

class repo: https://github.com/rebleo/intrntWeb_RH

class wiki: https://github.com/rebleo/intrntWeb_RH/wiki

if you do not have a github account yet:

<https://education.github.com/pack>

storm server: <http://storm.cis.fordham.edu/~rleopold>

what is the web?

A Web Page **WAS:**

HTML - Hyper Text Mark Up

is a grammar for structuring web pages. It defines paragraphs, headings, data tables + media elements. HTML describes the content of the page - not how it looks.

CSS - Cascading Style Sheet

rules for styling a web page. Setting colors, typeface, and the layout. It can be used to consider the design of your page across different platforms and screen sizes.

(Web. 1.0)

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>  Intro to CSS</title>
```

```
  </head>
```

```
  <body>
```

```
    <h1>
```

```
    This Webpage though....
```

```
    </h1>
```

```
    <p>
```

```
    here is a page of text. here is a page of text. here...
```

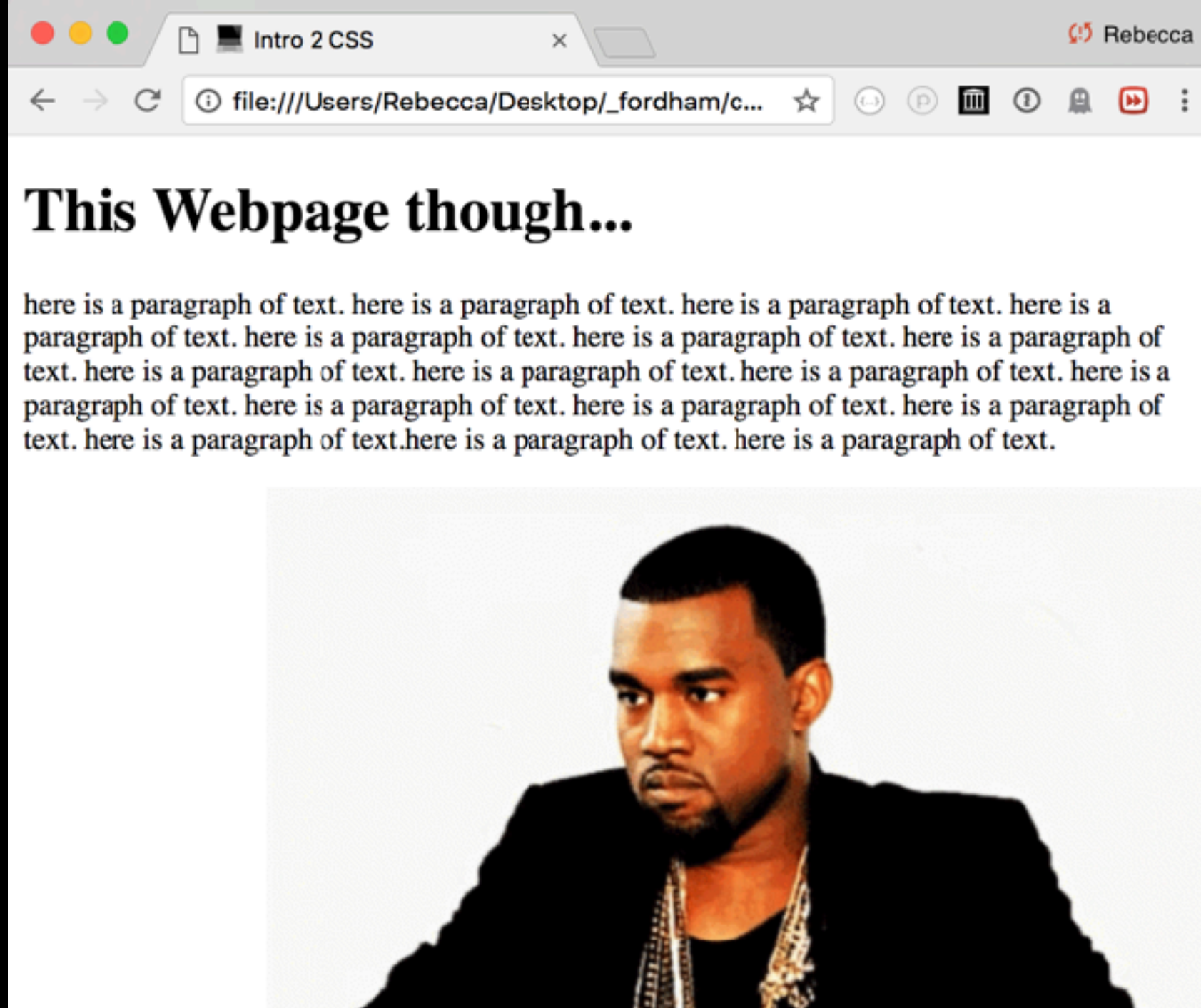
```
    </p>
```

```
    <img src = "kanye.gif">
```

```
  </body>
```

```
</html>
```

the world w/out css



CSS - Cascading Style Sheet

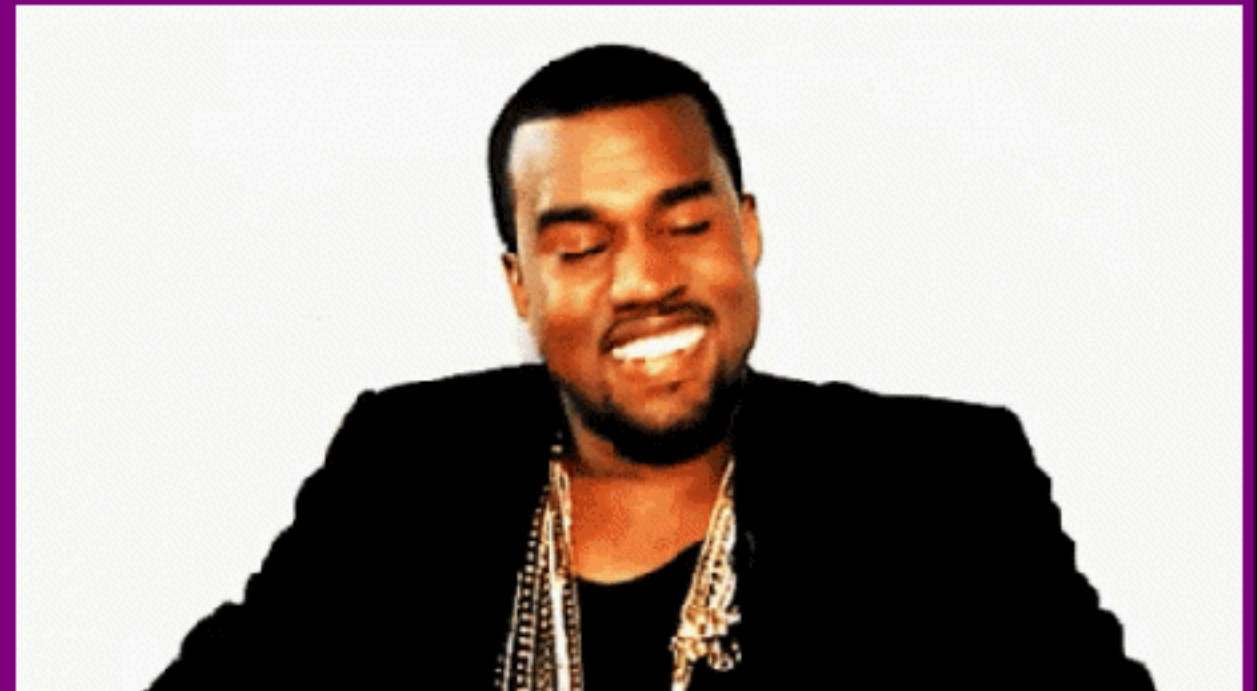
```
h1 {  
    color: #FF4500  
}
```

```
body {  
    background: #000080;  
    font-family: sans-serif;  
    color: rgb(255,255,255);  
}
```

This Webpage though...

Is what the world is like without css

here is a paragraph of text. here is a paragraph of text. here is a paragraph of text. here is a sentence that has a link. here is a paragraph of text. here is a paragraph of text. here is a paragraph of text. here is a paragraph of text. here is a paragraph of text. here is a paragraph of text. here is a paragraph of text. here is a paragraph of text. here is a paragraph of text. here is a paragraph of text.



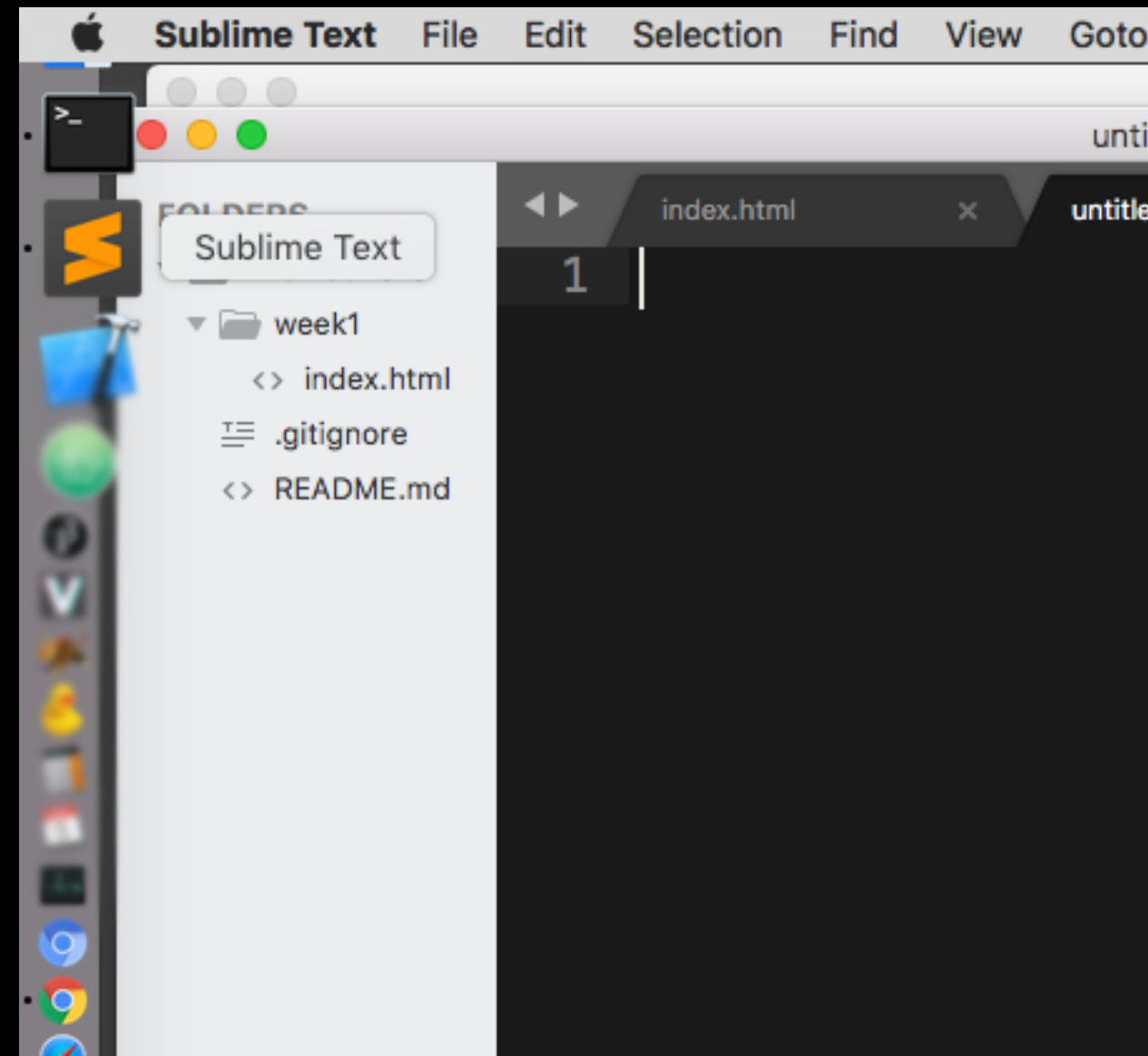
+ w/ CSS

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

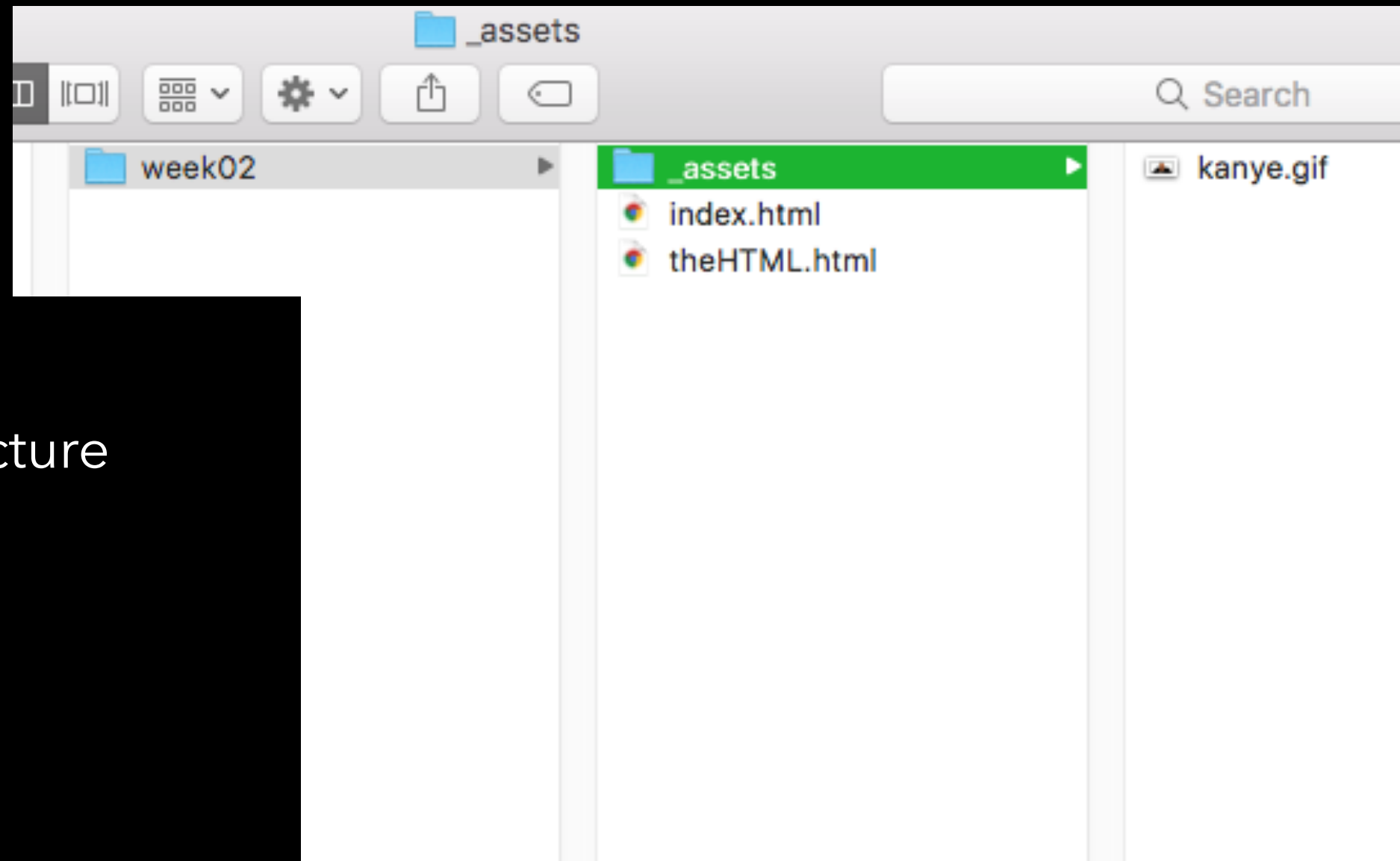
.html	hyper text mark up
.css	cascading style sheet
.js	javascript

We can write these files with a text editor.
Like **Sublime**. There are others of course...

WebDev WorkFlow – We will be hosting
local servers on our machines to **prototype**
our websites before serving them up in
public.



Parent + Child File Structure or: File Paths



prototype: local http server

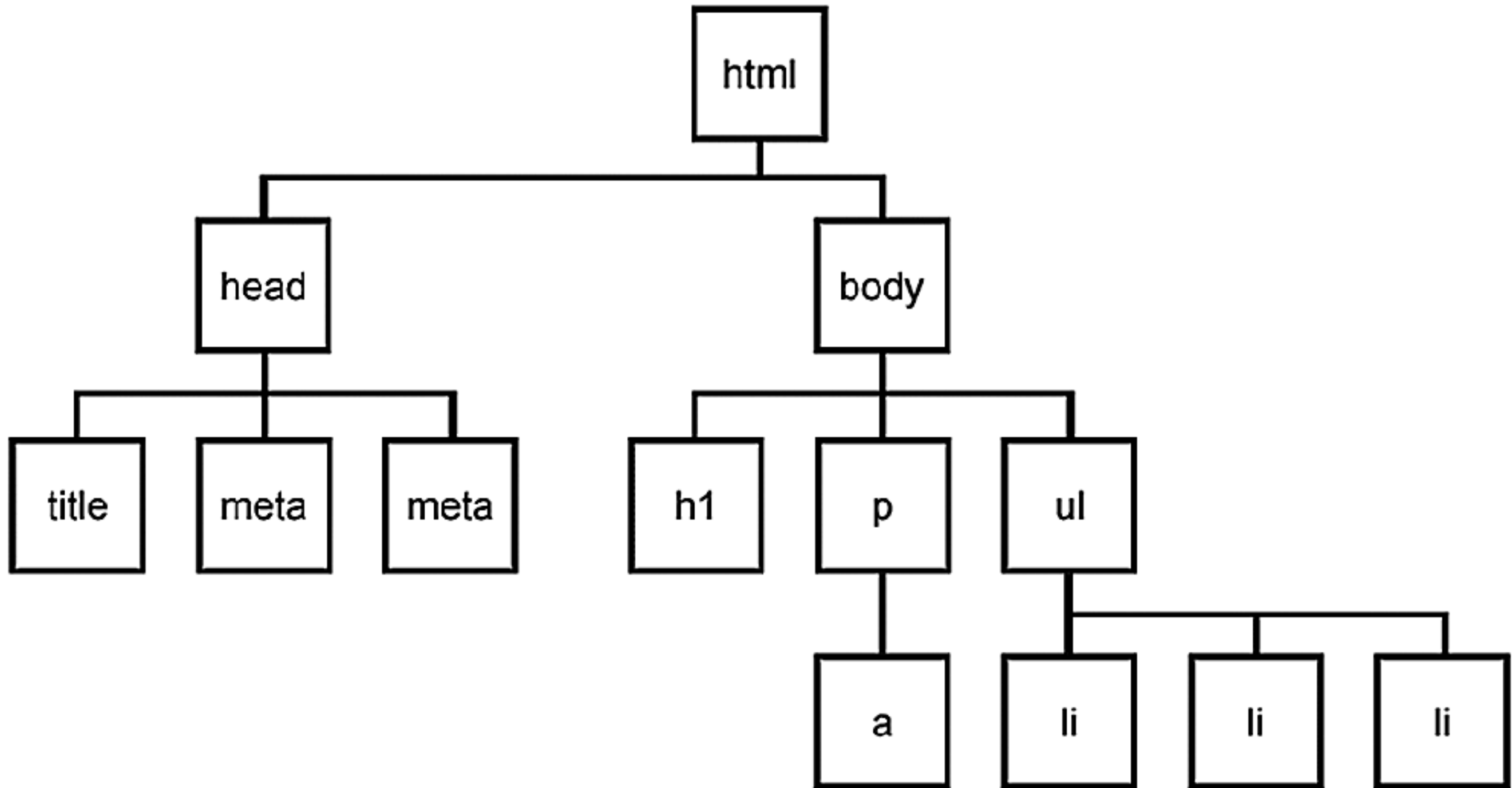
publish: posted to storm server via **ssh** or **sftp**

< HTML >


Hypertext Markup Language

Describes the **content** + **structure** of a web page;
NOT a programming language

Parent / Child Element Structure

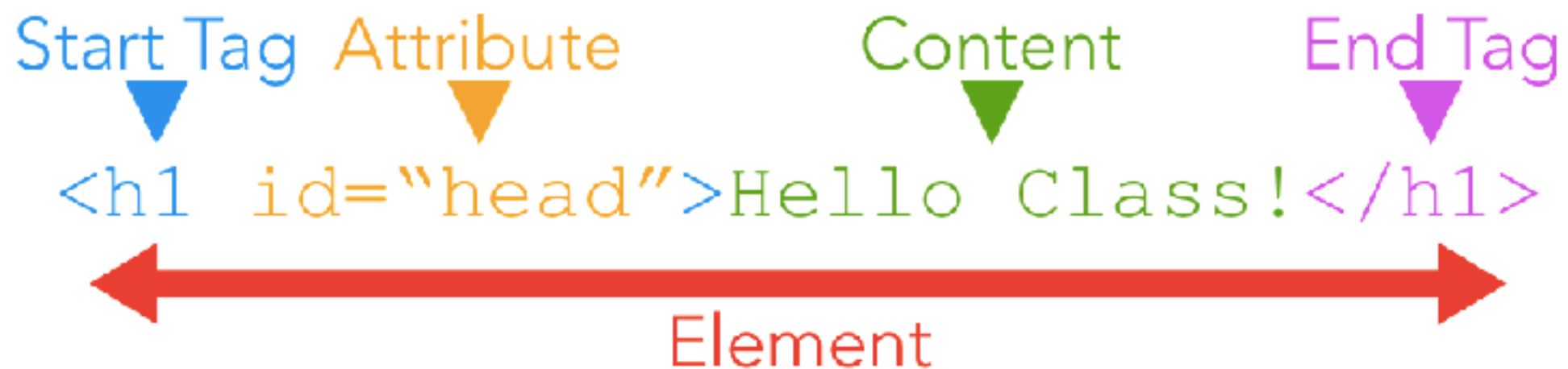


HTML - Hyper Text Mark Up

```
<!DOCTYPE html>
<html>
  <head>
    <title>  Internet + Web Week 1</title>
  </head>
  <body>
    this is a webpage of wonderful text
  </body>
</html>
```

HTML Elements / Tags, Attributes, Content

- Elements and tags used interchangeably



tag attribute value

```
<video src= "filepath/file.mov" alt= "this is the video" height="300"></video>
```

```
<html attribute= "value" attribute= "value" attribute= "value"> </html>
```

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>
<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 1 </title>
  </head>
  <body>
    div is the child of body
    <div>
      Here's a Great Site.
    </div>
  </body>
</html>
```

body is the child of 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
- **** is for **bold** **** is for *italics*
- **ul, ol, li** are used for making lists
 - **ul**: unordered lists
 - **ol**: ordered lists
 - **li**: an individual list tag
- **
** 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

OPENING
LINK TAG

URL WE ARE
DIRECTED TO

TEXT WE
CLICK ON

CLOSING
TAG

```
<!-- link -->  
<a href="https://www.fordham.edu/" target="_blank">Fordham University</a>
```



FORDHAM
UNIVERSITY

< a href — stands for *hyperlink reference*

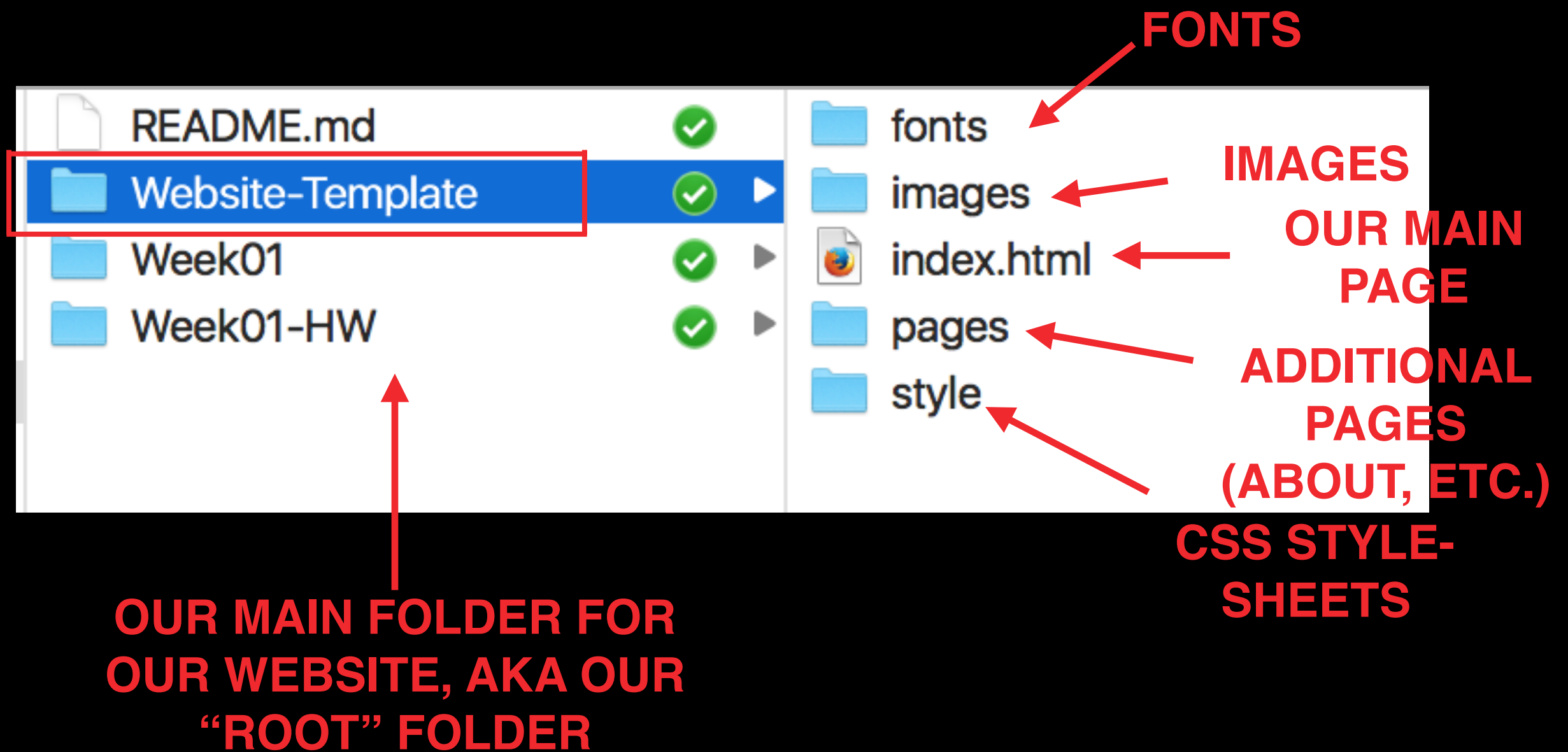
Linking to pages on the same site

RELATIVE URLS

Link types:

- parent folder: `Homepage`
- same folder: `Homepage`
 - child folder: `Photos`
 - id attribute: `Different element on page`

```
<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>
```



Why **index.html**?

The main homepage of a site written in HTML (and the homepage of each section in a child folder) is called index.html.

Web servers are usually set up to return the index.html file if no file name is specified. Therefore, it's always a good idea to name your "home" page index.html

The `` 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: Relative (local) vs. URL

- The **** tag is for images, which can be on your local directory or on another webpage. Read all about **** tag [here](#)

```
<!-- An image on the local directory -->
```

```

```

```
<!-- Or with size specs -->
```

```

```

```
<!-- Image from another site -->
```

```

```

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 Engine Optimization (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.

Like the `` tag, the `<video>` tag requires a `src` attribute with a link to the video source.

Unlike the `` tag however, the `<video>` element requires an opening and a closing tag.

<video /> structure

main
tag

poster

width/
height

control
attributes

```
<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>
```

different
sources

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 `` 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.

<audio /> structure

main
tag

control
attributes

```
<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>
```

different
sources

text is the
file cannot
be found

Some Media 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

`<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>
```

**** 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 `` tag will generally render as *italic* emphasis.

The `` will generally render as **bold** emphasis.

`
`

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.

Text input

Username: |

Password input

Username:

Password:

Text area

What is your favorite movie to watch?

What is your favorite movie to watch?

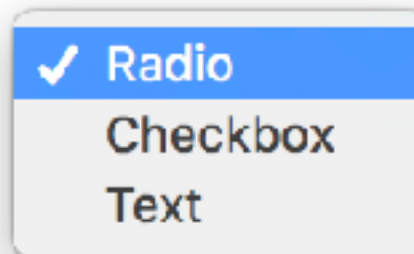
Checkbox

Select your favorite input type:

☐ Radio ☒ Checkbox ☒ Text

Drop down list

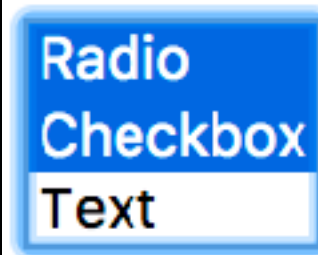
Select your favorite input type:



A drop-down menu with a light blue border and a light blue background. The menu is open, showing three options: 'Radio' (selected with a checkmark), 'Checkbox', and 'Text'.

Multiple select box

Select your favorite input type:



A multiple select box with a light blue border and a light blue background. The box is open, showing three options: 'Radio' (selected), 'Checkbox' (selected), and 'Text' (not selected).

Submit button

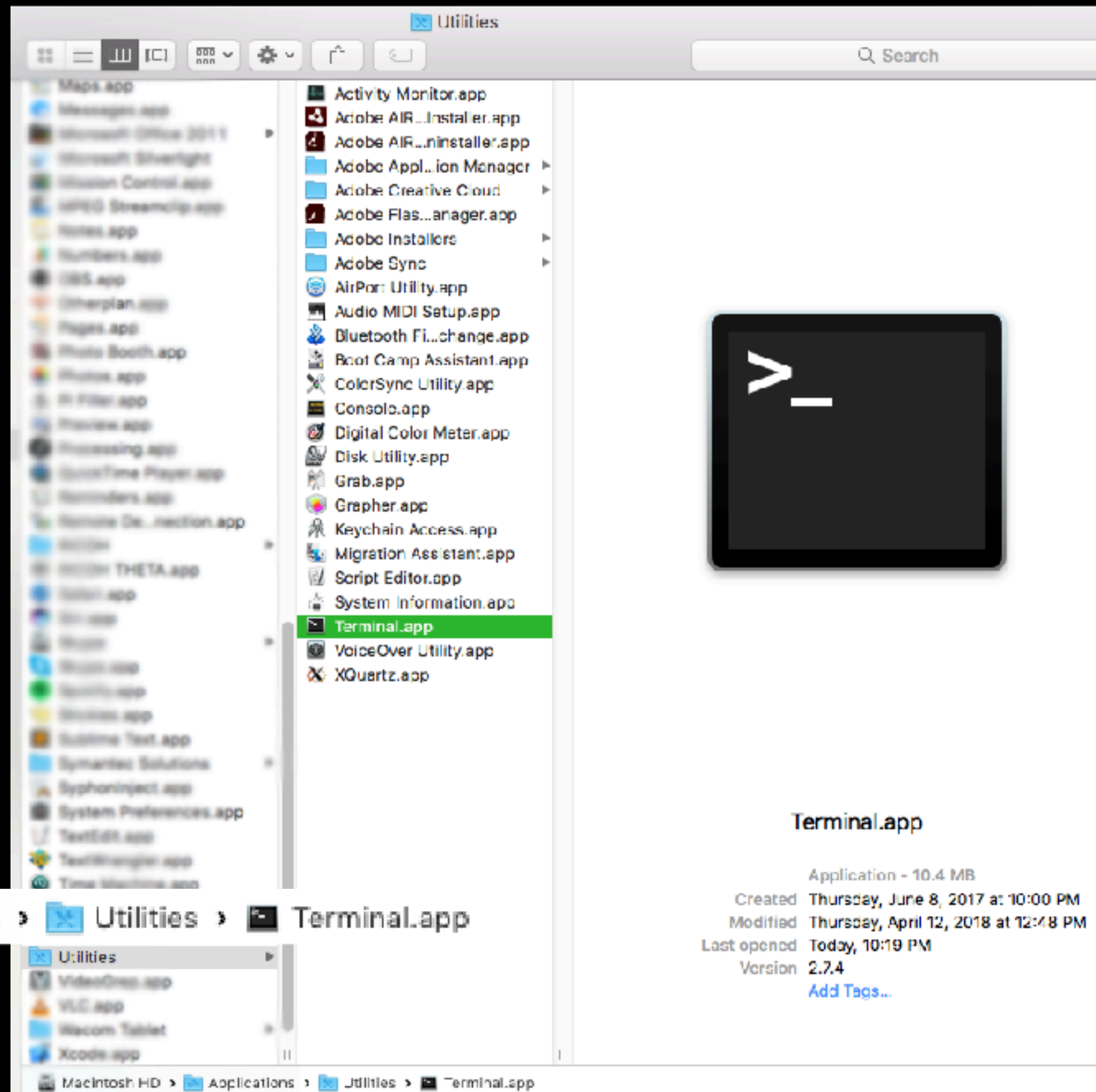
Are you ready to make that selection?

SUBMIT

for mac users -
to open your terminal

the file path is:

Macintosh HD > Applications > Utilities > Terminal.app



Starting a local http server from the command line...

If you have installed **Python 3.0+**:

```
python -m http.server
```

in browser address bar: **localhost:8000**

Mac - to close the server: **COMMAND C**

Wndws - to close the server: **CNTRL C**

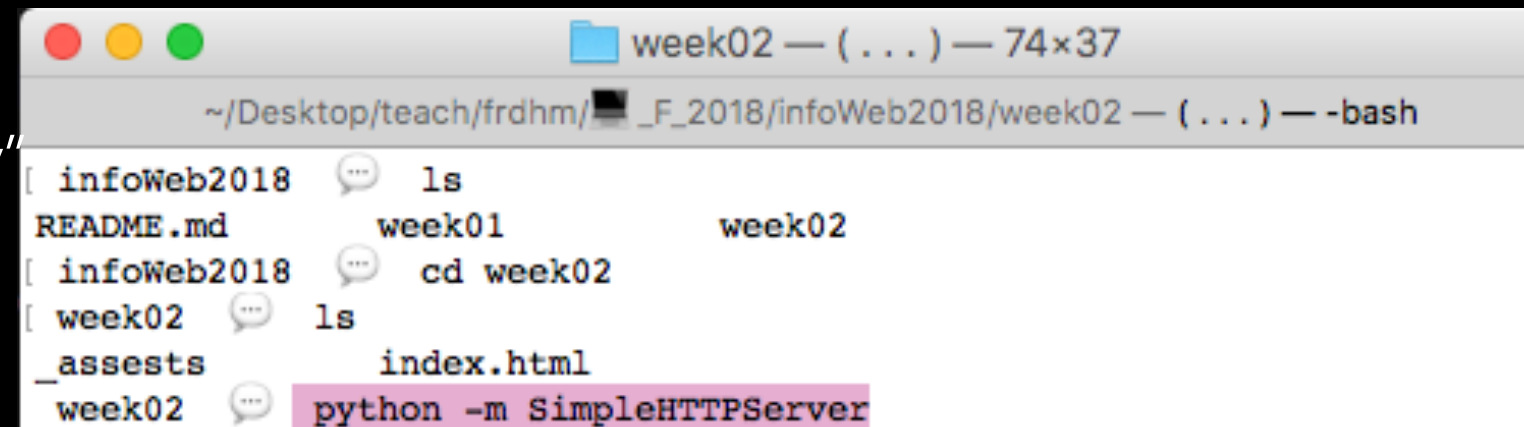
in Terminal we are speaking Unix :

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

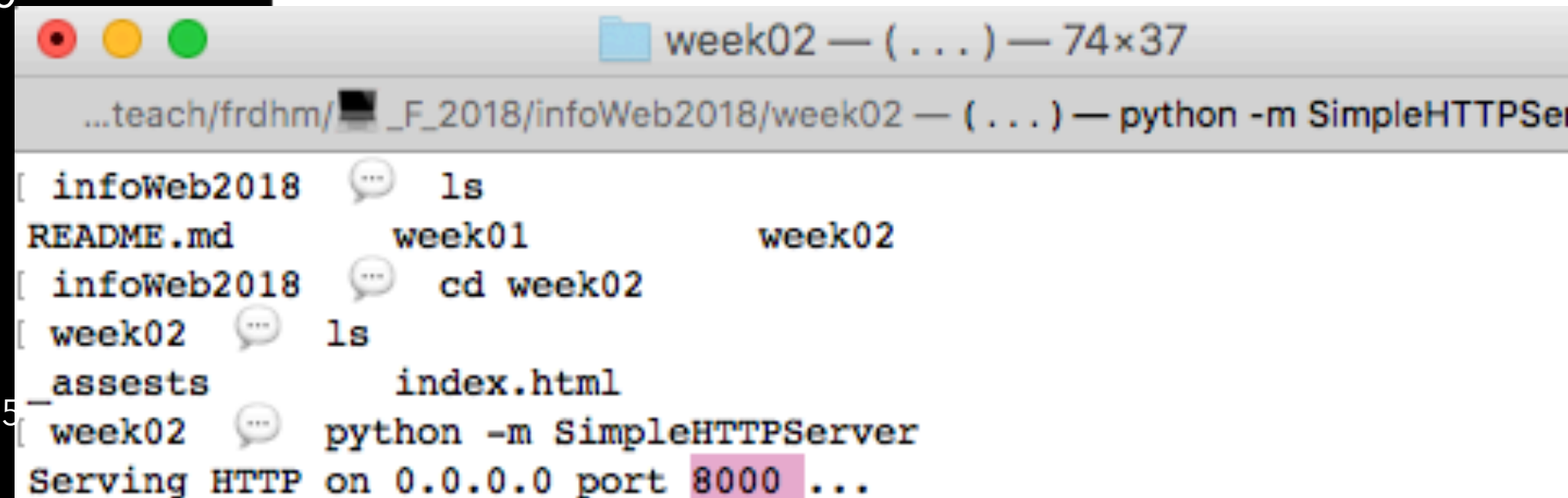
Running a local Python HTTP Server
in Mac OS - this is very simple :

When inside yr project folder simply
type the following command:

"python -m SimpleHTTPServer"
– defaults to port 8000



```
~/Desktop/teach/frdhn/_F_2018/infoWeb2018/week02 — ( ... ) — -bash
[ infoWeb2018  ...  ls
README.md      week01          week02
[ infoWeb2018  ...  cd week02
[ week02  ...  ls
_assests      index.html
week02  ...  python -m SimpleHTTPServer
```



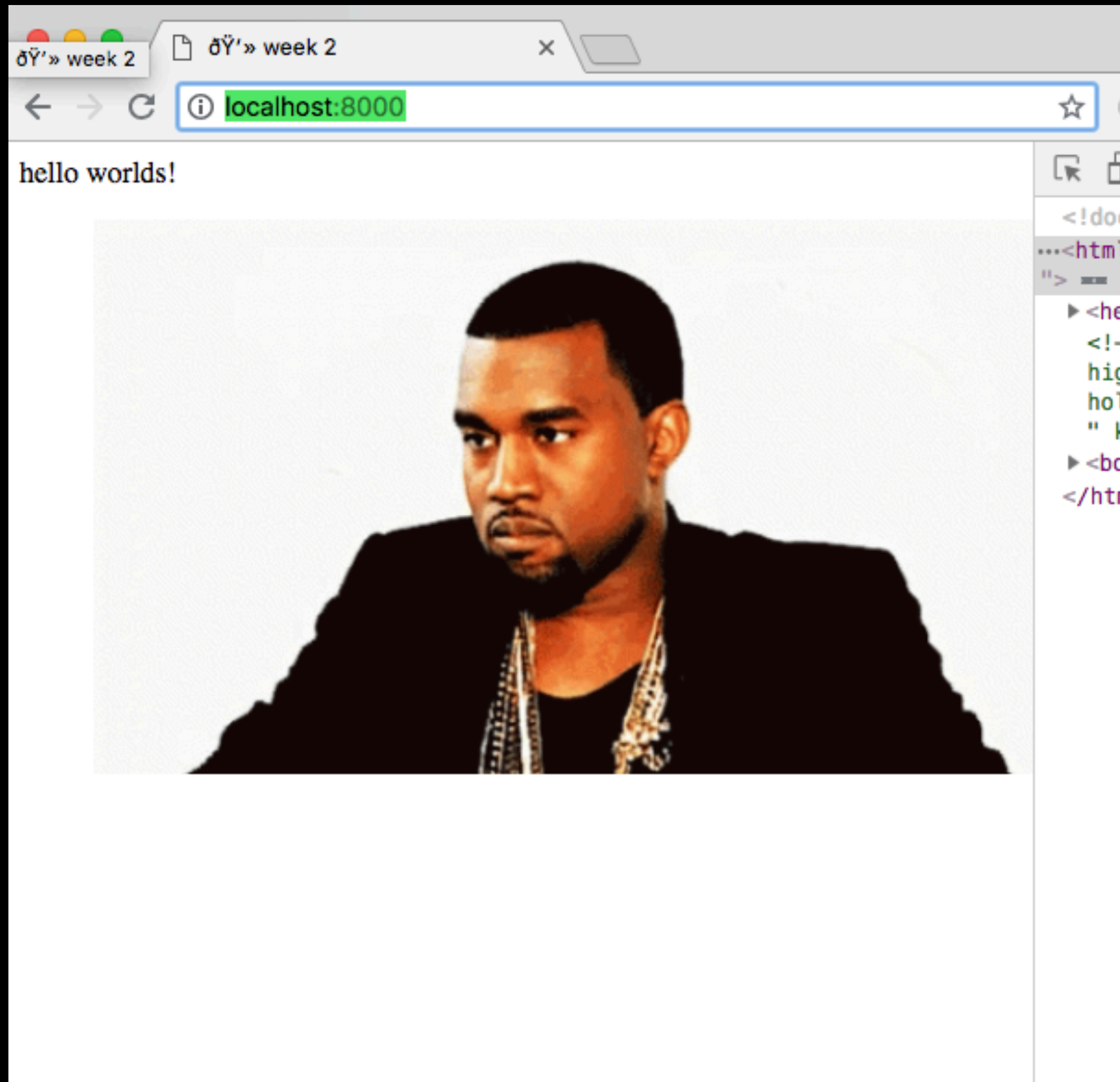
```
...teach/frdhn/_F_2018/infoWeb2018/week02 — ( ... ) — python -m SimpleHTTPServer
[ infoWeb2018  ...  ls
README.md      week01          week02
[ infoWeb2018  ...  cd week02
[ week02  ...  ls
_assests      index.html
week02  ...  python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

if we wrote:

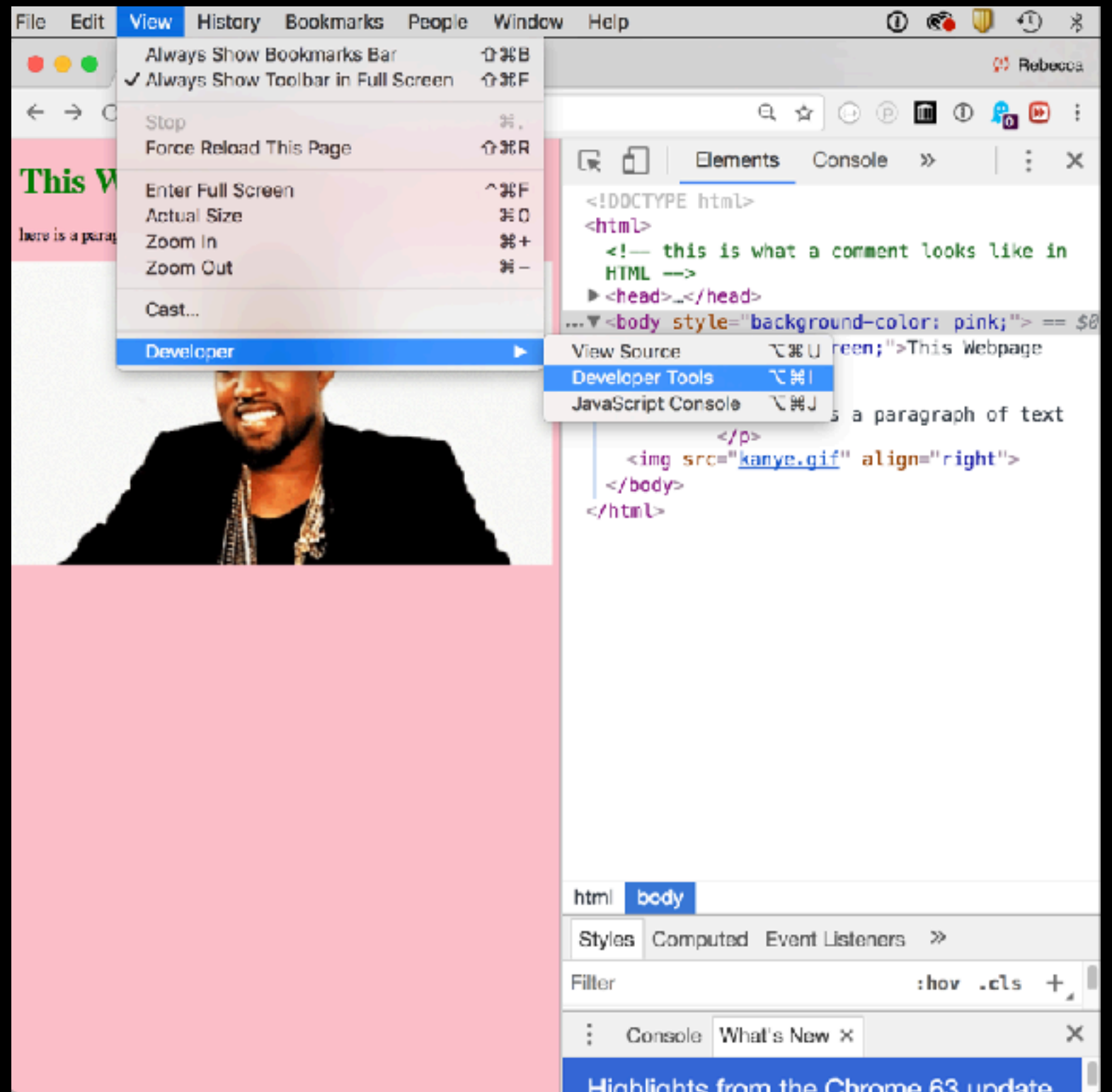
"python -m SimpleHTTPServer 12345"
- we would go to port 12345

url is:

localhost:8000



+ **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 equivalent to unplugging a hard drive w/ out "ejecting it" - BAD PRACTICE. As DIGITAL CITIZENS - we ♥ our hardware + software...)