

authoring the web: workflow

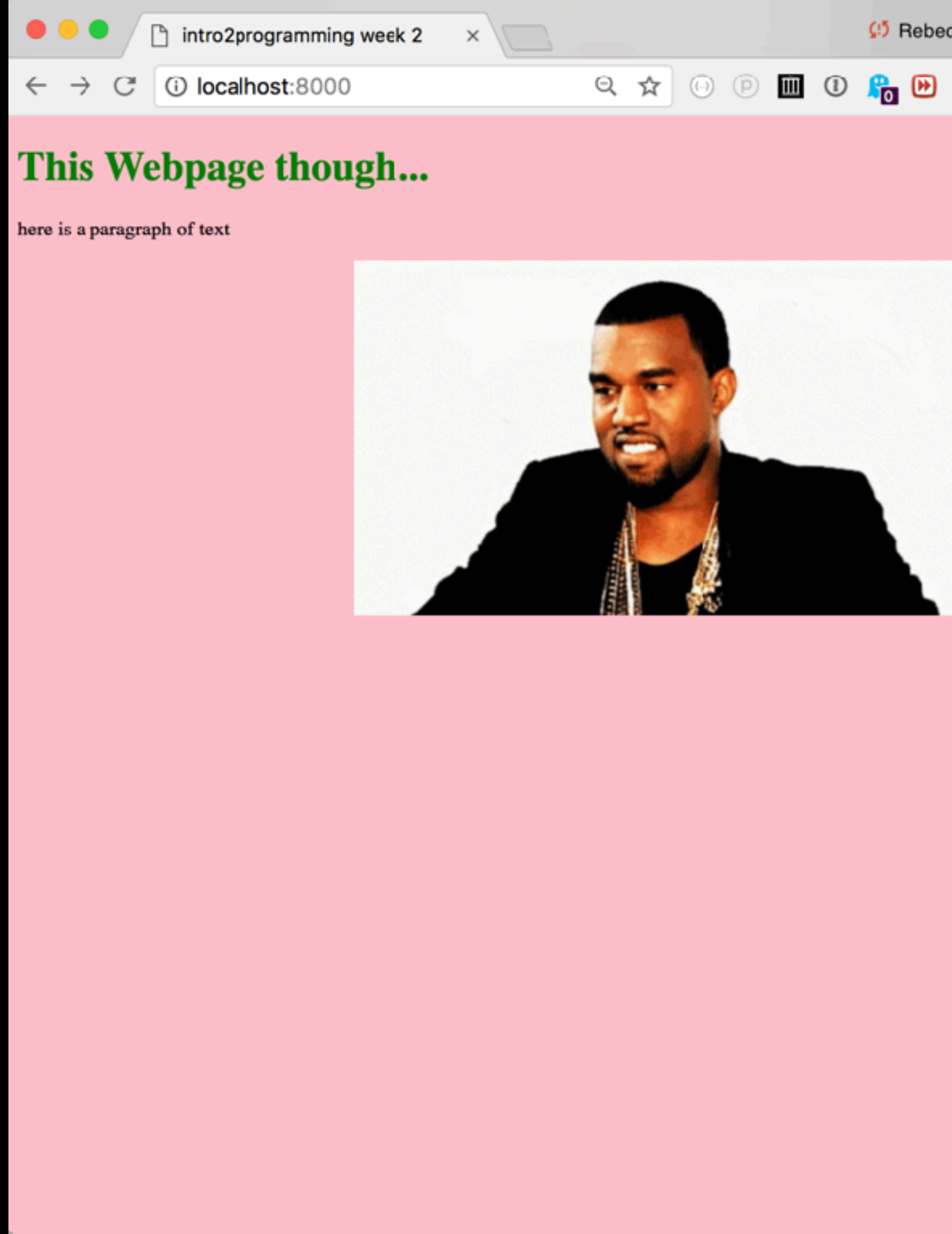
HTML
CSS

HTML - Hyper Text Mark Up


is a grammar for structuring web pages. It defines paragraphs, headings, data tables, images + video.

CSS - Cascading Style Sheet

rules for styling a web page. Setting colors, typeface, and laying out content into columns.



HTML - Hyper Text Mark Up

```
<!DOCTYPE html>
<html>
  <head>
    <title>  Emerging Media 2: Week 2</title>
  </head>
  <body>
    this is a webpage of wonderful text
  </body>
</html>
```

```
<body>      <!-- opening tag  
this is a webpage of wonderful text  
</body>     <!-- closing tag
```

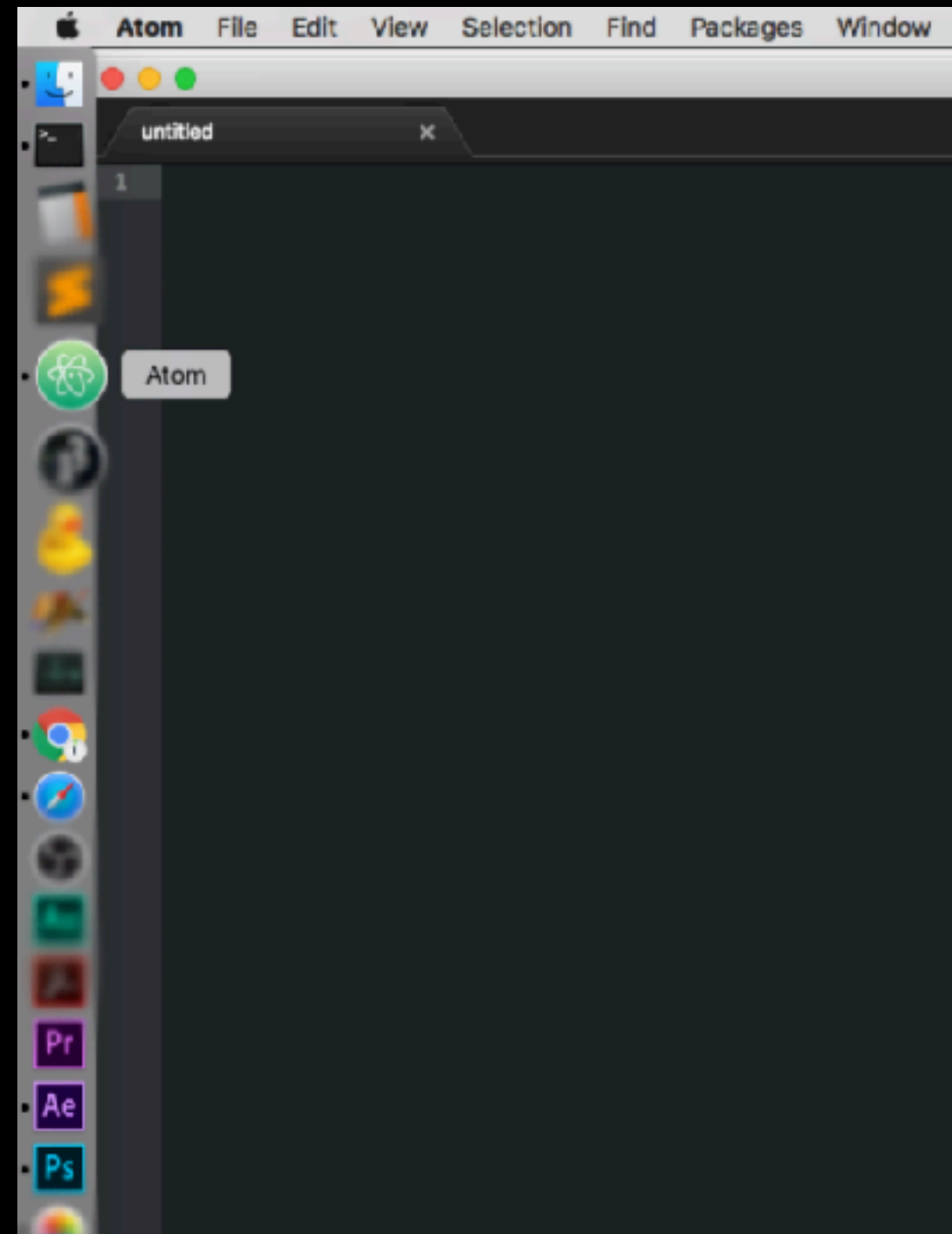
Web pages are made
of three different file types. All of which we
are capable of creating on our machines!

It's easier than you think...

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

We can create + write these files with a text editor.
Like Sublime.

Mimic a server w/ our command line
And look at our work in the browser.



Prototyping yr website: local http server

Publishing yr website: Film + Media Server

The image displays a web development workflow with three main components:

- Code Editor (Sublime Text):** Shows the `index.html` file with the following content:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title> Emerging Media 2: Week 2</title>
5     <!-- the following line of metadata enables the browser to render emoji -->
6     <meta charset="utf-8">
7     <!-- <link rel="stylesheet" type="text/css" href="theStyle.css"> -->
8   </head>
9   <body>
10    <h1>This Webpage though...</h1>
11    <h2>Is what the world is like with/ css</h2>
12    <p>
13      here is a paragraph of text. here is a paragraph of text. here is a
14      paragraph of text. here is a paragraph of text. <a href="https://developer.mozilla.org/en-US/docs/Web/CSS" target="blank">here
```
- Browser (Chrome):** Displays the rendered webpage at `localhost:8000/_code/`. The page has a purple background and contains the text:

This Webpage though...

Is what the world is like with/ css

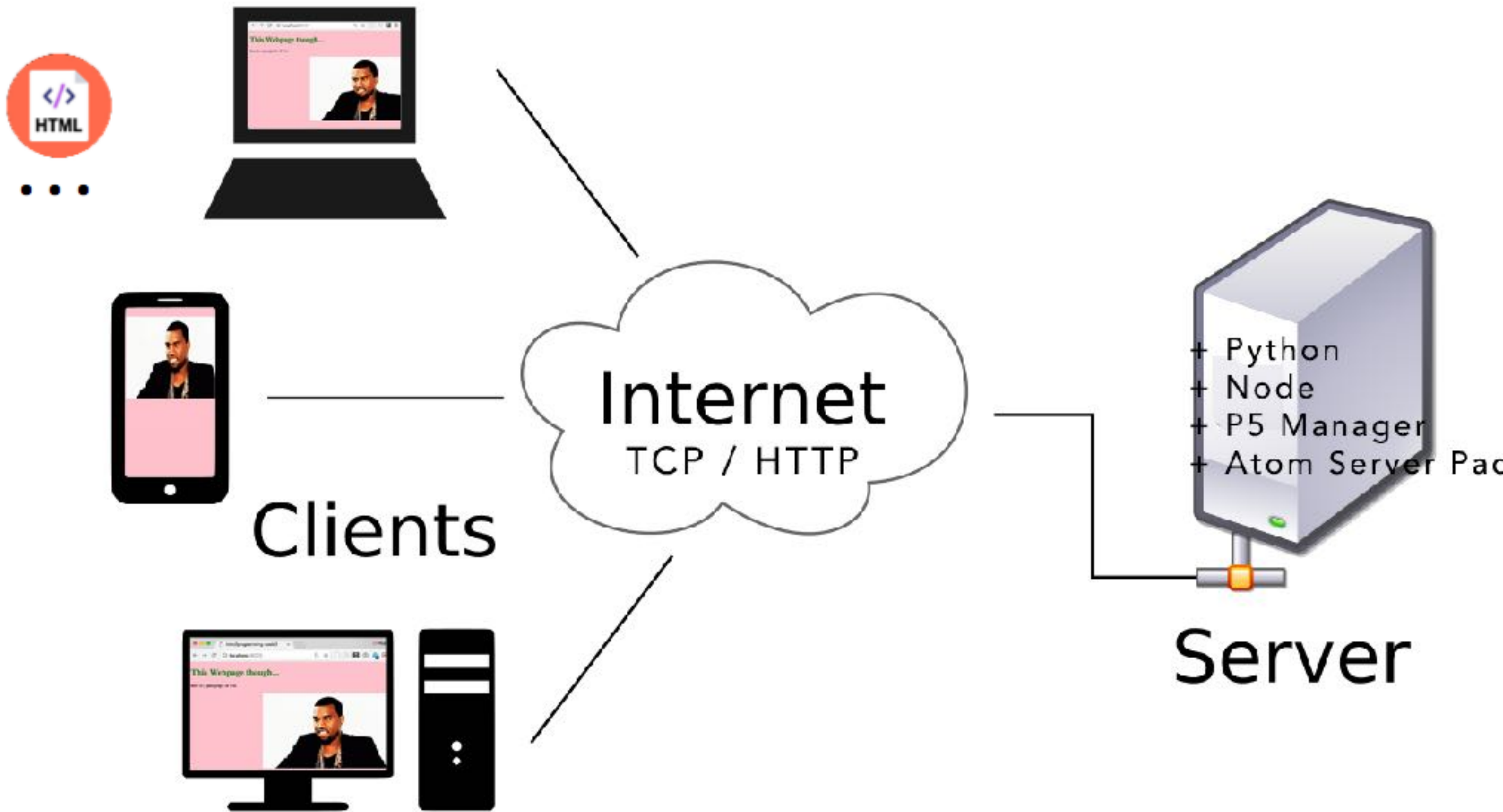
here is a paragraph of text. here is a paragraph of text. here is a paragraph of text. here is a sentence that is a link + some fine styling here is a paragraph of text. here is a paragraph of text. here is a paragraph of text.

this is a webpage of wonderful text

here is another paragraph of text. here is another paragraph of text. here is another paragraph of text. here is another paragraph of text. here is another paragraph of text. here is another paragraph of text. here is another paragraph of text. here is another paragraph of text. here is another paragraph of text. here is another paragraph of text.
- Terminal:** Shows the command prompt for a local HTTP server:

```
~/Desktop/teach/hntr/_s2019/week02 — (...) — python -m SimpleHTTPServer
Last login: Fri Feb  1 09:37:12 on console
~/Desktop/teach/hntr/_s2019/week02 — (...) — python -m SimpleHTTPServer
You have mail.
week02 ~ cd /Users/Rebecca/Desktop/teach/hntr/_s2019/week02
week02 ~ ls
week02 ~ index.html
week02 ~ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
127.0.0.1 - - [03/Feb/2019 17:56:31] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [03/Feb/2019 17:56:31] code 404, message File not found
127.0.0.1 - - [03/Feb/2019 17:56:31] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [03/Feb/2019 18:01:16] "GET / HTTP/1.1" 200 -
```

Web Dev Workflow: Text Editor (Sublime), Local HTTP Server (Terminal), Chrome



tcp ports

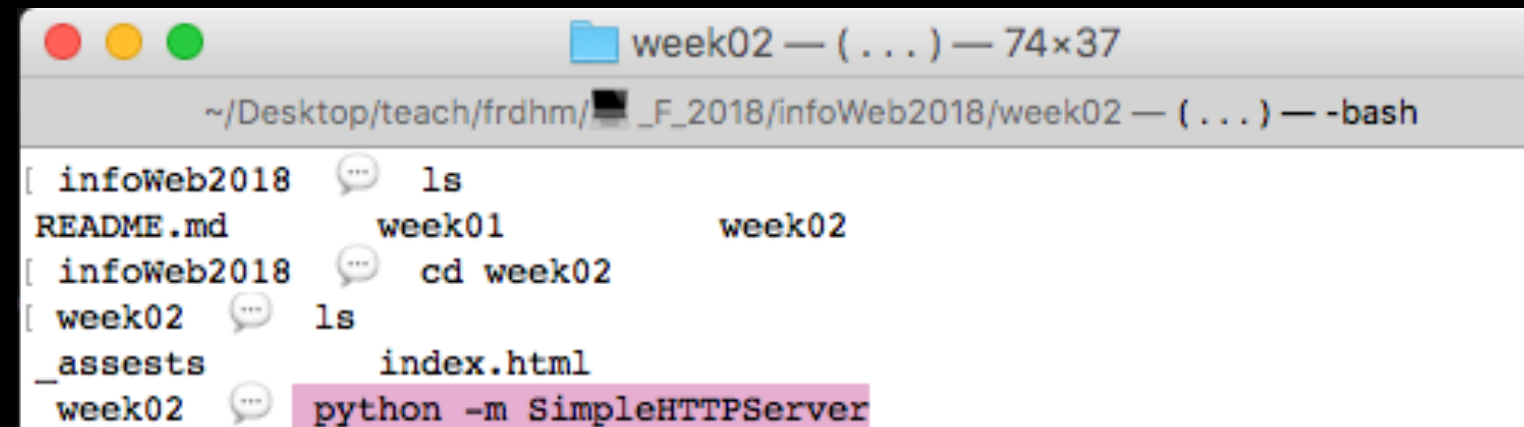
in Terminal we are speaking Unix :

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

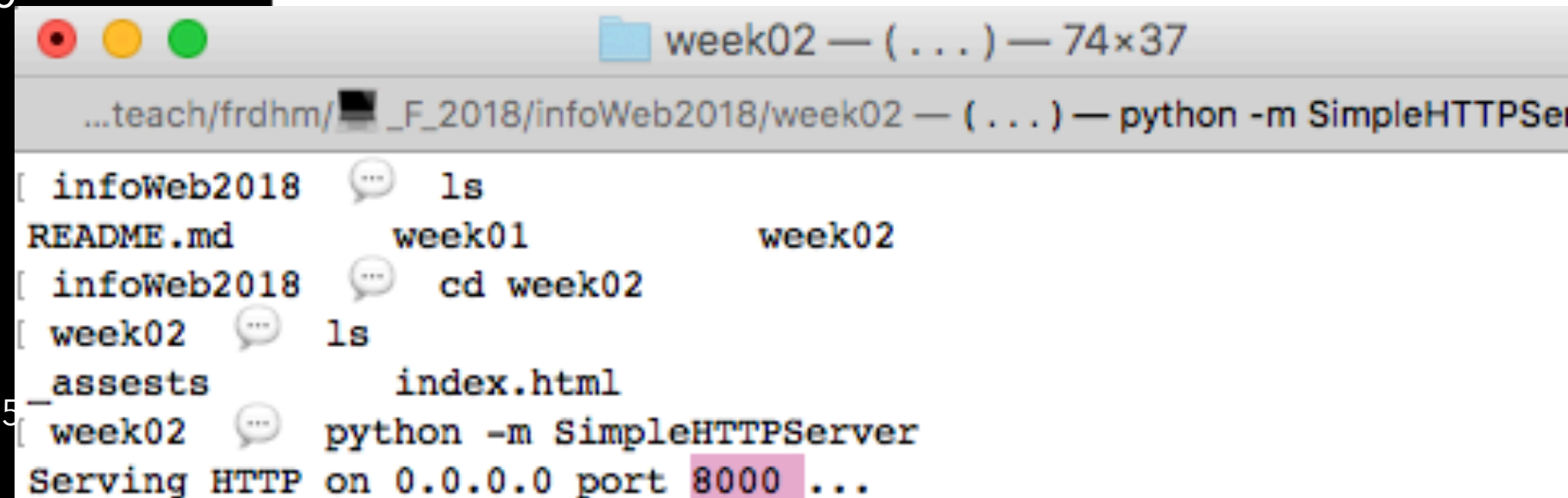
Running a local Python HTTP Server
in Mac OS - this is very simple :

When inside yr project folder simply
type the following command:

"python -m SimpleHTTPServer"
– defaults to port 8000



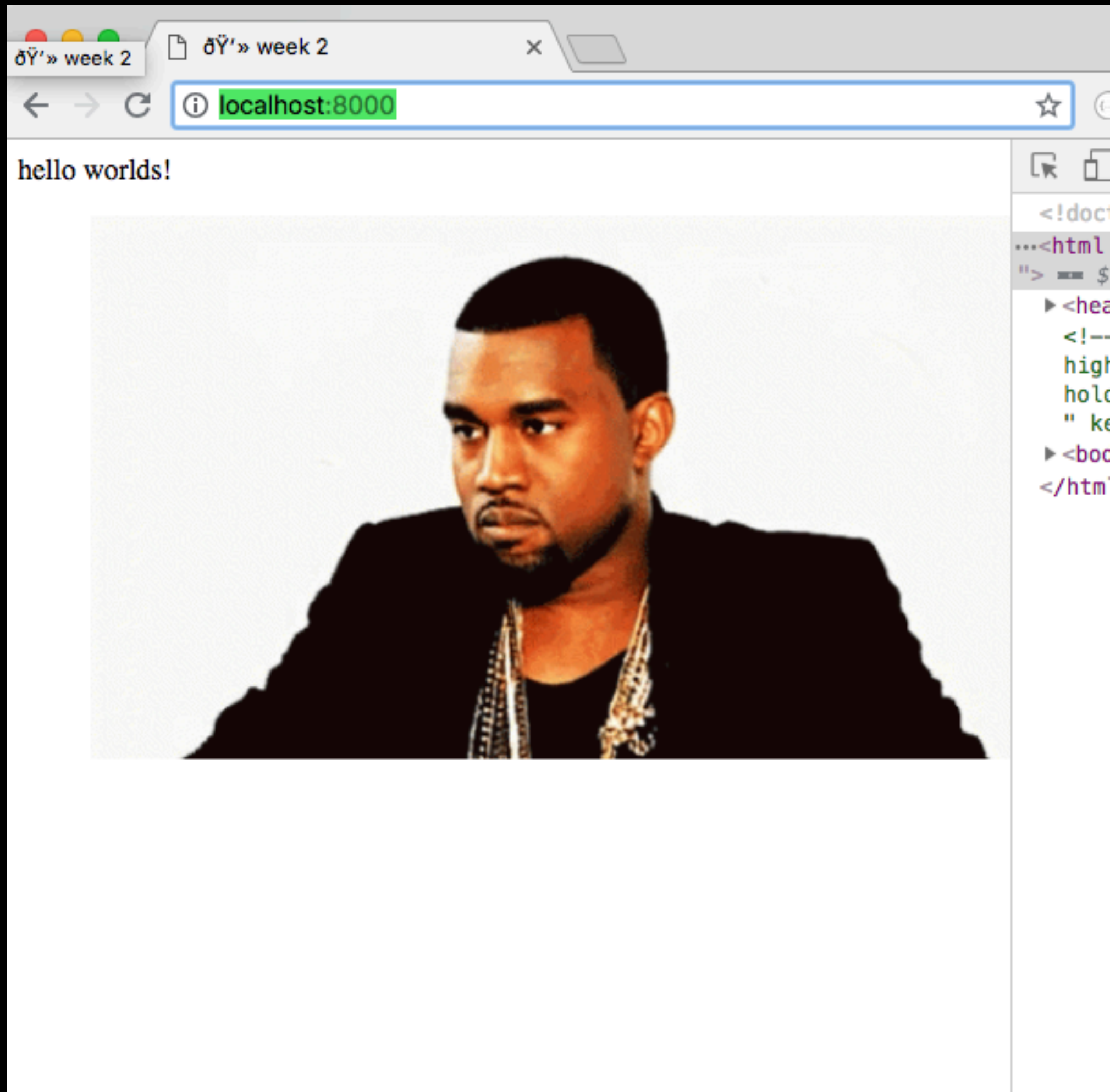
```
week02 — (...) — 74x37
~/Desktop/teach/frdhn/_F_2018/infoWeb2018/week02 — (...) — -bash
[ infoWeb2018 ... ls
README.md      week01          week02
[ infoWeb2018 ... cd week02
[ week02 ... ls
_assests       index.html
week02 ... python -m SimpleHTTPServer
```



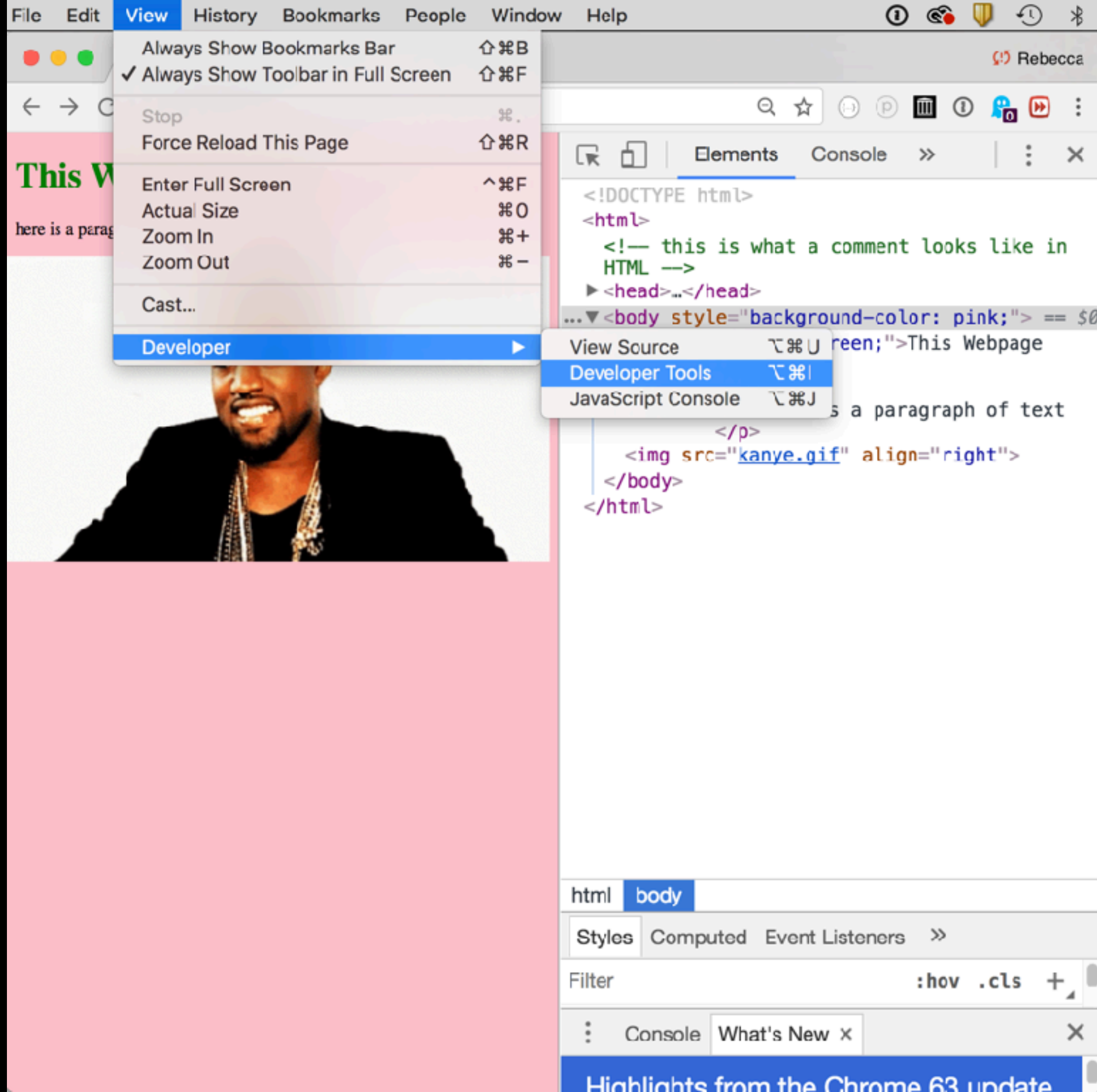
```
week02 — (...) — 74x37
...teach/frdhn/_F_2018/infoWeb2018/week02 — (...) — python -m SimpleHTTPServer
[ infoWeb2018 ... ls
README.md      week01          week02
[ infoWeb2018 ... cd week02
[ week02 ... ls
_assests       index.html
week02 ... python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

if we wrote:

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



url is:
localhost:8000



- + Terminal
- + Text Editor
- + Browser
- + Dev Tools

< HTML >

Hypertext Markup Language

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

< HTML >

3 categories of HTML elements

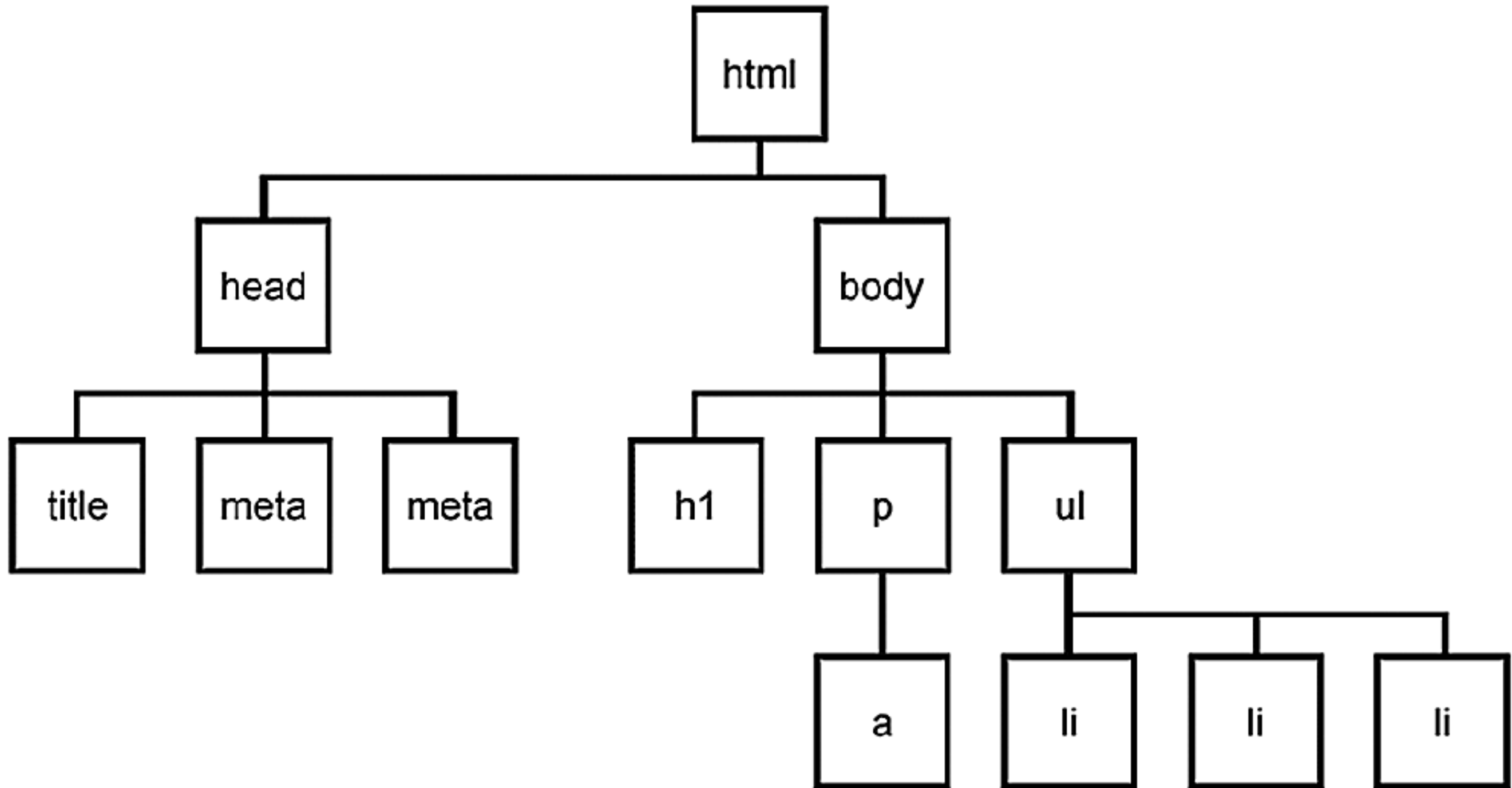
1 - **block**: large blocks of content has height + width
<p>, <h1>, <blockquote>, , , <table>

2 - **inline**: small about of content, no height or width
**<a>, , ,
**


a. **inline block**: inline content w/ height + width

3 - **metadata**: information about the page, usually not visible
<title>, <meta>, <script>

Parent / Child Element Structure

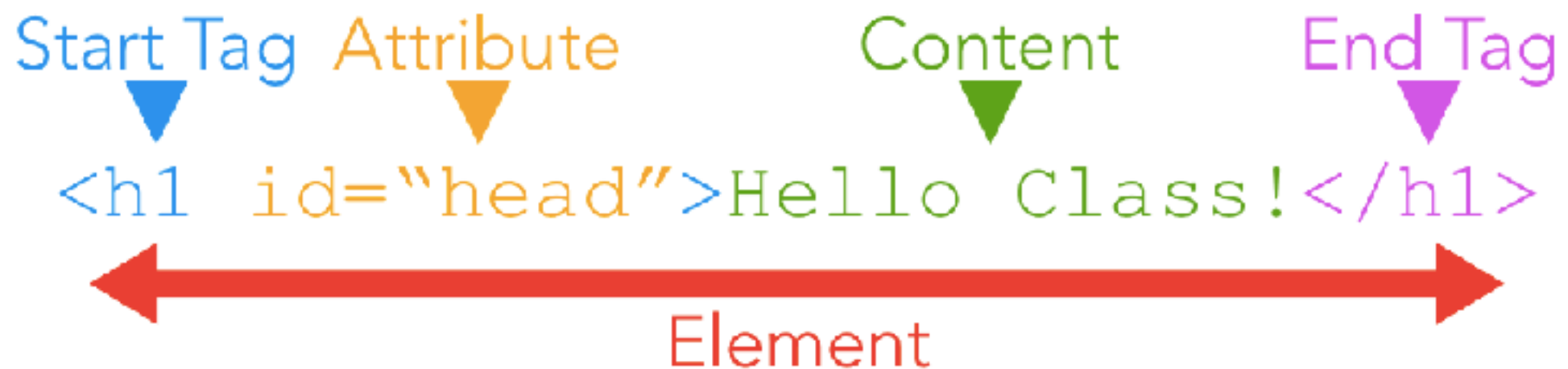


HTML - Hyper Text Mark Up

```
<!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 2 </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

**** is for **bold** **** is for *italics*

****, ****, **** are used for making lists

- **: unordered lists

- **: ordered lists

- **: 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>
```

<p>

<h1> - <h6>

<article>

A document, page or site. This is usually a root container element after body

<section>

Generic section of a document

<header>

Intro section of a document

<footer>

Footer at end of a document or section

<nav>

Navigational section

Use these **before** div when appropriate.

Structure of a link

OPENING
LINK TAG

URL WE ARE
DIRECTED TO

CLOSING
TAG

```
<a href="https://developer.mozilla.org/en-US/docs/Web/CSS" target="_blank">LINK!</a>
```



LINK

TEXT WE
CLICK ON

< a href — stands for *hyperlink reference*

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>
```

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
1  <!doctype html>
2  <html>
3    <head>
4      <title>Tables</title>
5    </head>
6    <body>
7      <!-- basic table structure -->
8      <table>
9        <tr>
10         <td>1</td>
11         <td>2</td>
12         <td>10</td>
13       </tr>
14       <tr>
15         <td>3</td>
16         <td>4</td>
17         <td>11</td>
18       </tr>
19       <tr>
20         <td>5</td>
21         <td>6</td>
22         <td>12</td>
23       </tr>
24     </table>
25
26   </body>
27 </html>
```


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	Run		Work	Drinks
Tuesday			Work	Dinner
Wednesday			Relax	Read
		Time off		

User Inputs

Text input

Username: |

Password input

Username:

Password:

Text area

What is your favorite movie to watch?

What is your favorite movie to watch?

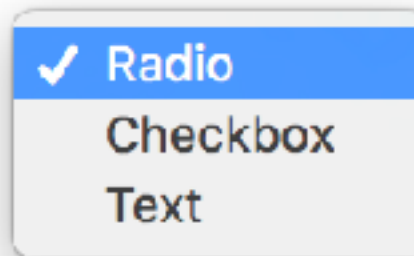
Checkbox

Select your favorite input type:

☐ Radio ☒ Checkbox ☒ Text

Drop down list

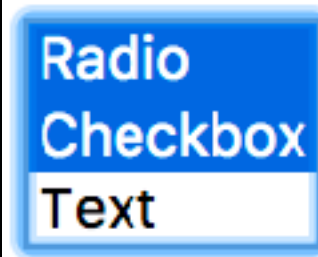
Select your favorite input type:



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', 'Checkbox', and 'Text'. 'Radio' and 'Checkbox' are selected, indicated by a blue background.

Submit button

Are you ready to make that selection?

SUBMIT