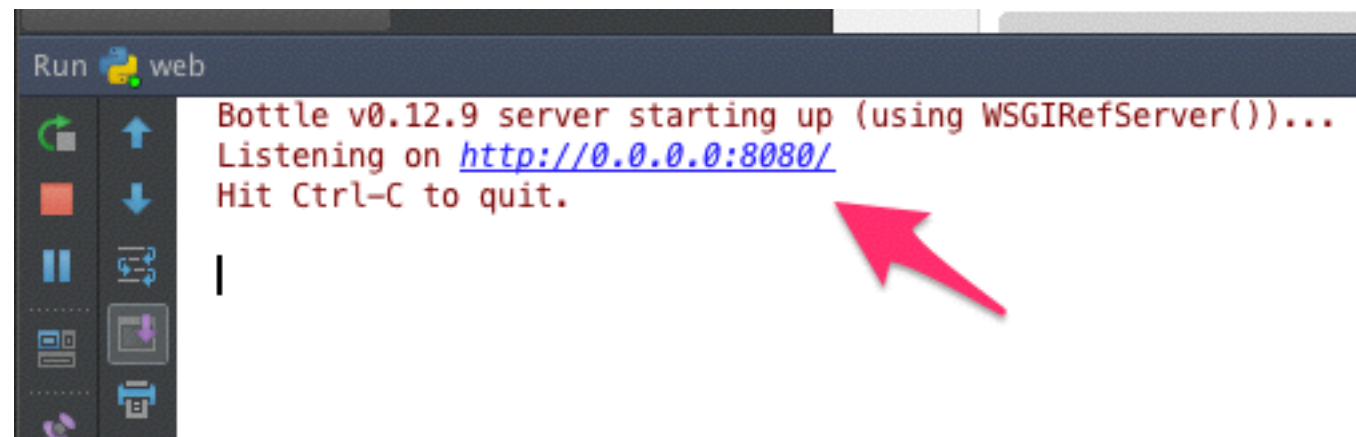
1. Inside `get_index()` create a variable called `web_page`. Assign `web_page` the value returned by `template('index')`, Function `template()` tells bottle to read your `index.html` file.

2. Return `web_page` from your function.

5. Outside of `get_index()` in the main block, call `run(host='0.0.0.0', port=8080)` notice that the parameters are named and notice that the host is a string and port is an integer.

View Page

In PyCharm, run web.py and click on the link in the console.



The image shows a screenshot of the PyCharm IDE's console window. The console title bar reads "Run web". The output text is as follows:

```
Bottle v0.12.9 server starting up (using WSGIRefServer())...  
Listening on http://0.0.0.0:8080/  
Hit Ctrl-C to quit.
```

A red arrow points to the URL <http://0.0.0.0:8080/> in the console output.

Change Page Style

To change colors, fonts, layout and other properties of your webpage elements use CSS Cascading Style Sheets.

```
<!doctype html>
<html>
  <head>
    <title>My Page</title>
  </head>
  <body>
    <h1>My Page</h1>
    <p>Enter your name</p>
    <form>
      <input type="text">
      <button type="submit">GO!</button>
    </form>
    <hr>
    <ul>
      <li>Scooby</li>
      <li>Shaggy</li>
      <li>Velma</li>
      <li>Daphne</li>
      <li>Fred</li>
    </ul>
  </body>
</html>
```



```
<!doctype html>
<html>
  <head>
    <title>My Page</title>
    <style type="text/css">
      body {
        color: white;
        background-color: darkblue;
      }
    </style>
  </head>
  <body>
    <h1>My Page</h1>
    <p>Enter your name</p>
    <form>
      <input type="text">
      <button type="submit">GO!</button>
    </form>
    <hr>
    <ul>
      <li>Scooby</li>
      <li>Shaggy</li>
      <li>Velma</li>
      <li>Daphne</li>
      <li>Fred</li>
    </ul>
  </body>
</html>
```

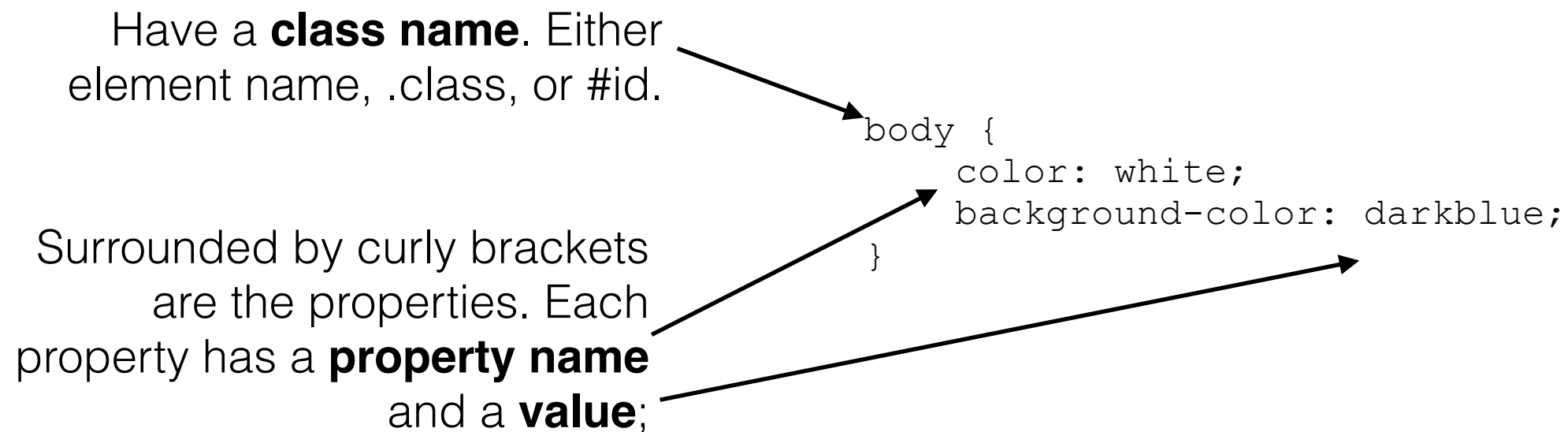


Styling an Element

There are three components that make up a CSS class.

Have a **class name**. Either element name, .class, or #id.

Surrounded by curly brackets are the properties. Each property has a **property name** and a **value**;



```
body {  
  color: white;  
  background-color: darkblue;  
}
```

The diagram shows three arrows originating from the explanatory text on the left and pointing to specific parts of the CSS code on the right. The first arrow points from 'Have a class name' to 'body'. The second arrow points from 'Surrounded by curly brackets' to the opening and closing curly braces of the CSS rule. The third arrow points from 'Each property has a property name and a value' to the 'background-color: darkblue;' line.

Element Name

`<body></body>`

```
body {  
  color: white;  
  background-color: darkblue;  
}
```

Style Class

`<p class="do-blue">Hello!</p>`

```
.do-blue {  
  color: white;  
  background-color: darkblue;  
}
```

Element Id

`<p id="first-name">John</p>`

```
#first-name {  
  color: white;  
  background-color: darkblue;  
}
```

Properties

Color and Background

color	Font color
background-color	Color of background
background-image	Image posted to background

Font

font-family	serif, sans-serif, monospace
font-size	px, pt, em, %
font-weight	normal, bold, 100-900
font-style	normal, italics

Width and Height

width, height	px, %, auto
max-height max-width	cannot be larger than
min-height min-width	cannot be smaller than

Margin and Padding

margin	space around outside of element border
padding	space around inside of element border
margin-left padding-left	top, bottom, left, right

Border

border-width	thickness of line, px
border-style	solid, dotted, dashed
border-color	color of border
border	1px solid red

Links <a>

link, visited	color for links not active
hover, active	colors for links hovered or being clicked
text-decoration	none, underline

Change Your Page

1. Add a style element to your head element.
2. The style element should have the type="text/css" attribute.
3. Identify the body element inside the style element.
4. Style the body to change color and background.
5. Pick other elements and try changing the style.