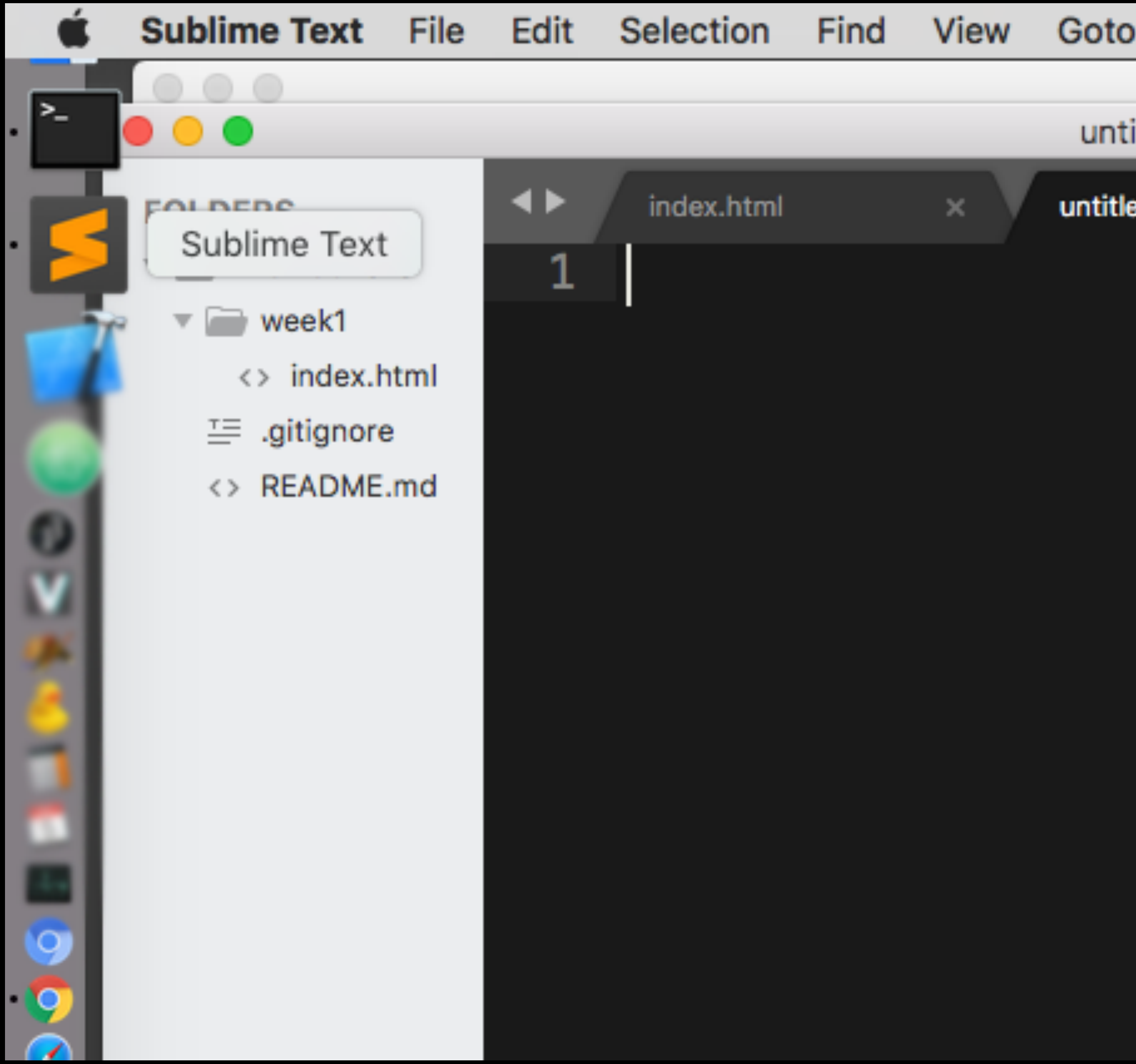
web pages are made
of three different file types
that we can author at the
granular level

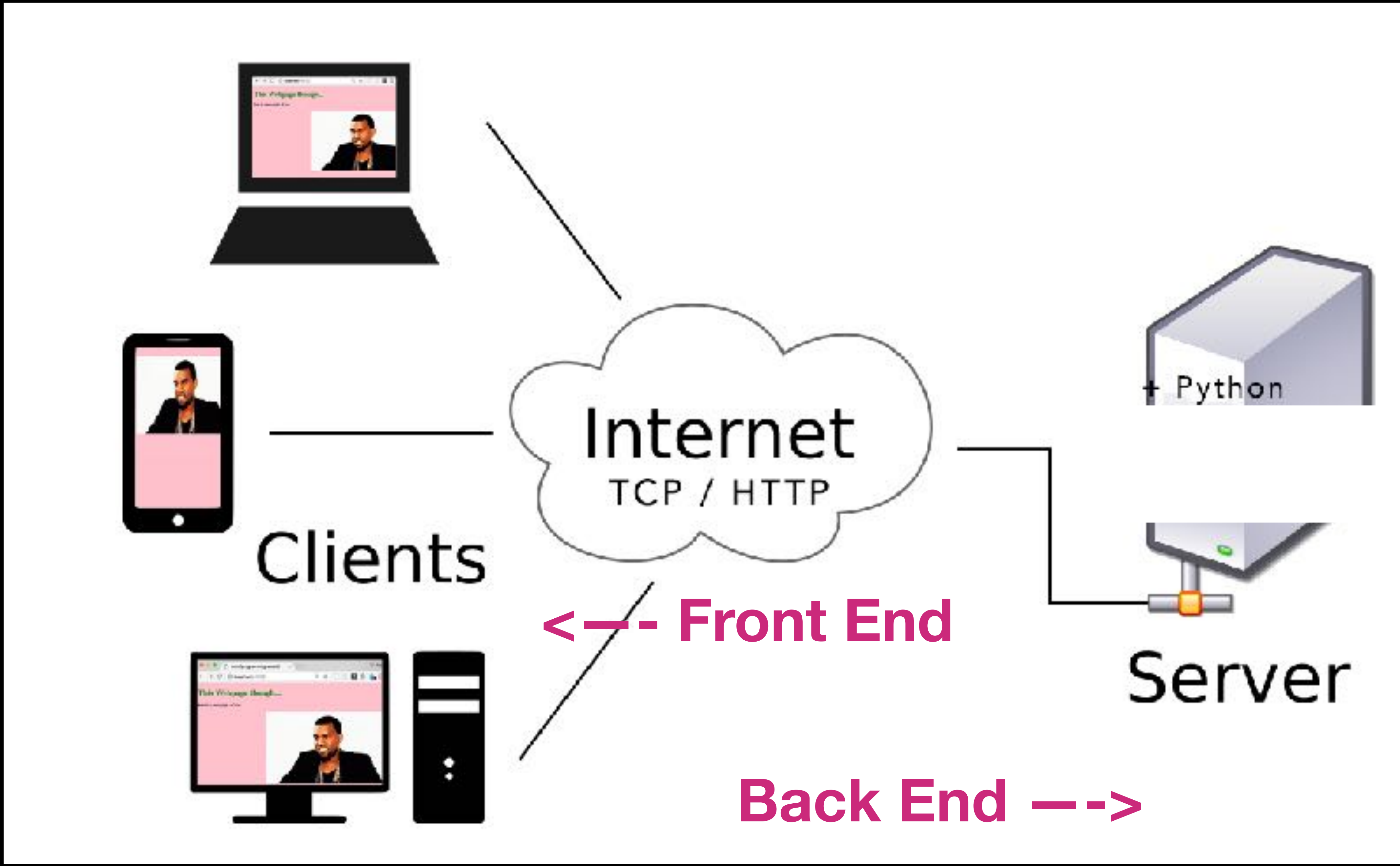
.html	hyper text mark up
.css	cascading style sheet
.js	javascript

We can write these files with a text editor. Like **Sublime**. There are many others but this one is great for those new to writing code or to web development.

WebDev WorkFlow – We will be hosting local servers on our machines to prototype websites. Mac OS users will become familiar with **Terminal** + running a local python server. Those using Windows Machines **Command Prompt** isn't quite as easy - I am not fluent in the various dialects. But we'll figure it out as we go…



TCP ports



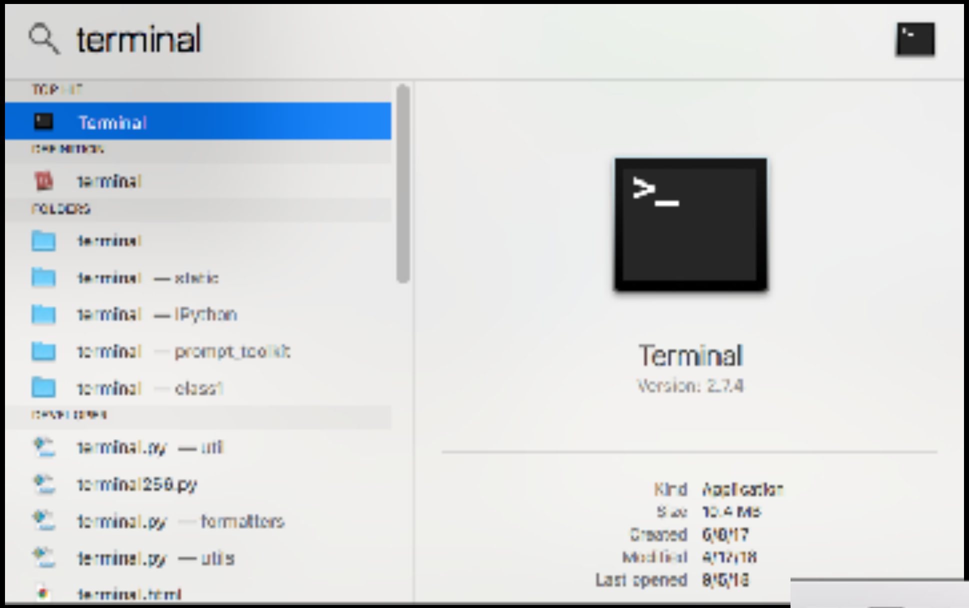
Even when programming for FRONT END WEB DEV —> Running a local server while working is best practice

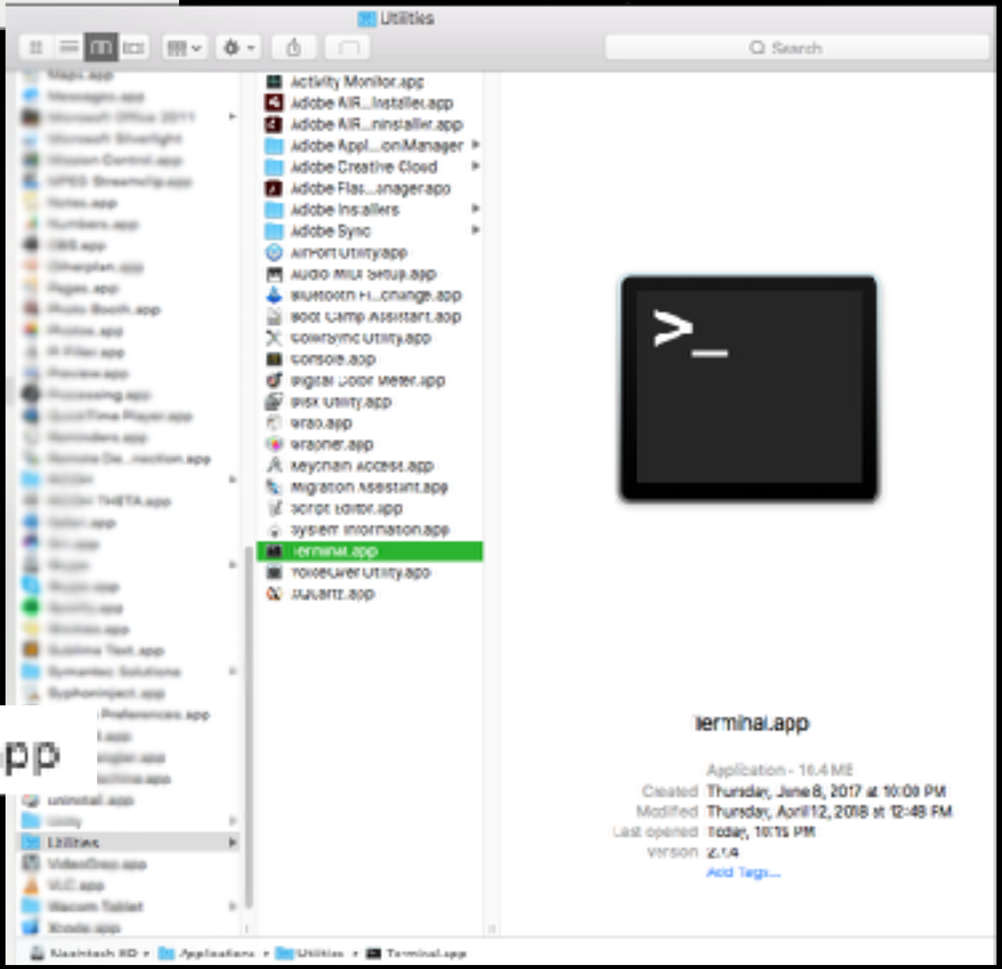
This will become ESSENTIAL when we get into JavaScript

Terminal is where we can interact w/ our Macs without using the OS and GUI. This is useful for WebDev but **BE CAREFUL**. Do not type in a command if yr note sure what it will do.

for those who like to search:

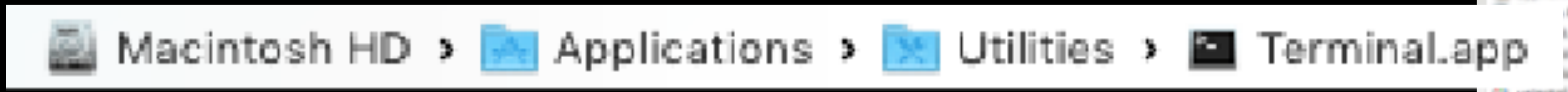
Hold down **COMMAND** + press **SPACEBAR**, and type **TERMINAL**





for those who like to find:

the file path is:



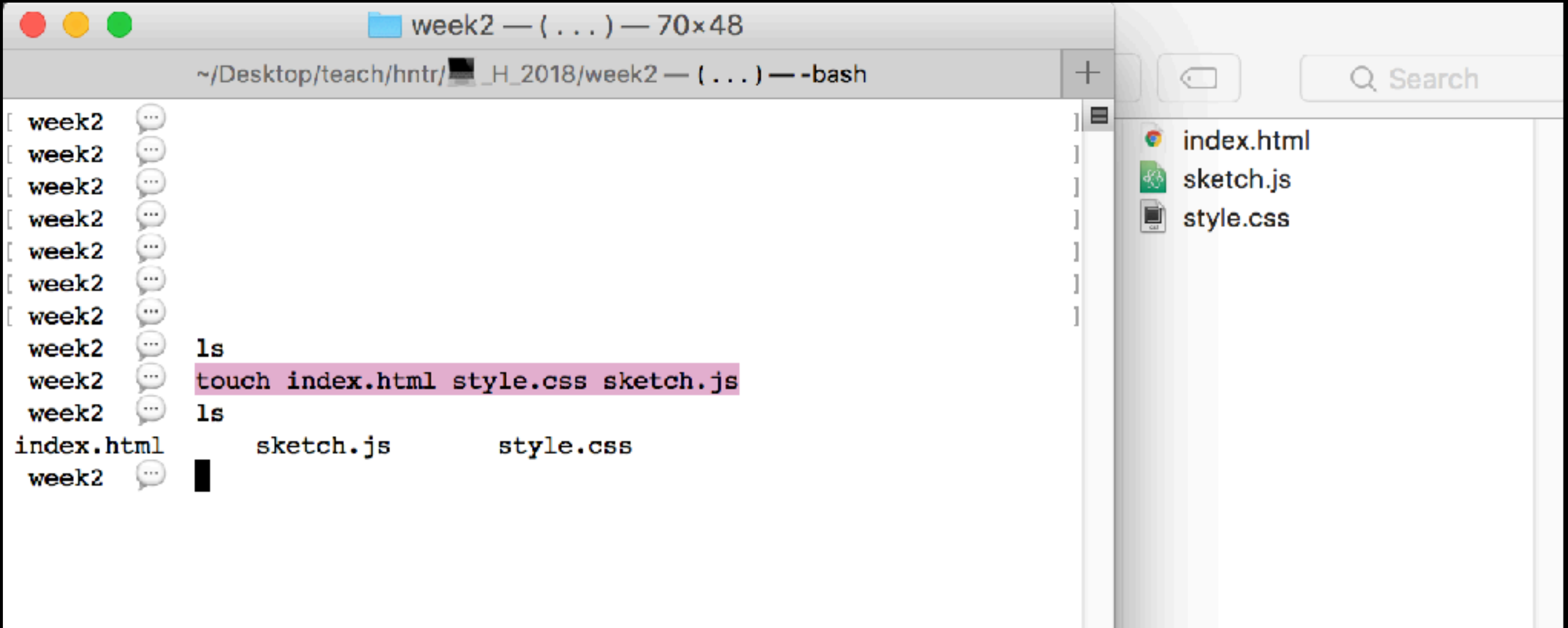
Macintosh HD › Applications › Utilities › Terminal.app

This is useful for WebDev but **BE CAREFUL**. Do not type in a command if yr note sure what it will do.

in Terminal we are speaking **Unix** :

Some useful commands:

- **cd** - "change directory"
- **ls** - "list items in this directory"
- **pwd** - "present working directory"
- **touch** - "create file"


week2 — (. . .) — 70×48
~/Desktop/teach/hntr/ _H_2018/week2 — (. . .) — -bash
Search
index.html
sketch.js
style.css

```
week2
week2
week2
week2
week2
week2
week2
week2      ls
week2      touch index.html style.css sketch.js
week2      ls
index.html      sketch.js       style.css
week2
```

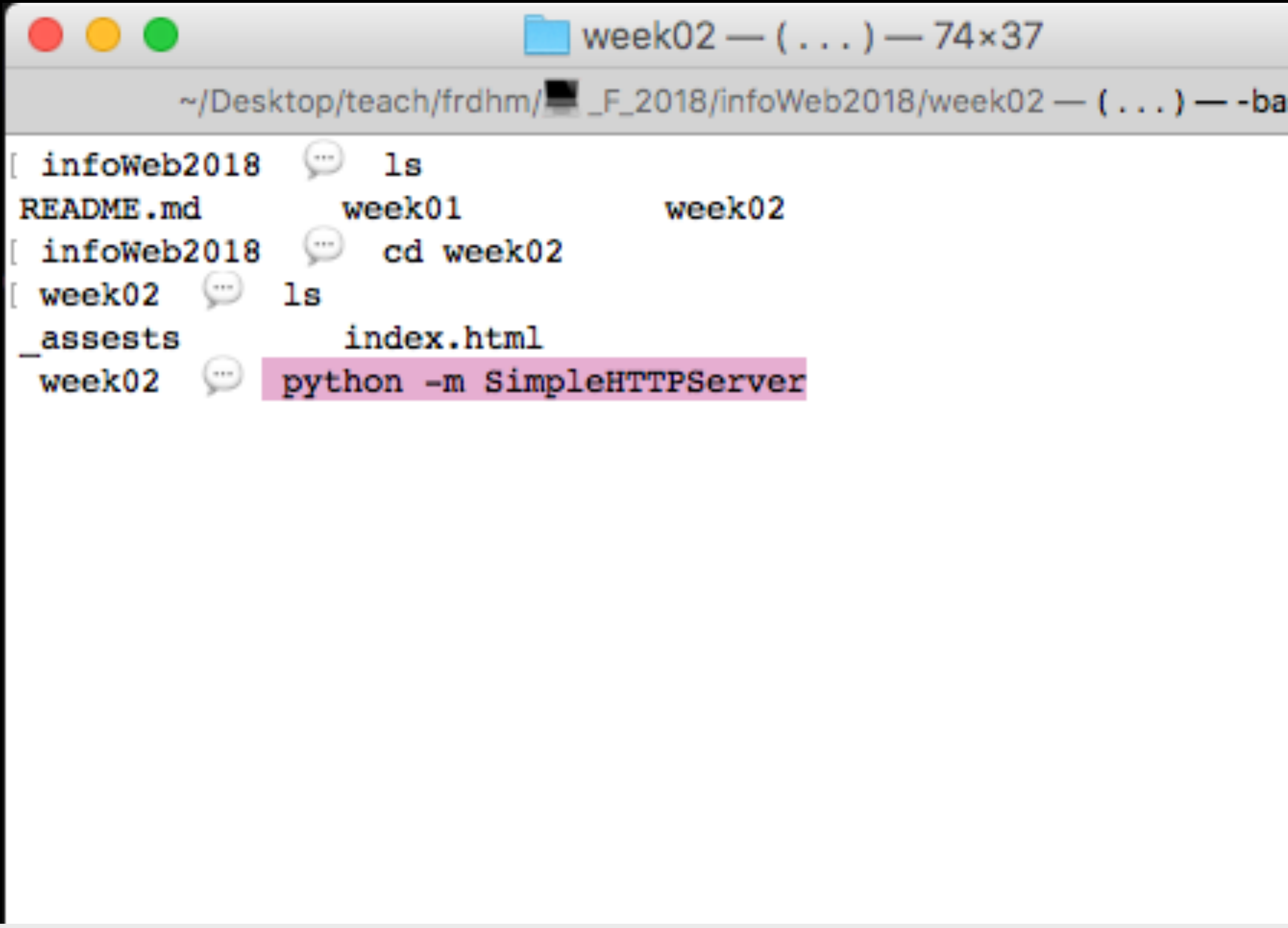
- **cd** - “change directory”
- **ls** - “list items in this directory”
- **pwd** - “present working directory”

Running a local **Python** HTTP Server
in Mac OS - this is very simple :
When inside yr project folder simply
type the following command:

“**python -m SimpleHTTPServer**”

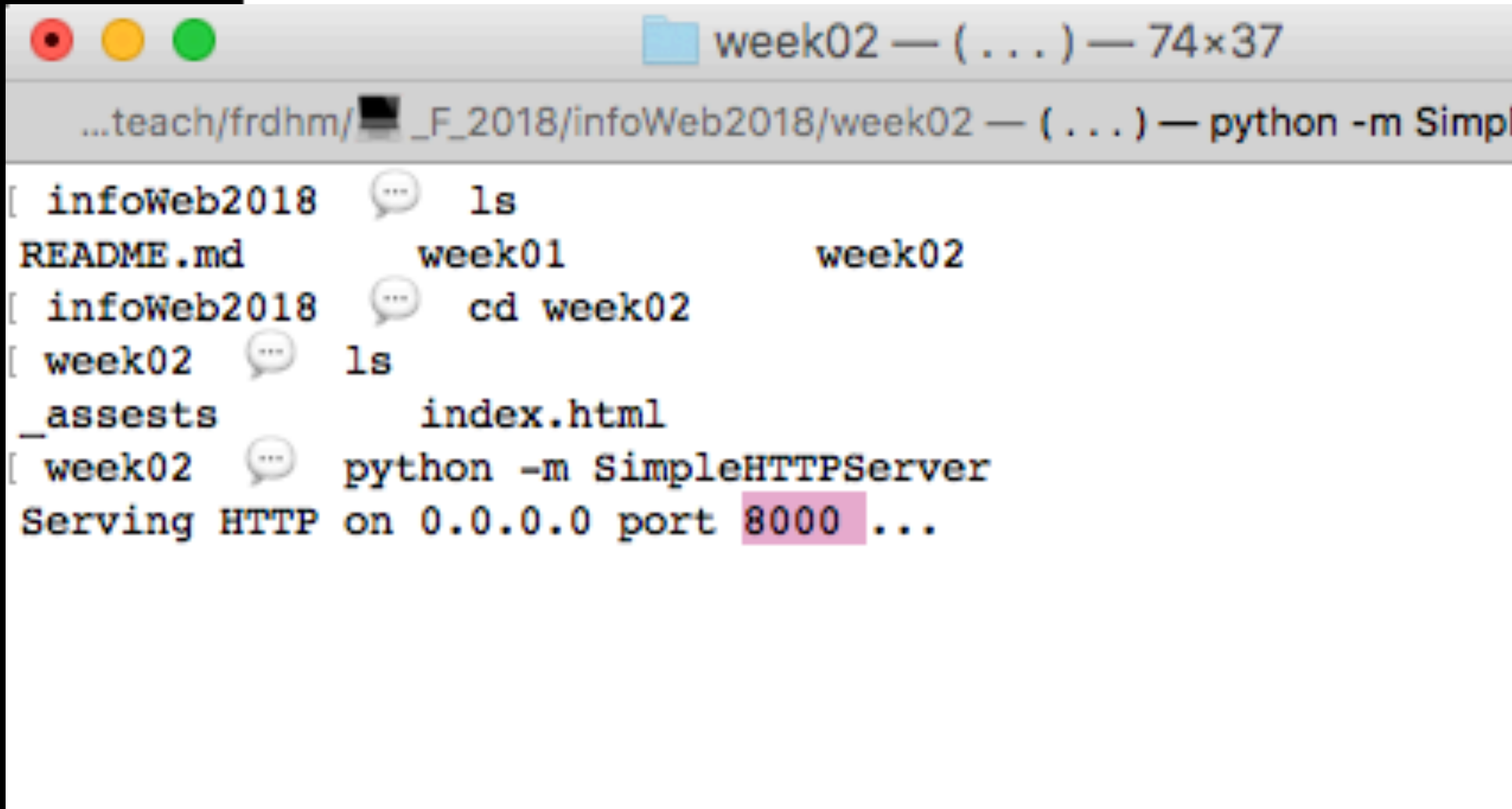
– defaults to port 8000

if we wrote:
“python -m SimpleHTTPServer 12345”
- we would go to port 12345


week02 — (. . .) — 74×37

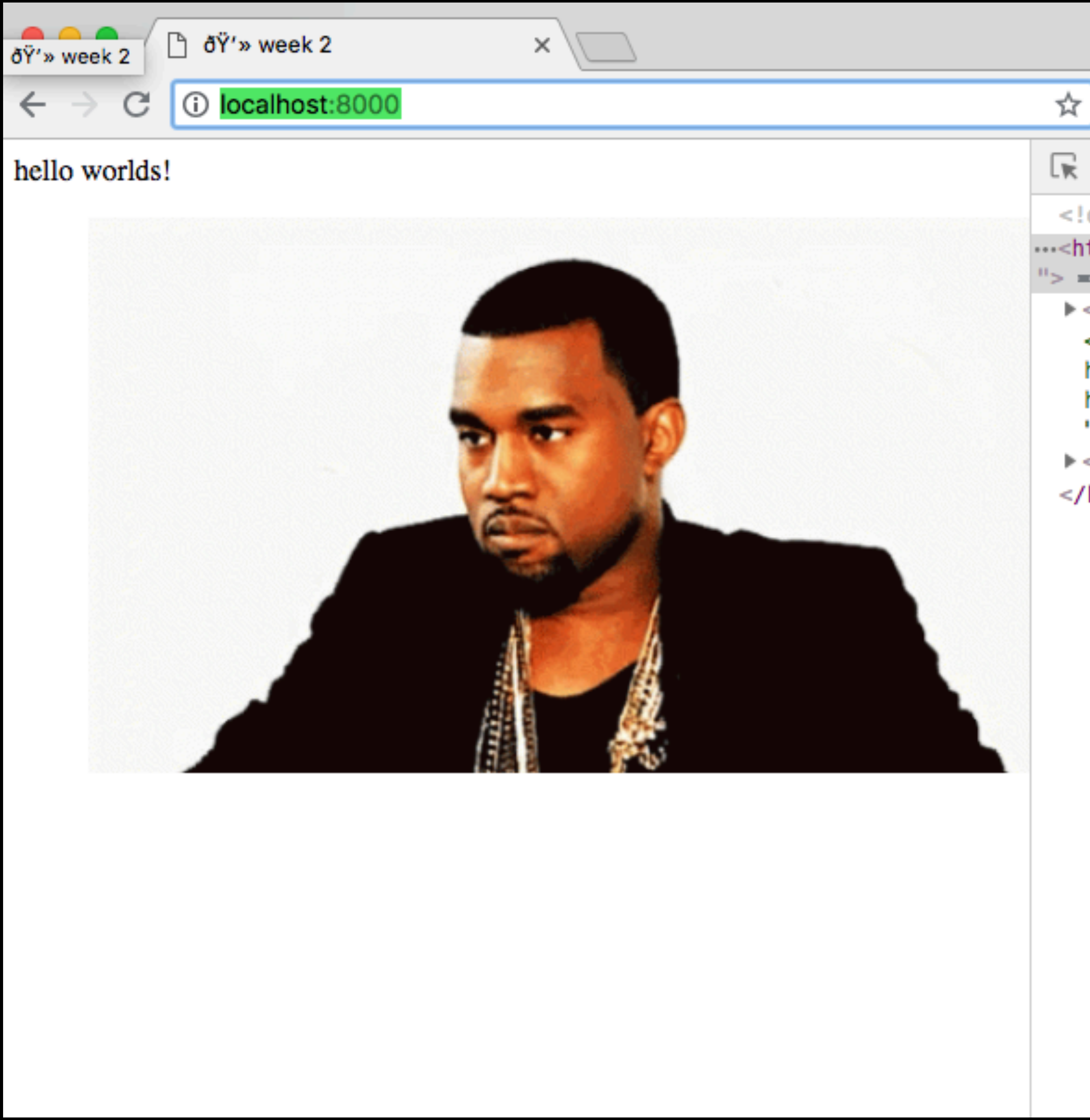
~/Desktop/teach/frdhm/ _F_2018/infoWeb2018/week02 — (. . .) — -ba

```
infoWeb2018     ls
README.md        week01          week02
infoWeb2018     cd week02
week02     ls
_assests         index.html
week02     python -m SimpleHTTPServer
```


week02 — (. . .) — 74×37

...teach/frdhm/ _F_2018/infoWeb2018/week02 — (. . .) — python -m Simp

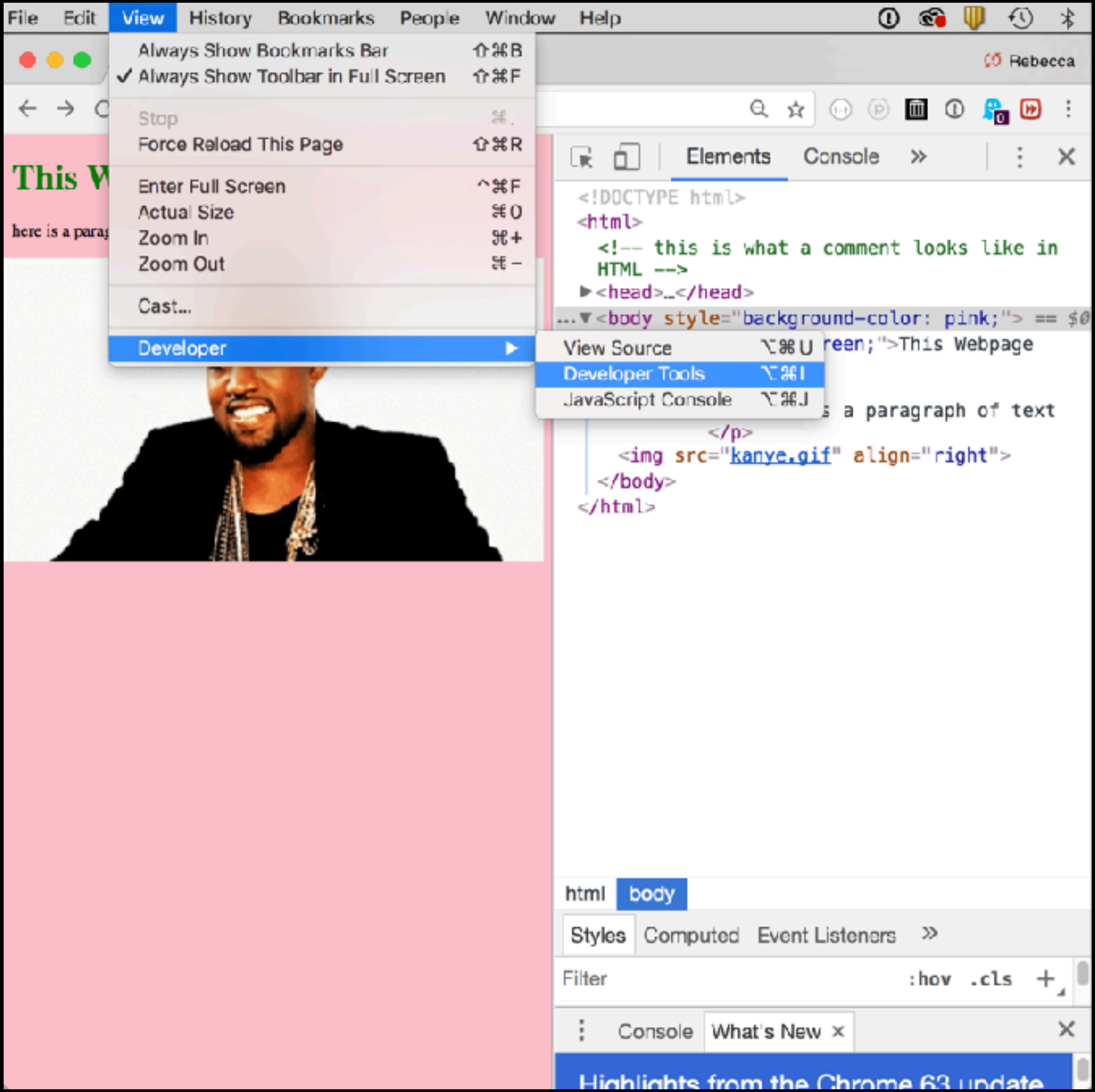
```
infoWeb2018     ls
README.md        week01          week02
infoWeb2018     cd week02
week02     ls
_assests         index.html
week02     python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```



hello worlds!

url is:

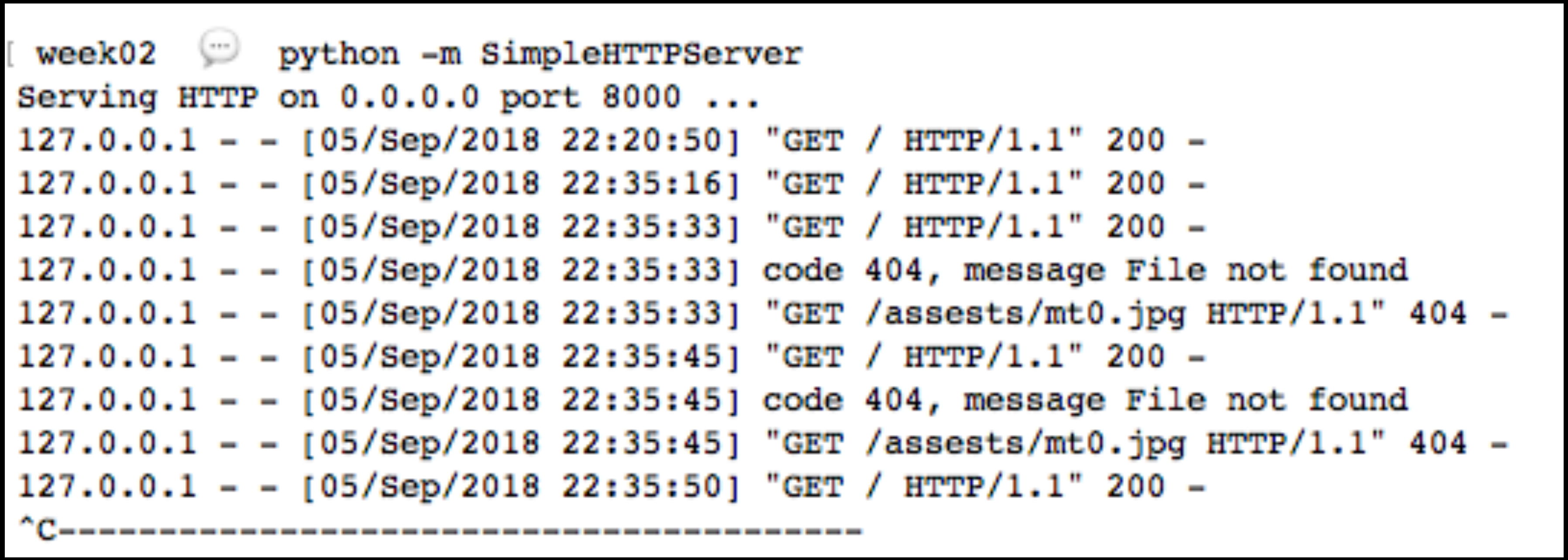
localhost:8000



```
<!DOCTYPE html>
<html>
  <!-- this is what a comment looks like in
  HTML -->
▶<head>…</head>
…▼<body style="background-color: pink;"> == $0
              reen;">This Webpage
              s a paragraph of text
        </p>
    <img src="kanye.gif" align="right">
  </body>
</html>
```

+ **Google Chrome** Browser
+ Dev Tools (cmmd i)

As you make changes to your design / code - you can "live" refresh the page, changes (+ bugs) will be noted by the server.



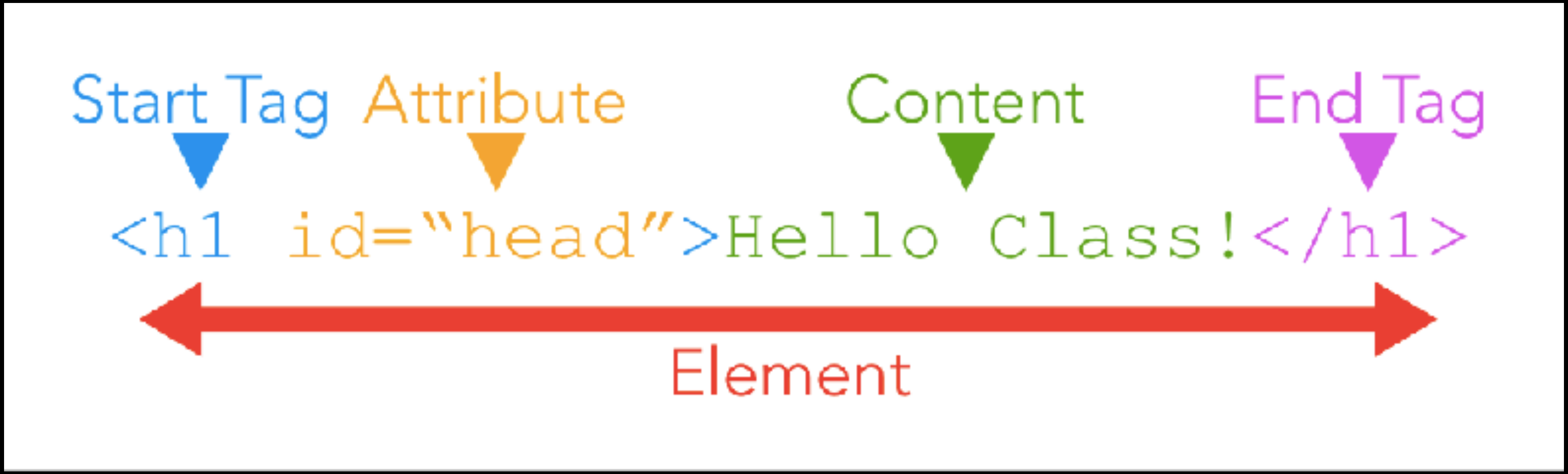
```
[ week02     python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
127.0.0.1 - - [05/Sep/2018 22:20:50] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Sep/2018 22:35:16] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Sep/2018 22:35:33] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Sep/2018 22:35:33] code 404, message File not found
127.0.0.1 - - [05/Sep/2018 22:35:33] "GET /assests/mt0.jpg HTTP/1.1" 404 -
127.0.0.1 - - [05/Sep/2018 22:35:45] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Sep/2018 22:35:45] code 404, message File not found
127.0.0.1 - - [05/Sep/2018 22:35:45] "GET /assests/mt0.jpg HTTP/1.1" 404 -
127.0.0.1 - - [05/Sep/2018 22:35:50] "GET / HTTP/1.1" 200 -
^C----------------------------------------
```

** Press "Control" + "C" to end the server session.

(Otherwise it's the equivelent to unplugging a hard drive w/ out "ejecting it" - BAD PRACTICE. As DIGITAL CITIZENS - we 💚 our hardware + software…)

HTML Elements / Tags, Attributes, Content

- Elements and tags used interchangeably



```
<h1 id="head">Hello Class!</h1>
```

The <head> element contains the metadata for a web page. Metadata is information about the page that isn't displayed directly on the web page. Unlike the information inside of the <body> tag, the metadata in the head is information about the page itself.

Structure tags

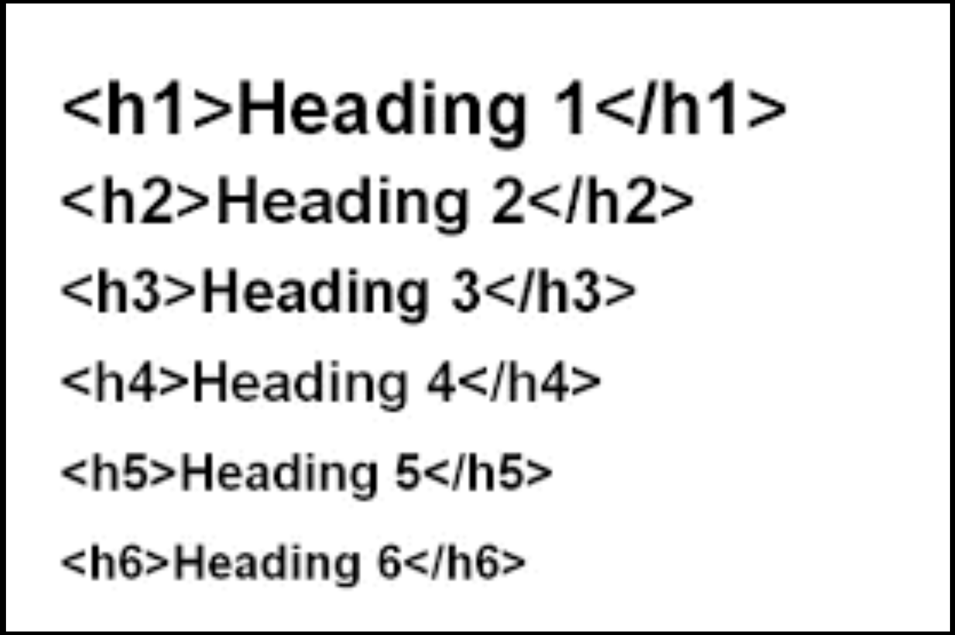
```
<!doctype html>
      <head>
         <title> Week 3 </title>
      </head>
      <body>
         <div>
         Here’s a Great Site.
         </div>
      <body>
</html>
```

Parent + Child

```
                                <!doctype html>
                                      <head>
head is the parent of title               <title> Week 3 </title>
                                      </head>
                                      <body>
div is the child of body                  <div>
                                          Here's a Great Site.
body is the child of html                 </div>
                                      </body>
                                </html>
```

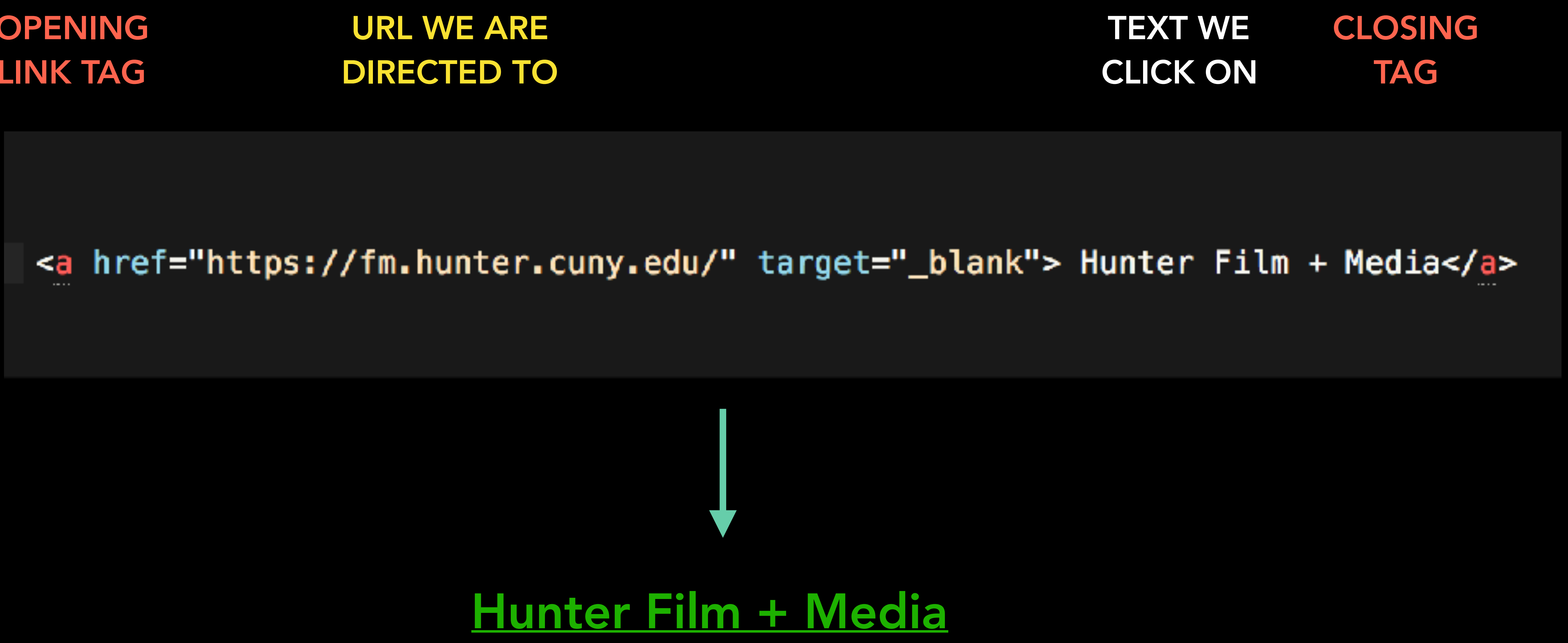
Text tags

- h1, h2, h3, h4, h5, h6 are text tags for headings
- p is a tag for paragraphs
- b is for bold, i is for italics
- <strong> is for **bold** <em> is for *italics*
- ul, ol, li are used for making lists
  - ul: unordered lists
  - ol: ordered lists
  - li: an individual list tag
- <br/> will break to a new line



```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Structure of a link



```
<a href="https://fm.hunter.cuny.edu/" target="_blank"> Hunter Film + Media</a>
```

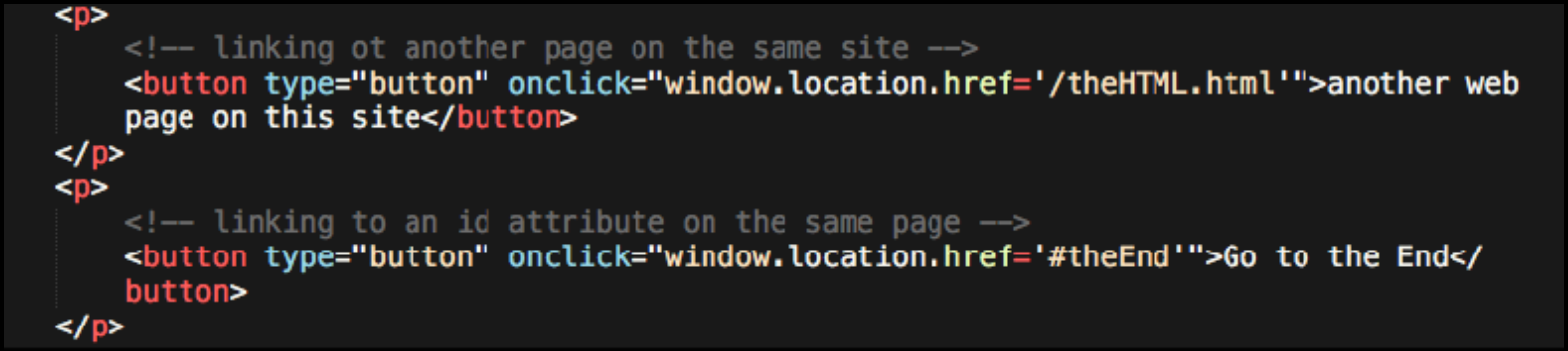
< a href — stands for *hyperlink reference*

Linking to pages on the same site

RELATIVE URLS

Link types:

- parent folder: <a href="../index.html">Homepage</a>
  - same folder: <a href="/index.html">Homepage</a>
    - child folder: <a href="images/photos.html">Photos</a>
    - id attribute: <a href="#thisID">Different element on page</a>



```
<p>
    <!-- linking ot another page on the same site -->
    <button type="button" onclick="window.location.href='/theHTML.html'">another web
    page on this site</button>
</p>
<p>
    <!-- linking to an id attribute on the same page -->
    <button type="button" onclick="window.location.href='#theEnd'">Go to the End</
    button>
</p>
```

The <img> tag has a required attribute called src. The src attribute must be set to the image's source, or the location of the image. In some cases, the value of src must be the *uniform resource locator* (URL) of the image. A URL is the web address or local address where a file is stored.

Images: Local vs. URL

- The **<img>** tag is for images, which can be on your local directory or on another webpage. Read all about **<img>** tag here

```
<!-- An image on the local directory -->
<img src="images/superCuteDog.jpg" alt="Cutest dog">

<!-- Or with size specs -->
<img src="images/superCuteDog.jpg" alt="Cutest dog" width="400" height="300">

<!-- Image from another site -->
<img src="http://www.petguide.com/wp-content/uploads/2014/10/cutest-dog-breeds-main.jpg"
alt="Not so cute dog">
```

<div> Division </div>

```
<body>
  <div>
    <h1>Why use divs?</h1>
    <p>Great for grouping elements!</p>
  </div>
</body>
```

<div>s can contain any text or other HTML elements, such as links, images, or videos. Remember to always add two spaces of indentation when you nest elements inside of <div>s for better readability.

Attributes

If we want to expand an element's tag, we can do so using an attribute. Attributes are content added to the opening tag of an element and can be used in several different ways, from providing information to changing styling. Attributes are made up of the following two parts:

1) The name of the attribute
2) The value of the attribute

One commonly used attribute is the id.

We can use the id attribute to specify different content (such as <div>s) and is really helpful when you use an element more than once.

```
<div id="intro">
   <h1>Technology</h1>
</div>
```

<span> contains short pieces of text or other HTML. They are used to separate small pieces of content that are on the same line as other content.

```
<div>
   <h1>Technology</h1>
</div>
<div>
   <p> Wherever there's a
   <span>computer</span>, there's a skilled
   person developing, maintaining, hacking,
   advancing or simply using it.</p>
</div>
```

The <em> tag will generally render as *italic* emphasis.

The <strong> will generally render as bold **emphasis.**

<br>

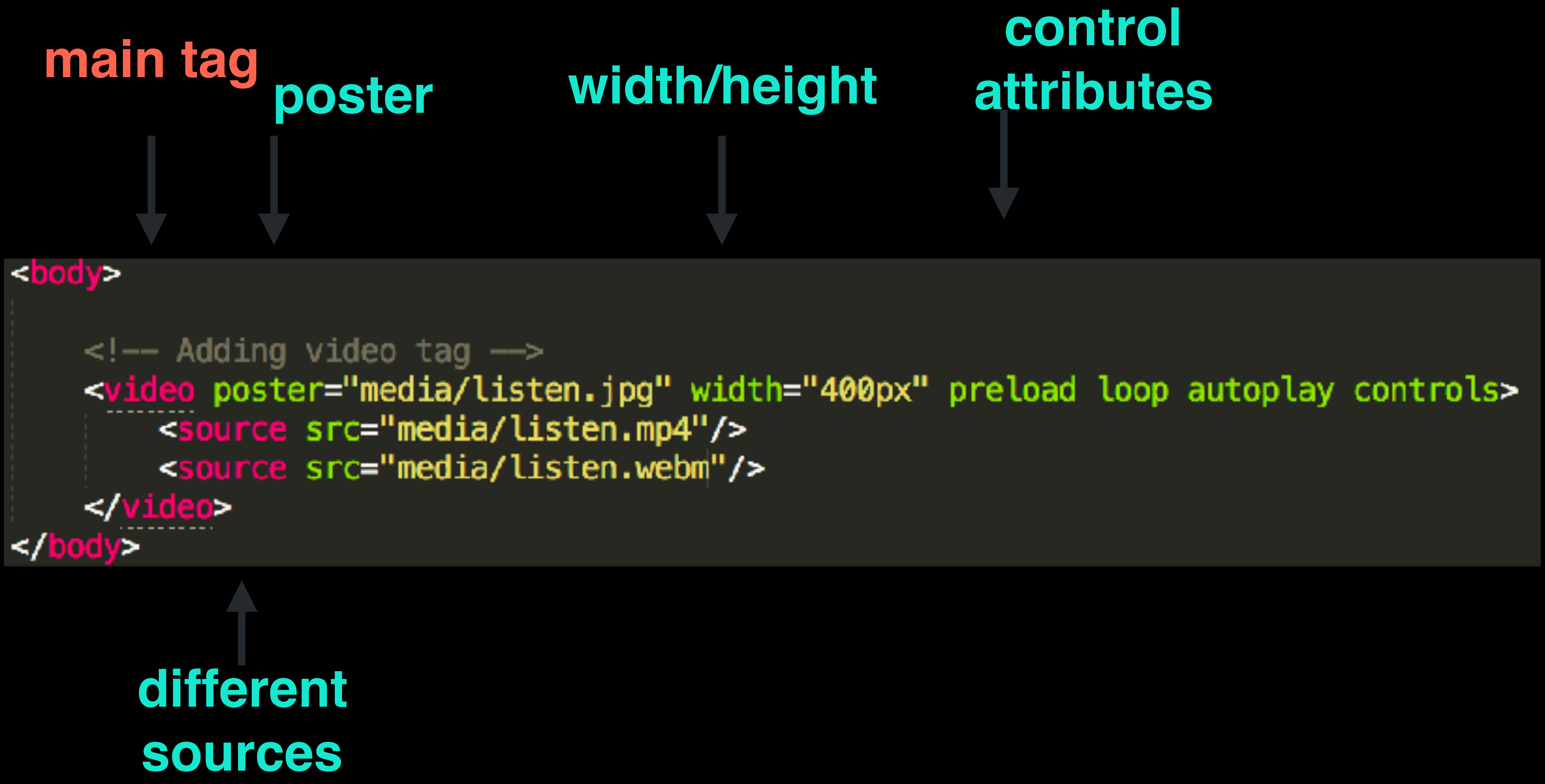
The line break element is unique because it is only composed of a starting tag. You can use it anywhere within your HTML code and a line break will be shown in the browser.

The alt attribute, which means alternative text, brings meaning to the images on our sites. The alt attribute can be added to the image tag just like the src attribute. The value of alt should be a description of the image.

1. If an image fails to load on a web page, a user can mouse over the area originally intended for the image and read a brief description of the image. This is made possible by the description you provide in the alt attribute.
2. Visually impaired users often browse the web with the aid of screen reading software. When you include the alt attribute, the screen reading software can read the image's description out loud to the visually impaired user.
3. The alt attribute also plays a role in Search EngineOptimization (SEO), because search engines cannot "see" the images on websites as they crawl the internet. Having descriptive alt attributes can improve the ranking of your site.

In addition to images, HTML also supports displaying videos. Like the <img> tag, the <video> tag requires a src attribute with a link to the video source. Unlike the <img> tag however, the <video> element requires an opening and a closing tag.

<video /> structure


main tag
poster
width/height
control attributes
different sources

```
<body>

    <!-- Adding video tag -->
    <video poster="media/listen.jpg" width="400px" preload loop autoplay controls>
        <source src="media/listen.mp4"/>
        <source src="media/listen.webm"/>
    </video>
</body>
```

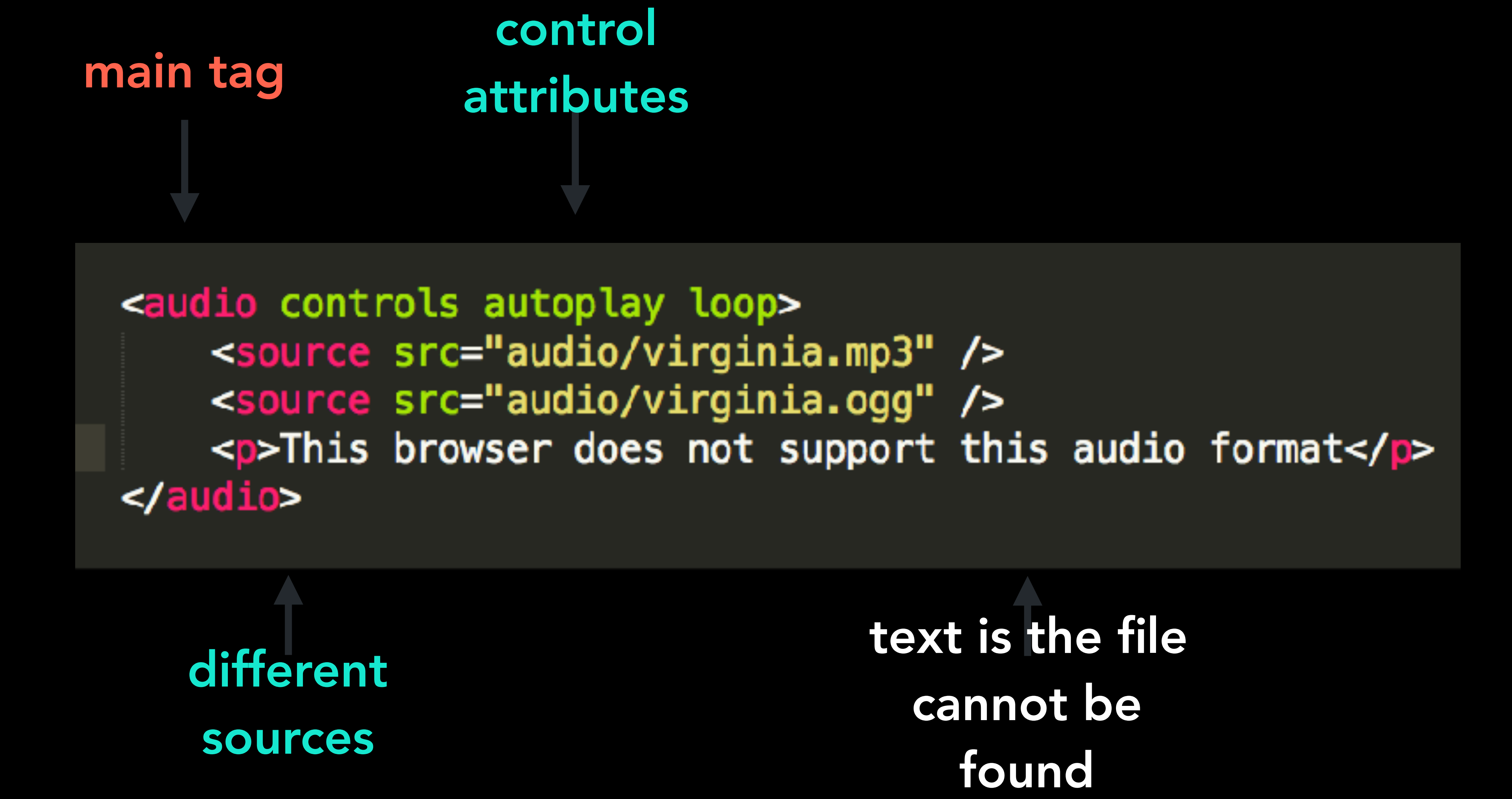
After the src attribute, the width and height attributes are used to set the size of the video displayed in the browser.
The controls attribute instructs the browser to include basic video controls: pause, play and skip.Unlike the <img> tag however, the <video> element requires an opening and a closing tag.

The text, "Video not supported", between the opening and closing video tags will only be displayed if the browser is unable to load the video.

Attributes

- Check out all the things you can control when using video
- For now, we'll look into these attributes:
  - Preload - what preloads when the page loads
  - Controls - if the play/stop buttons are visible
  - Autoplay - if the video should start playing automatically
  - Loop - if the video should loop on completion

<audio /> structure

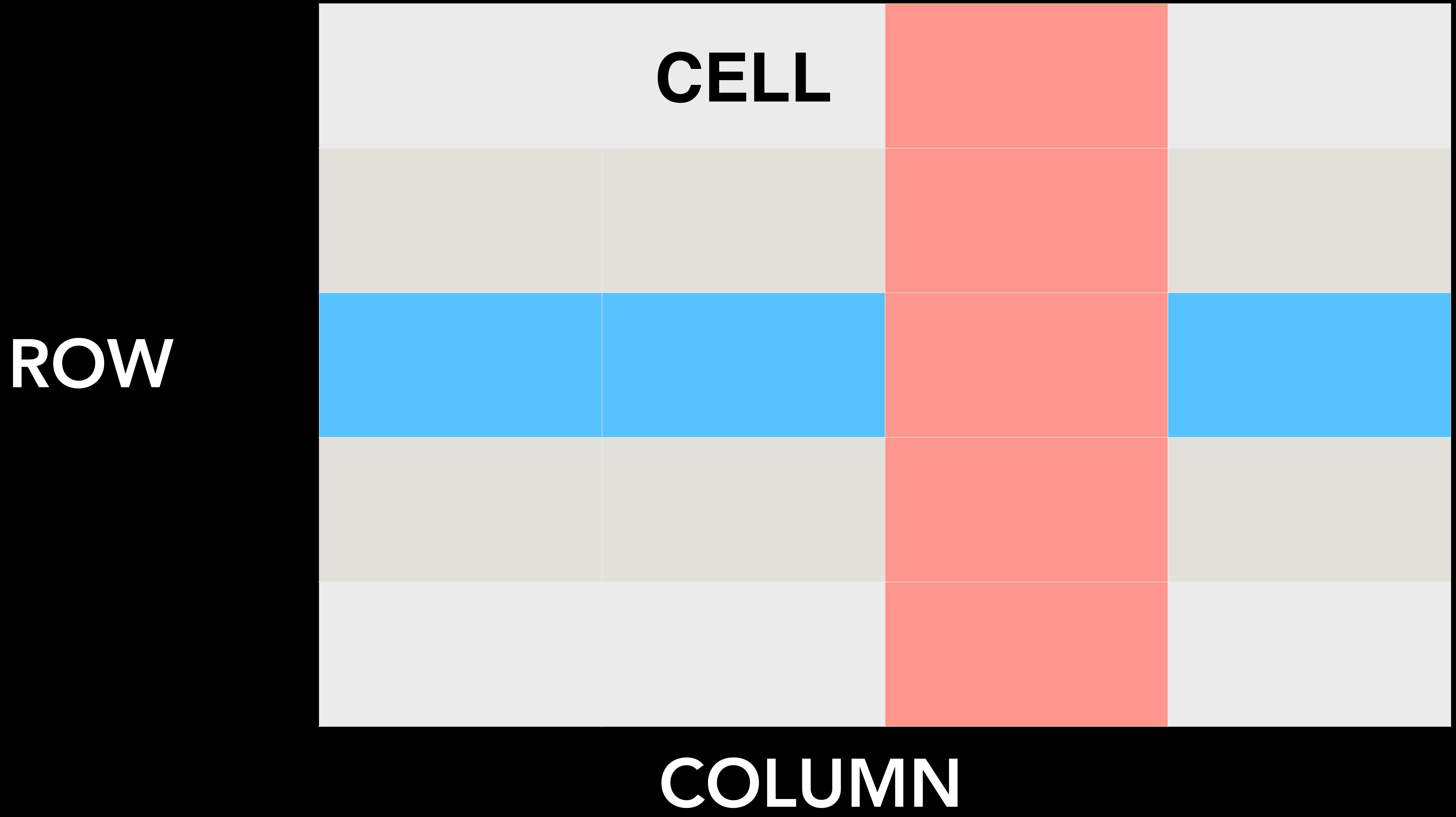


```
<audio controls autoplay loop>
    <source src="audio/virginia.mp3" />
    <source src="audio/virginia.ogg" />
    <p>This browser does not support this audio format</p>
</audio>
```

Attributes

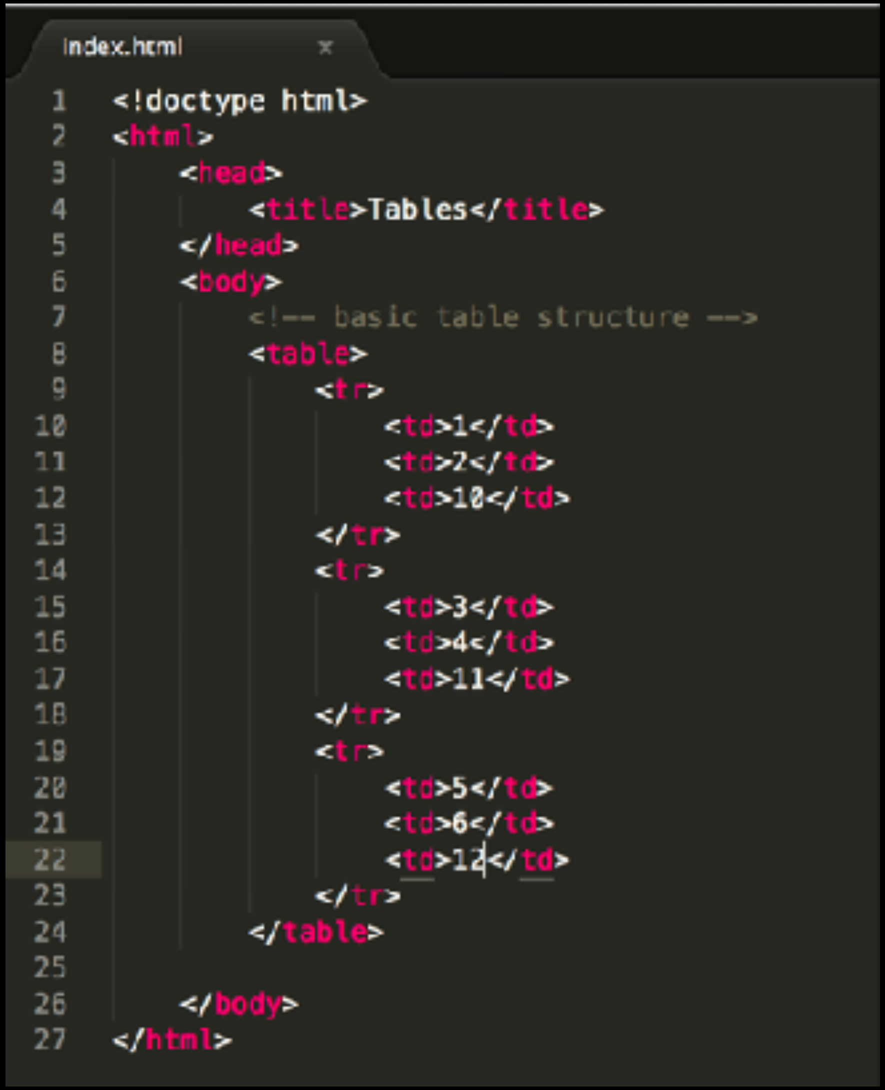
- **Preload** - what preloads when the page loads
- **Controls** - if the play/stop buttons are visible
- **Autoplay** - if the video should start playing automatically
- **Loop** - if the video should loop on completion

Table structure



Basic table structure

- <table> element is used to create a table (written out row by row)
- <tr> indicates each row
- <td> indicates each cell of a table



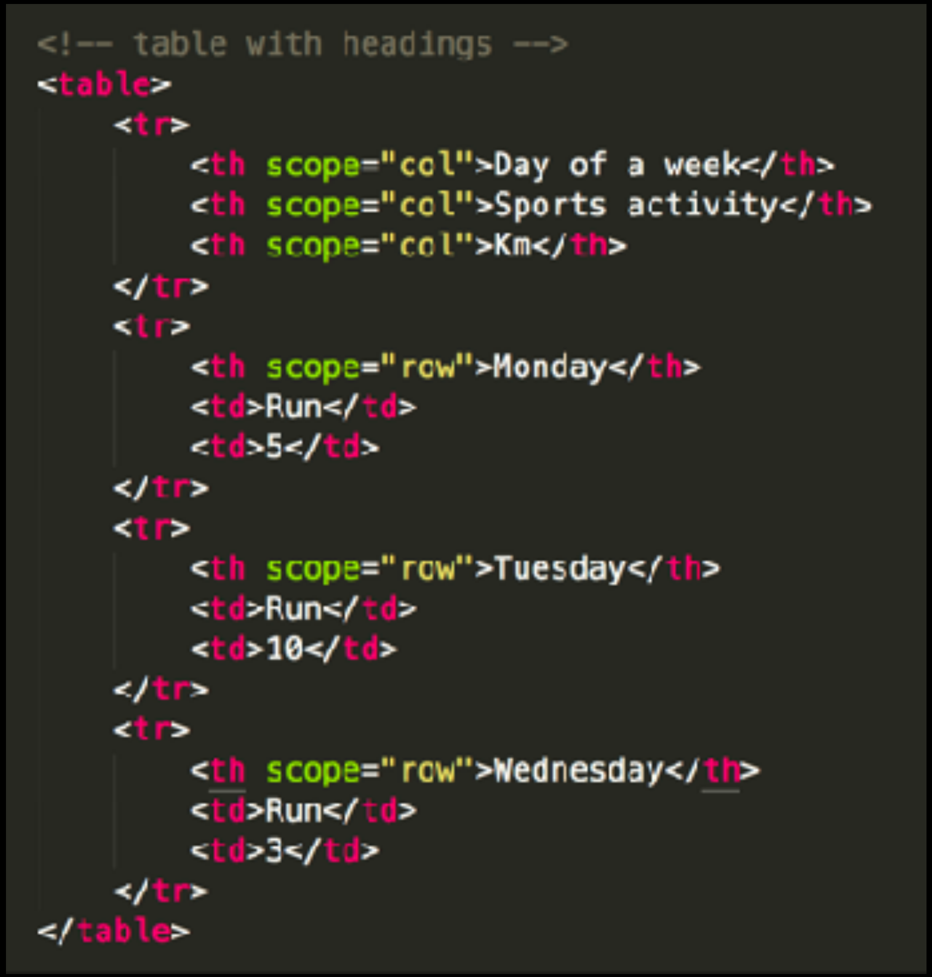
index.html

```html
<!doctype html>
<html>
    <head>
        <title>Tables</title>
    </head>
    <body>
        <!-- basic table structure -->
        <table>
            <tr>
                <td>1</td>
                <td>2</td>
                <td>10</td>
            </tr>
            <tr>
                <td>3</td>
                <td>4</td>
                <td>11</td>
            </tr>
            <tr>
                <td>5</td>
                <td>6</td>
                <td>12</td>
            </tr>
        </table>

    </body>
</html>
```

Adding table headings

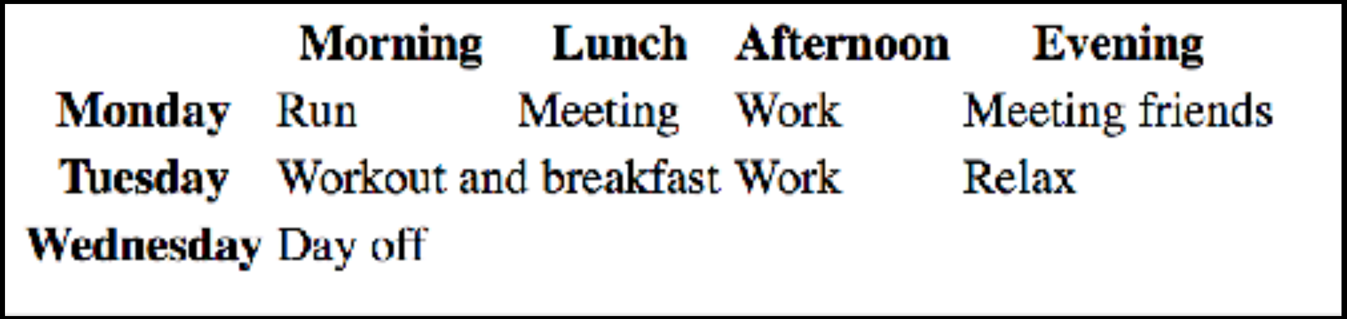
- **<th>** is used to represent the heading for either a column or a row
- Even though there is no content, you should still use it to represent an empty cell
- Add **<scope>** to indicate if it's a heading for row or column



```
<!-- table with headings -->
<table>
    <tr>
        <th scope="col">Day of a week</th>
        <th scope="col">Sports activity</th>
        <th scope="col">Km</th>
    </tr>
    <tr>
        <th scope="row">Monday</th>
        <td>Run</td>
        <td>5</td>
    </tr>
    <tr>
        <th scope="row">Tuesday</th>
        <td>Run</td>
        <td>10</td>
    </tr>
    <tr>
        <th scope="row">Wednesday</th>
        <td>Run</td>
        <td>3</td>
    </tr>
</table>
```

Spanning columns

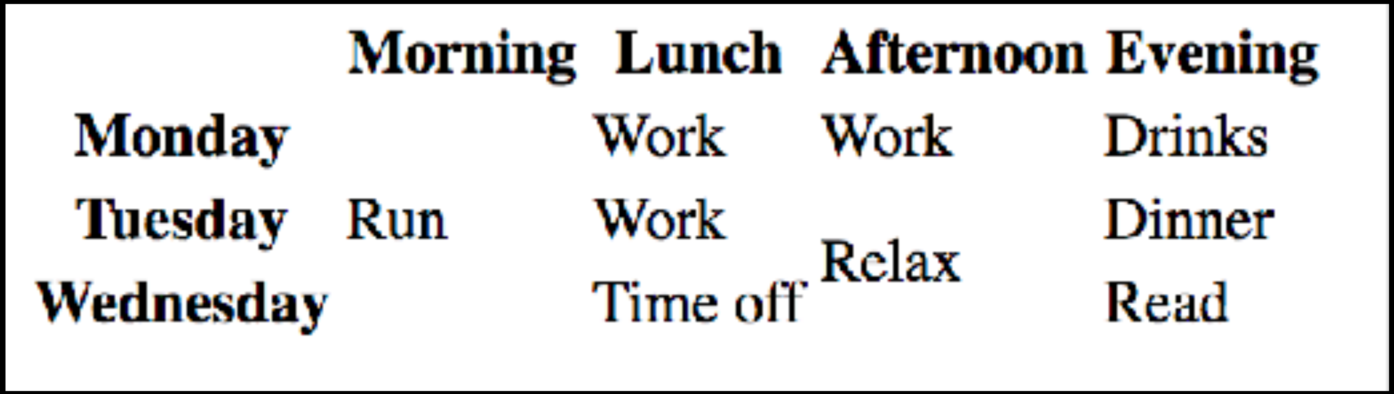
- Sometimes you may need the entries in a table to stretch across more than one column
- You can add ***colspan*** attribute on **<th>** or **<td>** to indicate how many columns that cell should run across



	Morning	Lunch	Afternoon	Evening
Monday	Run	Meeting	Work	Meeting friends
Tuesday	Workout and breakfast		Work	Relax
Wednesday	Day off			

Spanning rows

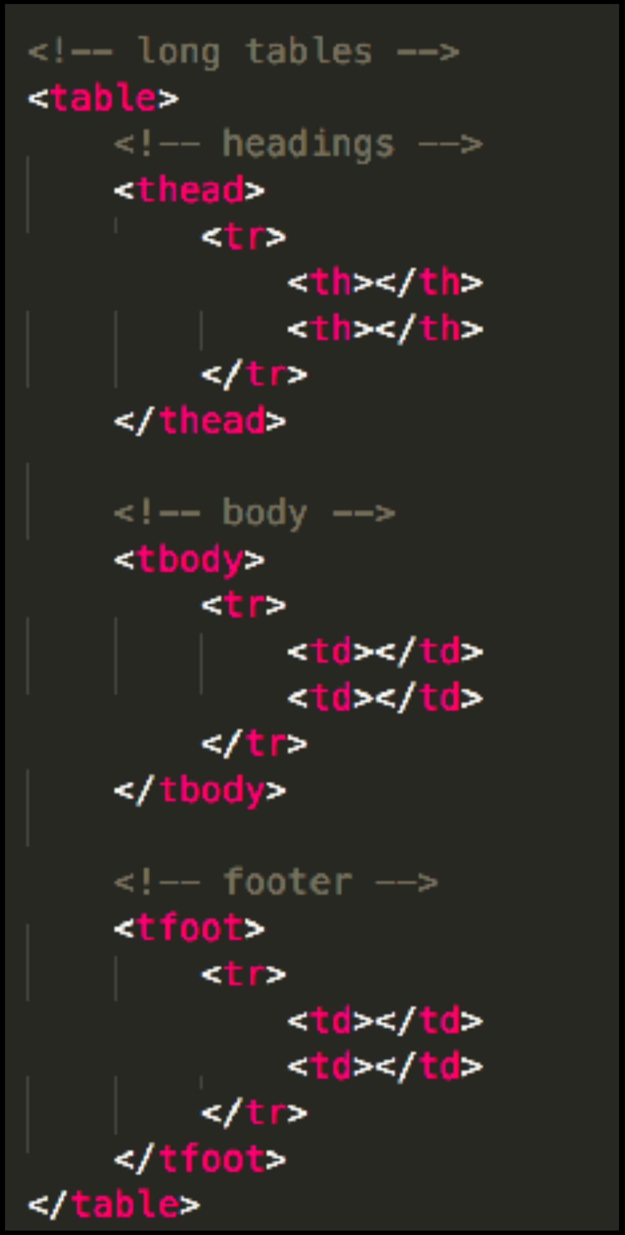
- Add ***rowspan*** attribute on **<th>** or **<td>** to indicate how many columns that cell should run across



	Morning	Lunch	Afternoon	Evening
Monday		Work	Work	Drinks
Tuesday	Run	Work	Relax	Dinner
Wednesday		Time off		Read

Long tables

- Sometimes tables contain a lot of rows and columns
- <thead>, <tbody> and <tfoot> help distinguish between the main content of the table and the first and last rows
- <thead>: the headings of the table should sit here
- <tbody>: the body of the table should sit here
- <tfoot>: the footer of the table should sit here
- Browsers don't really treat these elements any different, but it's helpful for designers working with CSS



```html
<!-- long tables -->
<table>
    <!-- headings -->
    <thead>
        <tr>
            <th></th>
            <th></th>
        </tr>
    </thead>

    <!-- body -->
    <tbody>
        <tr>
            <td></td>
            <td></td>
        </tr>
    </tbody>

    <!-- footer -->
    <tfoot>
        <tr>
            <td></td>
            <td></td>
        </tr>
    </tfoot>
</table>
```

HTML Forms and Inputs

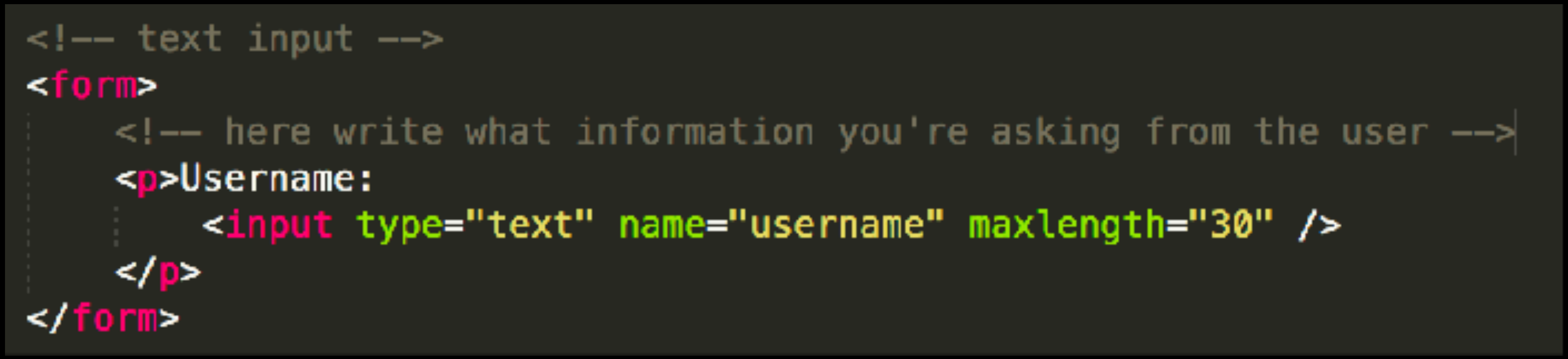
For now we will make the forms on the client side and assume the other side will get taken care of.

Form tag

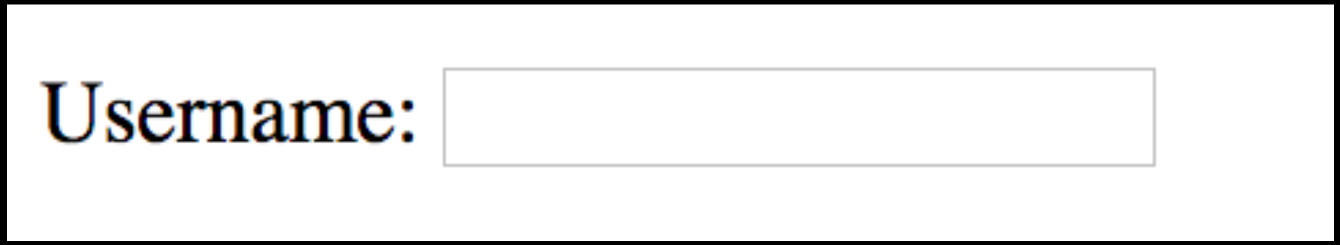
- Usually there would be a **<form> </form>** tag around your forms
- That tag would specify where the information is going (so a server somewhere) and the method for that
- There's also a value which is the answer that's filled into forms which is sent to the server as well
- We are still going to use the form tag, but without the other information, it's a good practice for future life as a web developer

Text input

- Single line text input
- The type is text because it's text input
- It has a name which would be accessible later, but just be clear with it
- Has a max length option for maximum character

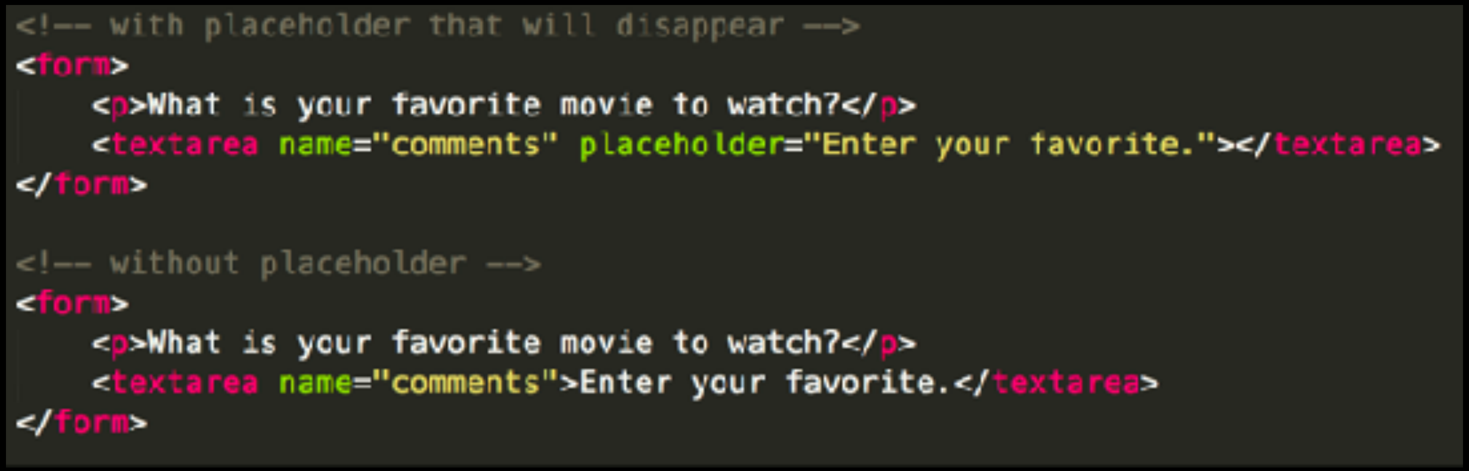


```
<!-- text input -->
<form>
    <!-- here write what information you're asking from the user -->
    <p>Username:
        <input type="text" name="username" maxlength="30" />
    </p>
</form>
```



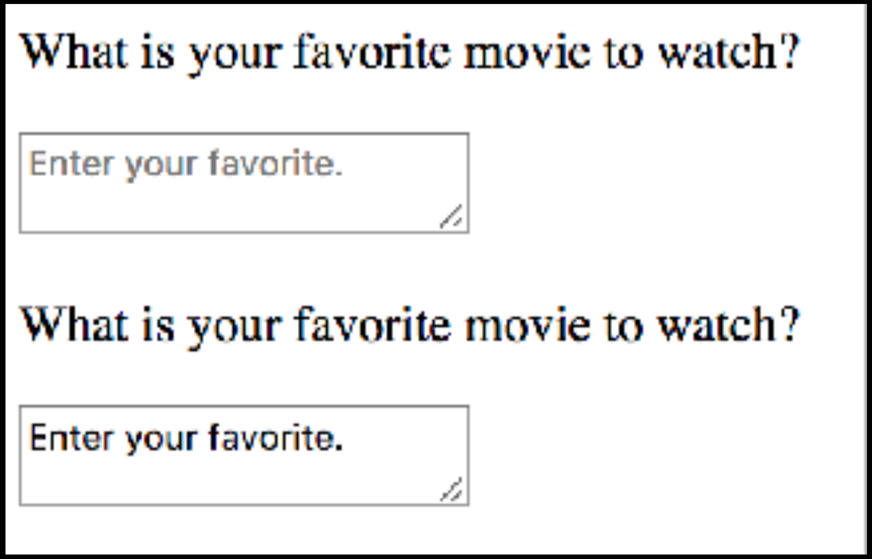
Text area

- For multi-line text input
- Has opening / closing tags <textarea></textarea>
- Text in between tags appears as a placeholder when the page loads
- Text under placeholder attribute disappears when clicked on



```
<!-- with placeholder that will disappear -->
<form>
    <p>What is your favorite movie to watch?</p>
    <textarea name="comments" placeholder="Enter your favorite."></textarea>
</form>

<!-- without placeholder -->
<form>
    <p>What is your favorite movie to watch?</p>
    <textarea name="comments">Enter your favorite.</textarea>
</form>
```



What is your favorite movie to watch?

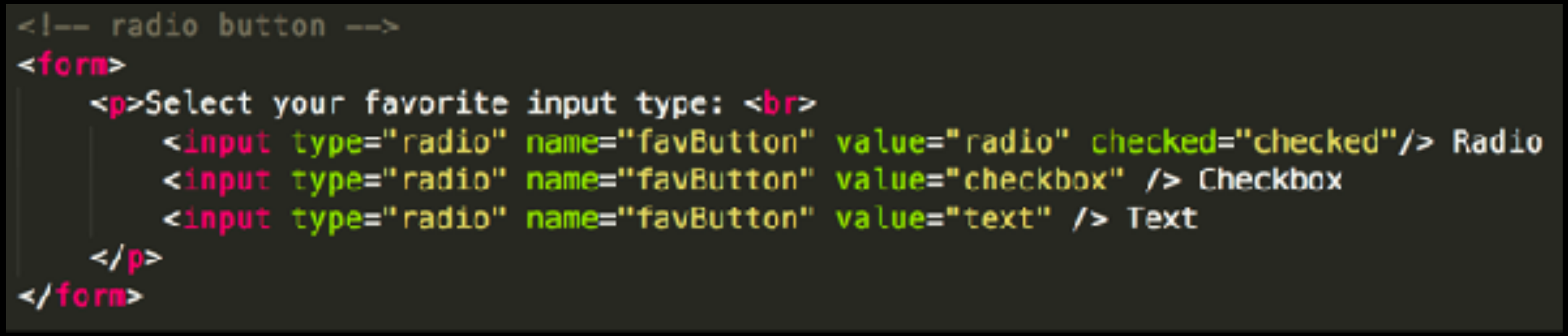
Enter your favorite.

What is your favorite movie to watch?

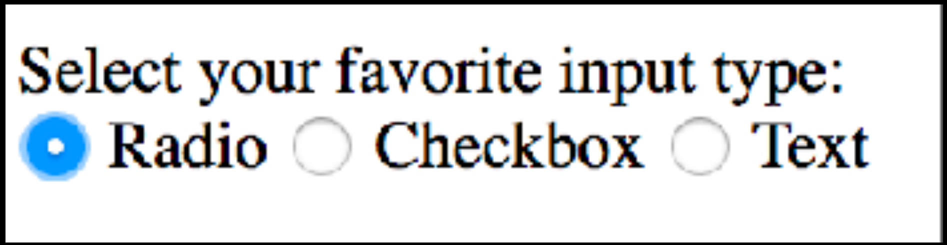
Enter your favorite.

Radio buttons

- The little circle buttons we see everywhere are referred to as radio buttons
- They'll have a value attribute because the possible answers are already written
- Can have a default checked button
- Allow users to pick only one option



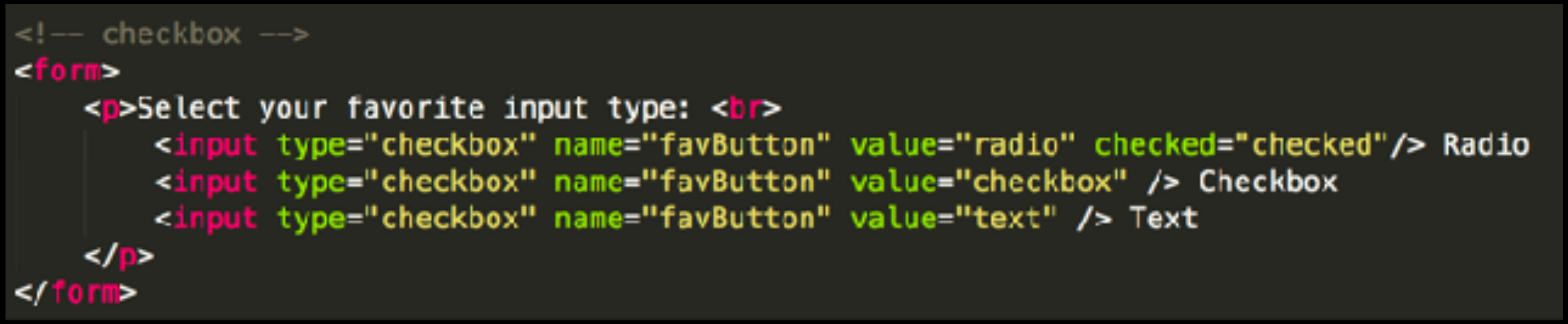
```
<!-- radio button -->
<form>
    <p>Select your favorite input type: <br>
        <input type="radio" name="favButton" value="radio" checked="checked"/> Radio
        <input type="radio" name="favButton" value="checkbox" /> Checkbox
        <input type="radio" name="favButton" value="text" /> Text
    </p>
</form>
```



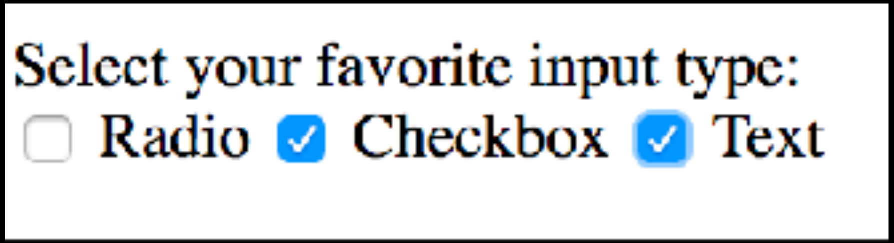
Select your favorite input type:
[✓ Radio] Checkbox Text

Checkbox

- Check one or more options



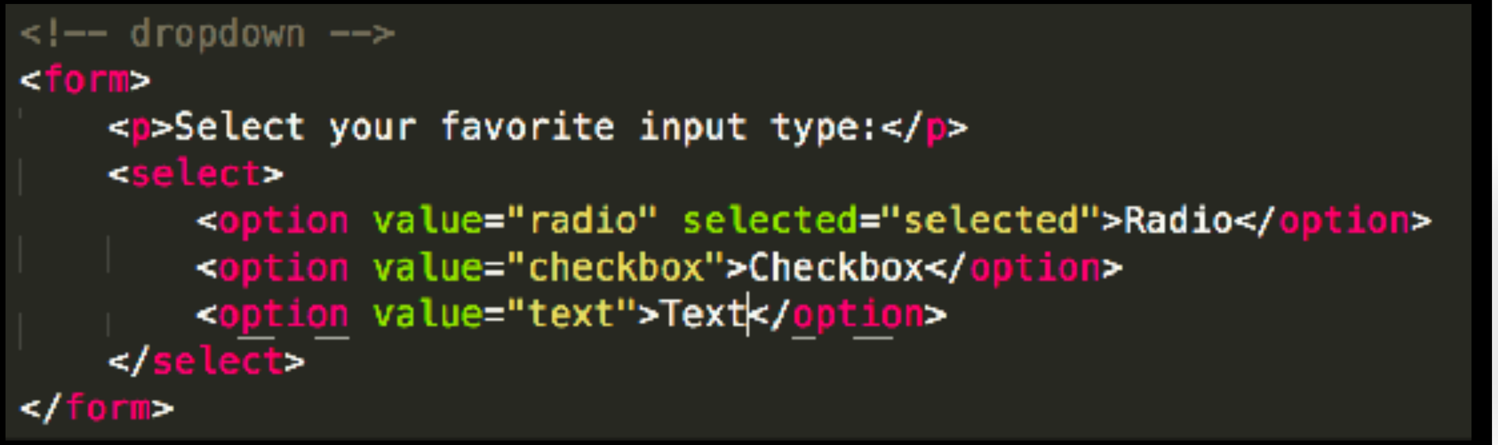
```
<!-- checkbox -->
<form>
    <p>Select your favorite input type: <br>
        <input type="checkbox" name="favButton" value="radio" checked="checked"/> Radio
        <input type="checkbox" name="favButton" value="checkbox" /> Checkbox
        <input type="checkbox" name="favButton" value="text" /> Text
    </p>
</form>
```



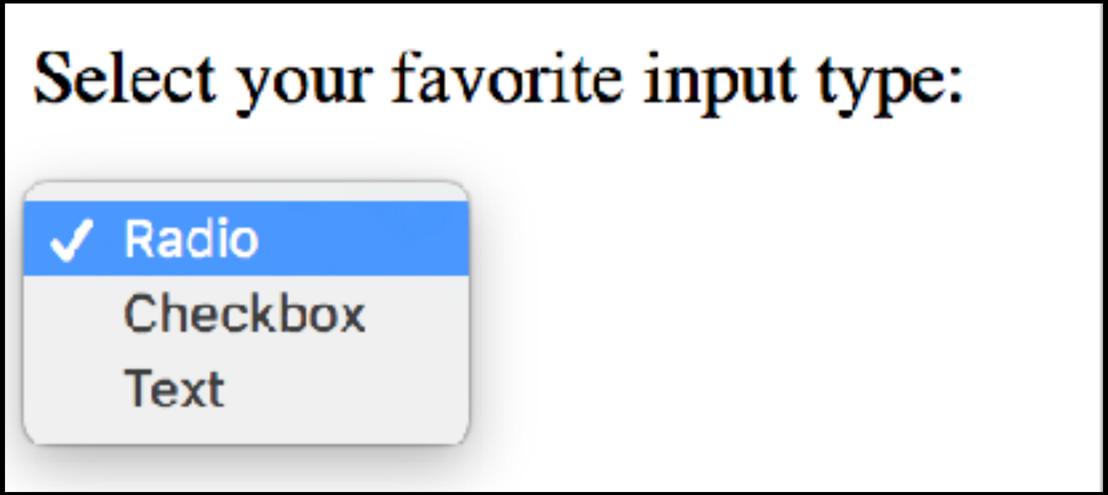
Select your favorite input type:
☐ Radio ✓ Checkbox ✓ Text

Drop down list

- Also known as select box
- Allows users to select one option from drop down list
- **<select>** element is used to create a dropdown list box
- It contains two or more **<option>** elements



```html
<!-- dropdown -->
<form>
    <p>Select your favorite input type:</p>
    <select>
        <option value="radio" selected="selected">Radio</option>
        <option value="checkbox">Checkbox</option>
        <option value="text">Text</option>
    </select>
</form>
```

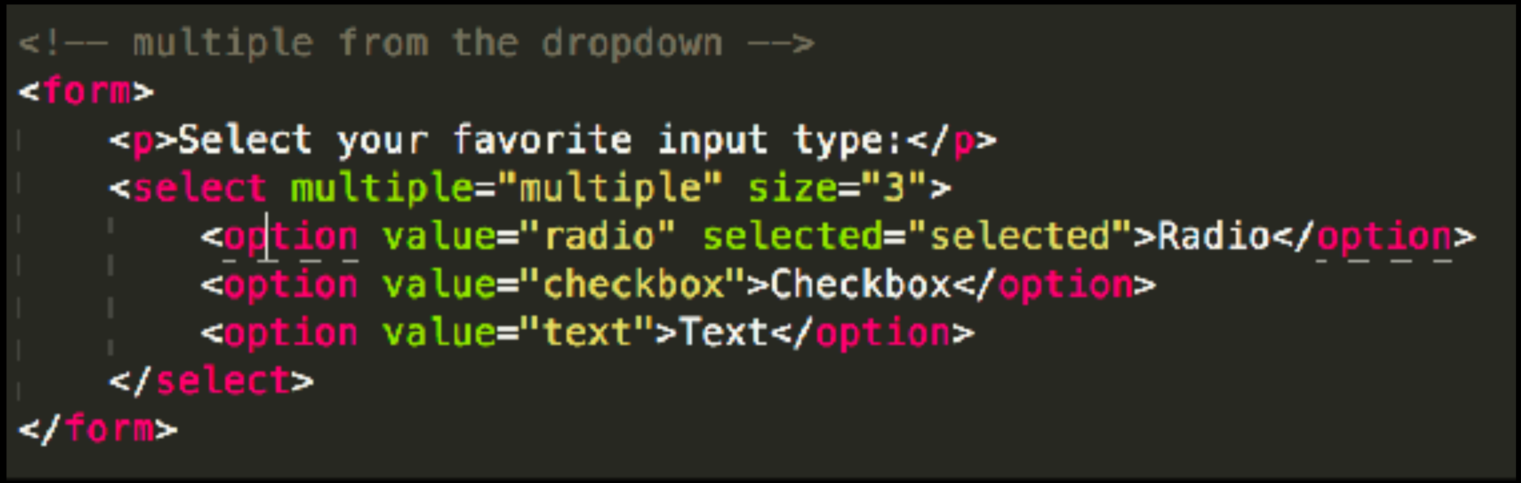


Select your favorite input type:

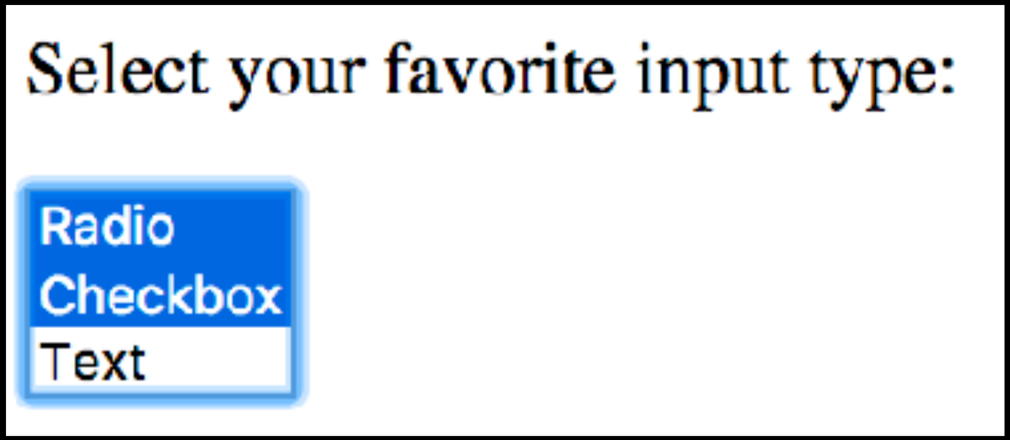
✓ Radio
Checkbox
Text

Multiple select box

- size attribute turns dropdown into a box that shows more than one option
- multiple attribute allows users to select more than one option



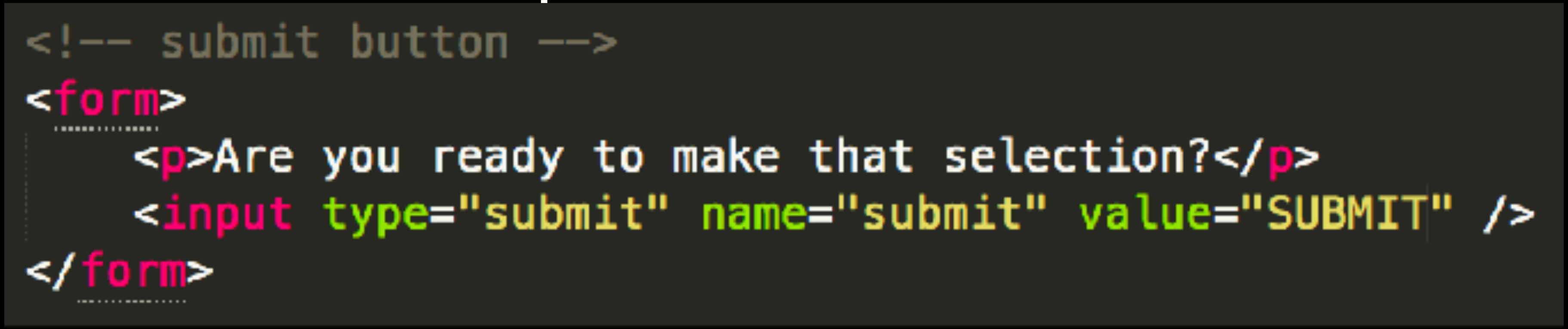
```
<!-- multiple from the dropdown -->
<form>
    <p>Select your favorite input type:</p>
    <select multiple="multiple" size="3">
        <option value="radio" selected="selected">Radio</option>
        <option value="checkbox">Checkbox</option>
        <option value="text">Text</option>
    </select>
</form>
```



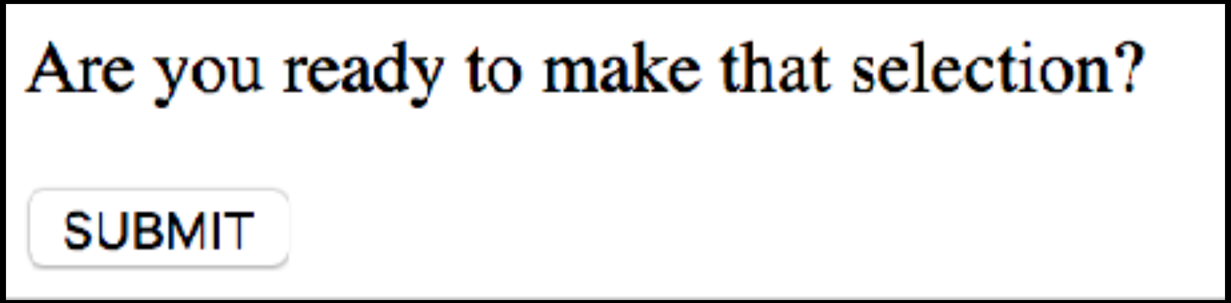
Select your favorite input type:

Submit button

- Probably most common button type to submit user input



```
<!-- submit button -->
<form>
    <p>Are you ready to make that selection?</p>
    <input type="submit" name="submit" value="SUBMIT" />
</form>
```



Are you ready to make that selection?