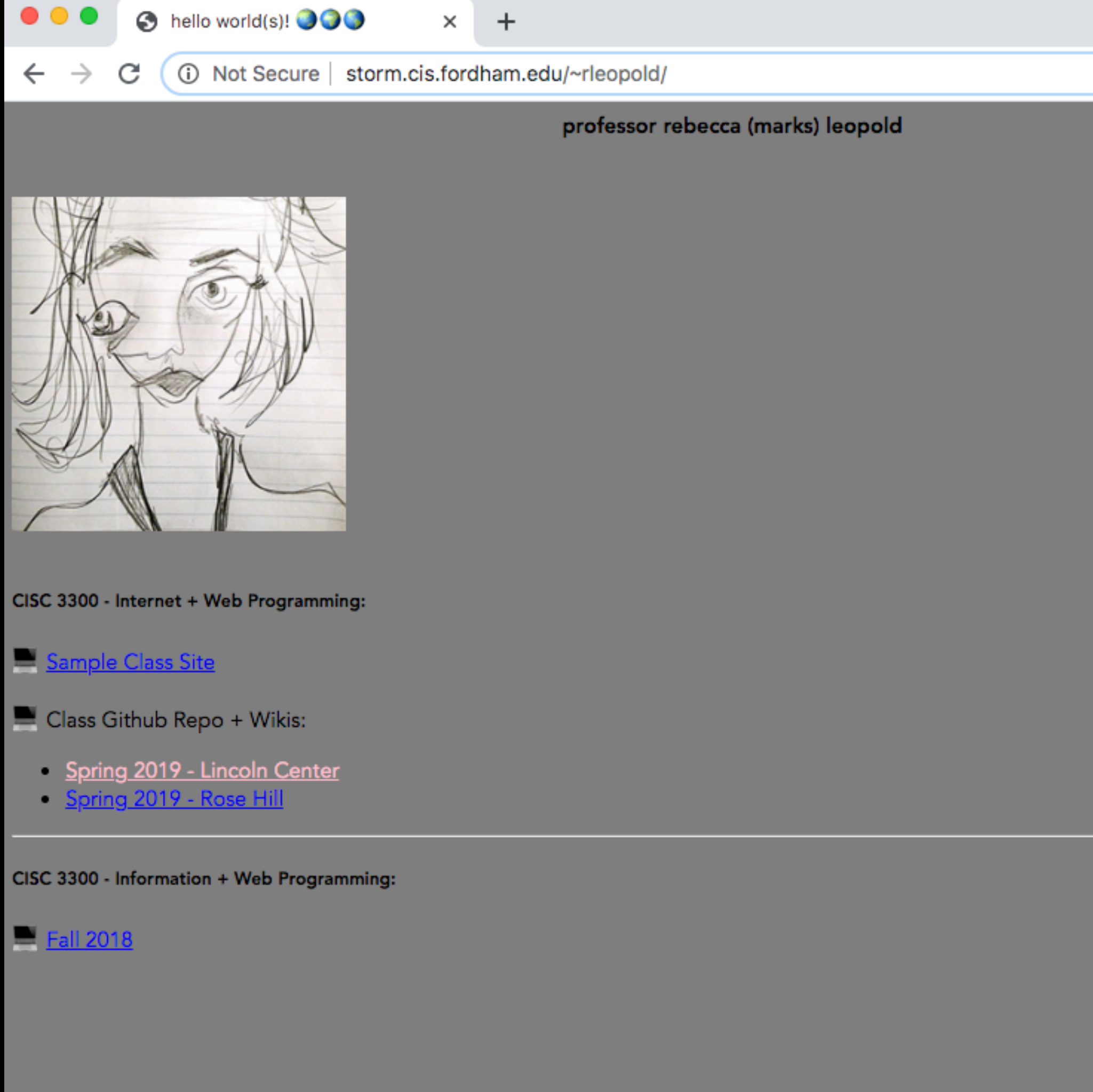


Storm

What is Storm?



# Git

## What is Git?

A version control system meant to make it easier to have multiple versions of code, sometimes across multiple developers or teams

At its most simple, git helps with the '**indexv1.html, indexv2.html, indexv3FINAL.html**' problem

At its most complex, git allows developers to work together worldwide on code without stepping on each other's toes

GitHub

**Github is a service to host yr projects on the web.**

Code is pushed (uploaded) from a local directory (folder) called a repository or rep.

example - our class site:

<http://www.github.com/rebleo/webDevSpring2020>

## Git vs GitHub.com

git is a version control system that takes snapshots of your code at certain points in development

These snapshots are stored in a '**repo**', or '**repository**' on your local machine

GitHub.com is a website that hosts git repositories on a remote server + is available for all the web to see, copy + implement.

# Git Terminology

**repository** - where data is managed. the directory containing your files.

**local** - the copy that exists on your machine, no one else can access this

**remote** - the copy in your github account, anyone with access to your github repo can access the remote instance (we won't be doing this!)

**push** - once you make changes to the local copy you \*upload the changes to the remote copy

**pull** - if someone else makes changes to the remote copy (we won't be doing this this semester)

**clone a repository** - download the entire codebase of the repo you can pull in changes + and push your own changes if you are given access



# Github pages

# github.io

easily allows you to host web pages using github servers + workflow

url (uniform resource locator)

http://

yrUsername.github.io



[yrUsername.github.io](https://yrUsername.github.io)

Unix!



< HTML >

# Hypertext Markup Language

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





Unlike other programming languages - HTML, CSS + JavaScript were authored for their outputs to **be read or seen** by human persons on glowing rectangular screens.



< HTML >

**block** vs. **inline** display

The key to understanding how **HTML** + **CSS** works is to imagine that there is an invisible box around every **HTML** element.

Block level elements are outlined w/ red + inline elements in green.

**<body>** creates 1st box, then **<h1>**, **<h2>**, **<p>**, **<i>** + **<a>** each create their own boxes within it.

## The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

## The Cottage Garden

The *cottage garden* is a distinct style of garden that uses an informal design, dense plantings, and a mixture of ornamental and edible plants.

The Cottage Garden originated in England and its history can be traced back for centuries, although they were re-invented in 1870's England, when stylized versions were formed as a reaction to the more structured and rigorously maintained English estate gardens.

The earliest cottage gardens were more practical than their modern descendants, with an emphasis on vegetables and herbs, along with some fruit trees.

## < HTML >

### 3 categories of HTML elements

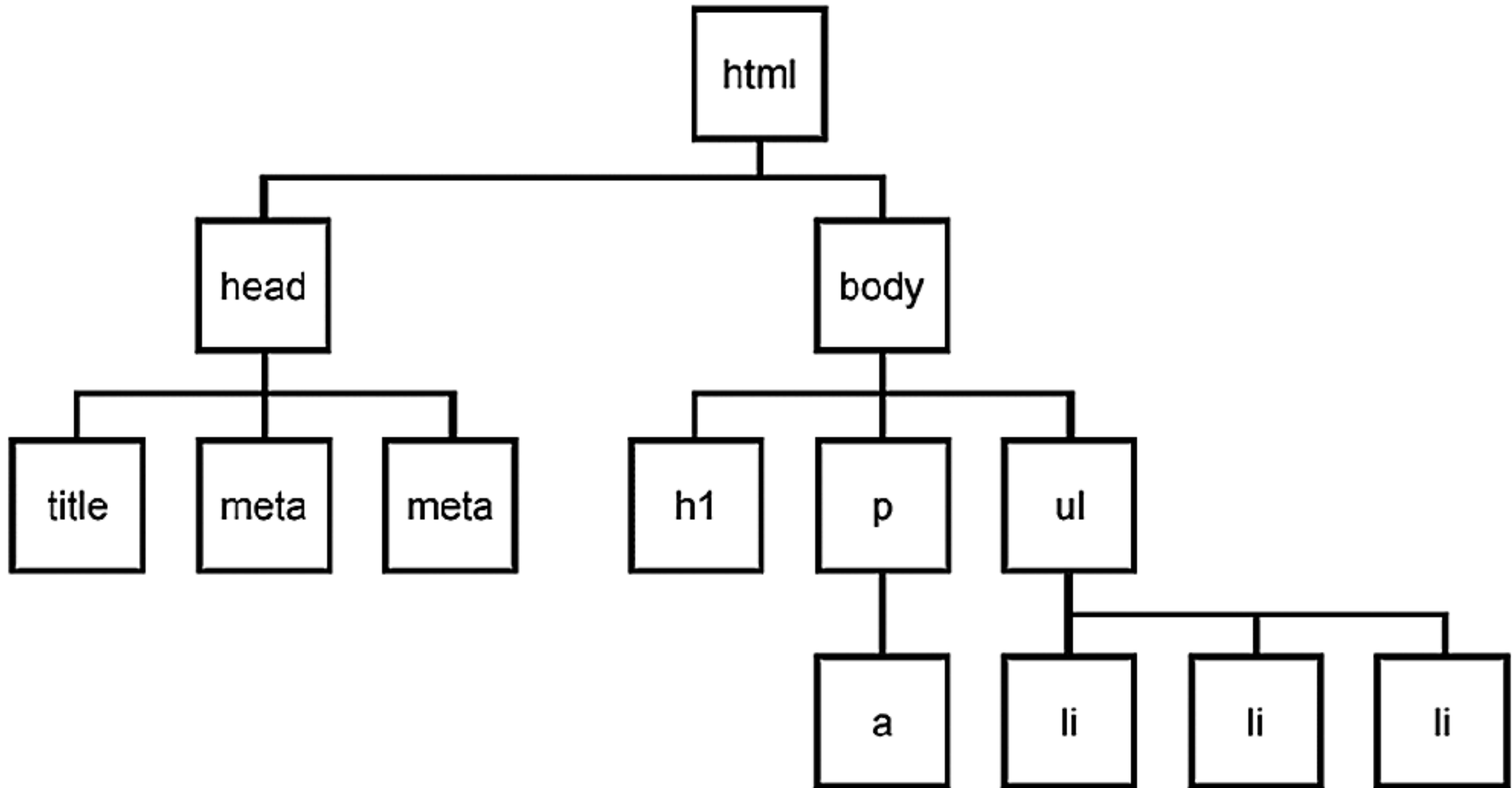
1 - **block**: large blocks of content has height + width  
**<p>, <h1>, <blockquote>, <ol>, <ul>, <table>**

2 - **inline**: small about of content, no height or width  
**<a>, <em>, <strong>, <br>, <span>, <time>**

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

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

# Parent / Child Element Structure



# Parent + Child

```
<!doctype html>
  <head>
    <title> Week 1 </title>
  </head>
  <body>
    <div>
      Here's a Great Site.
    </div>
  </body>
</html>
```

head is the parent of title

div is the child of body

body is the child of html

Structure tags

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.

`<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.

# Semantic HTML

HTML should be coded to represent the data that will be populated and not based on its default presentation styling. Presentation (how it should look), is the sole responsibility of CSS.

Some of the benefits from writing semantic markup are as follows:

- Search engines will consider its contents as important keywords to influence the page's search rankings (see SEO)
- Screen readers can use it as a signpost to help visually impaired users navigate a page
- Finding blocks of meaningful code is significantly easier than searching through endless divs with or without semantic or namespaced classes
- Suggests to the developer the type of data that will be populated
- Semantic naming mirrors proper custom element/component naming

<https://developer.mozilla.org/en-US/docs/Glossary/Semantics>



`<p>`

`<h1>` - `<h6>`

## Semantic elements

`<main>`

dominant content of the `<body>` element

`<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.

# Semantic elements

## `<aside>`

represents a portion of a document whose content is only indirectly related to the document's main content. Asides are frequently presented as sidebars or call-out boxes.

## `<details>`

creates a disclosure widget in which information is visible only when the widget is toggled into an "open" state.

## `<figcaption>`

represents a caption or legend describing the rest of the contents of its parent `<figure>` element.

## `<mark>`

represents text which is marked or highlighted for reference or notation purposes, due to the marked passage's relevance or importance in the enclosing context.

## `<summary>`

element specifies a summary, caption, or legend for a `<details>` element's disclosure box. Clicking the `<summary>` element toggles the state of the parent `<details>` element open and closed.

## `<time>`

represents a specific period in time.

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

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

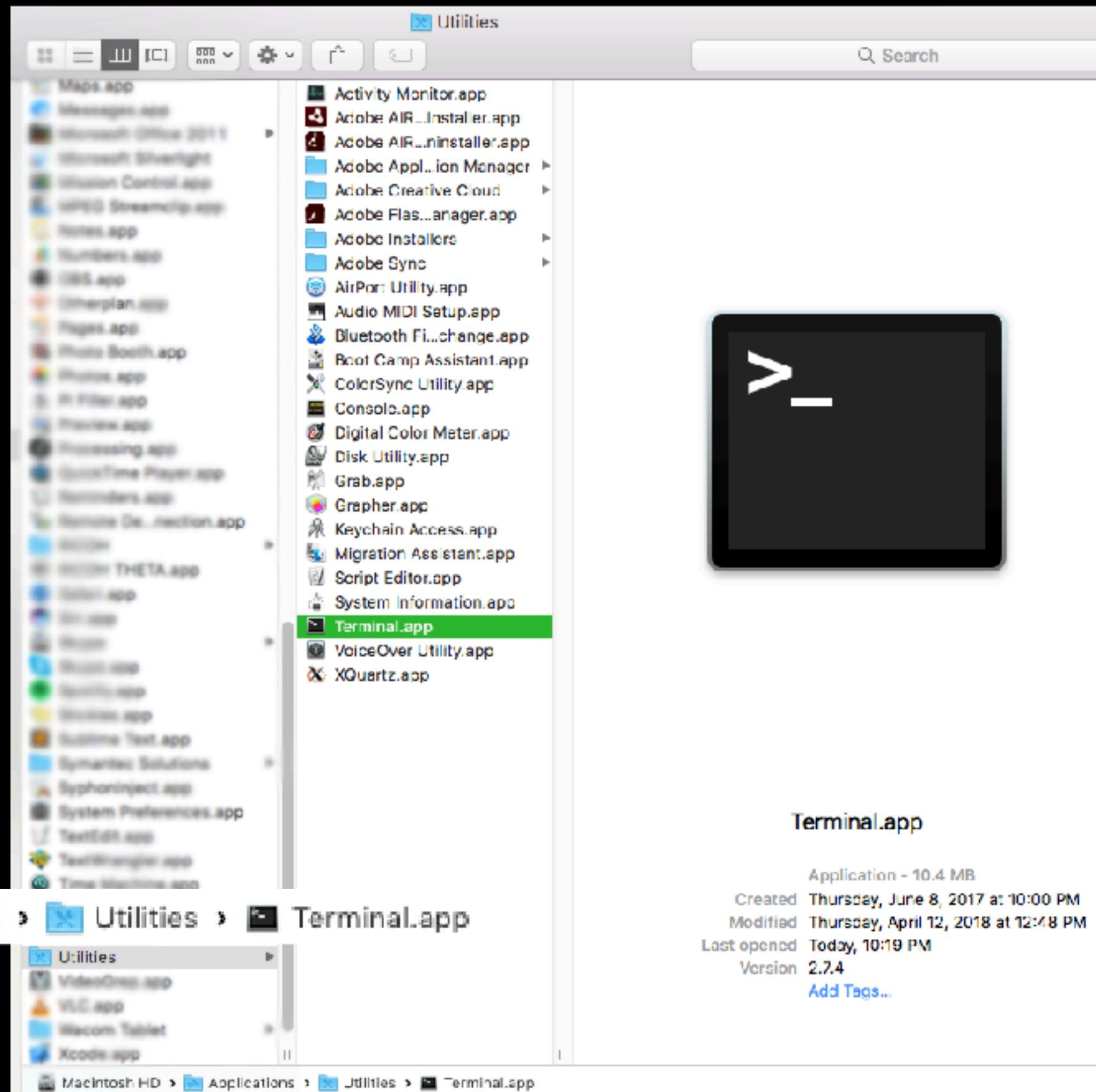
BBC FOUR

What was real was what you could  
put your hands on.

for mac users -  
to open your terminal

the file path is:

Macintosh HD > Applications > Utilities > Terminal.app



on the command line we are asking our computer questions and telling it what to do in Unix :

- **pwd** - "present working directory" === *where am I!?*
- **cd** - "change directory" === *go inside this folder please.*
- **ls** - "list items in this directory" === *what's in this folder?*
- **rm** - "remove file" === *GET RID OF THIS FOREVER* (doesn't work w/ directories)
- **mkdir** - "make directory" === *make a new folder*
- **touch** - "create file" === *create a name.fileExtension (index.html)*

When in doubt - don't forget to ask the interwebs! [Example](#).

```
infoWeb2020 — (...) — 100x43
~/Desktop/teach/frdhn/2020/infoWeb2020 — (...) — bash

~  pwd
/Users/Rebecca
~  ls
Applications      Library           Sites             hs
Creative Cloud Files  Movies           goodnight_log.txt  hs
Desktop            Music            hs_err_pid1560.log  hs
Documents          Pictures         hs_err_pid18370.log nl
Downloads          Public           hs_err_pid19169.log po

~  cd Desktop
Desktop  ls
Scott McCloud - Understanding Comics.pdf      teach
Screen Shot 2020-09-03 at 2.37.02 PM (2).png   think
Screen Shot 2020-09-03 at 2.37.02 PM.png       untitled folder
etc                                              work
itp

Desktop  cd teach/frdhn/2020
2020     ls
fridge           infoWeb2020       week02
img.key          leopold_CISC_2350_R01.pdf  week03.key
info+Web_Fall2020.pages  week01
2020     cd infoWeb2020/
infoWeb2020  ls
CISC_2350_R01_F2020_Syllabu.pdf  schedule.md       week02
README.md          week01
infoWeb2020  mkdir week03
```



When in doubt - don't forget to ask the interwebs! [Example.](#)

The screenshot shows a Google search interface. The search bar contains the text "command to create file in unix". Below the search bar, the Google logo is visible on the left, and navigation links for "All", "Videos", "Images", "News", "Shopping", and "More" are on the right. The search results indicate "About 58,200,000 results (0.58 seconds)".

The first featured snippet is titled "Creating a File with cat Command". The text reads: "To create a new file run the cat command followed by the redirection operator > the name of the file you want to create. Press Enter type the text and once you are done press the CTRL+D to save the files. May 10, 2019". The source is cited as "linuxize.com > post > create-a-file-in-linux". The link title is "How to Create a File in Linux | Linuxize".

Below the featured snippet is a section titled "People also ask" with five questions:

- How do you create a file in Unix?
- What is the command to create a file in Linux?
- How do you create a file using a CAT command in Unix?
- How do I make a file executable in Unix?

The second search result is from "www.cyberciti.biz > faq > unix-create-file-from-termina...". The title is "Unix Create a File Command - nixCraft". The text says: "Oct 6, 2013 - What's the easiest and best way to create a file in Unix? UNIX operating system provides many command line tools and text editors for creating ...".

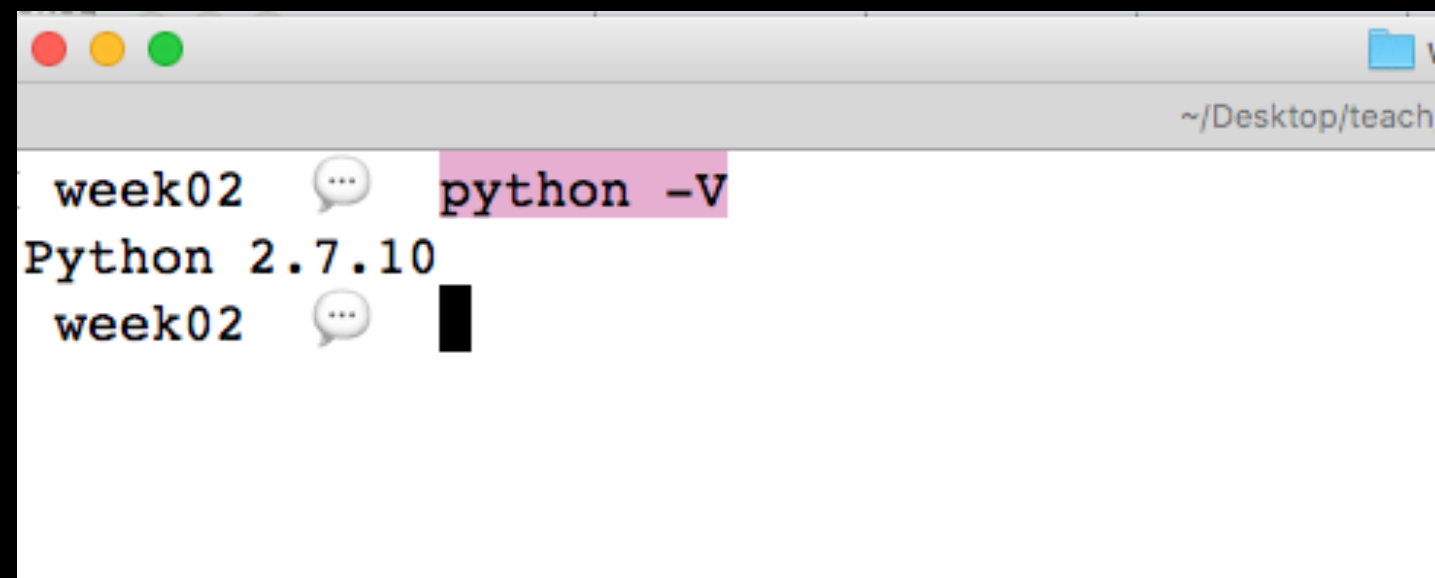
The third search result is from "phoenixnap.com > how-to-create-a-file-in-linux". The title is "How to Create a Linux File Using the Command Line (8 Easy ...". The text says: "Jun 27, 2019 - Create a File with Touch Command The easiest way to create a new file in Linux is by using the touch command. The ls command lists the contents of the current directory."

We also use the command line to run a local server when prototyping websites.

On a Mac this is easy.

Step 1. Check which version of Python is on yr machine

# python -V



```
week02  ... python -V
Python 2.7.10
week02  ...
```

The image shows a macOS terminal window with a title bar containing red, yellow, and green window control buttons. The address bar at the top right shows the path ~/Desktop/teach. The terminal content shows a prompt 'week02' followed by a command 'python -V' which has been executed, resulting in the output 'Python 2.7.10'. A second prompt 'week02' is visible on the next line with a cursor.



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

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

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

otherwise: **python -m SimpleHTTPServer**

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

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

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

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... )

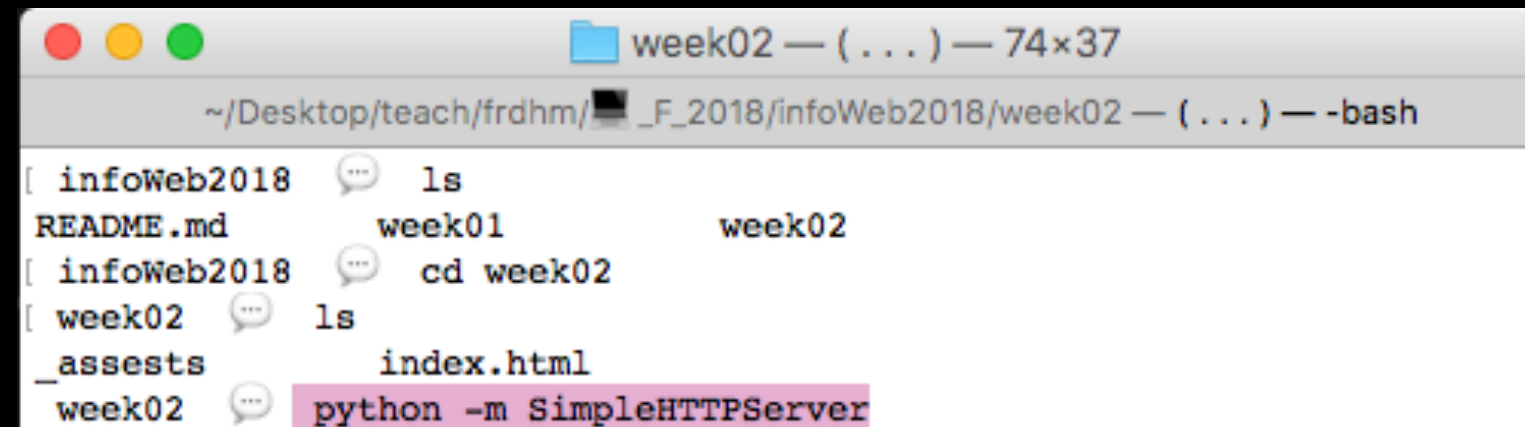
Running a local **Python** HTTP Server  
in Mac OS - this is very simple :

When inside yr project folder simply  
type the following command:

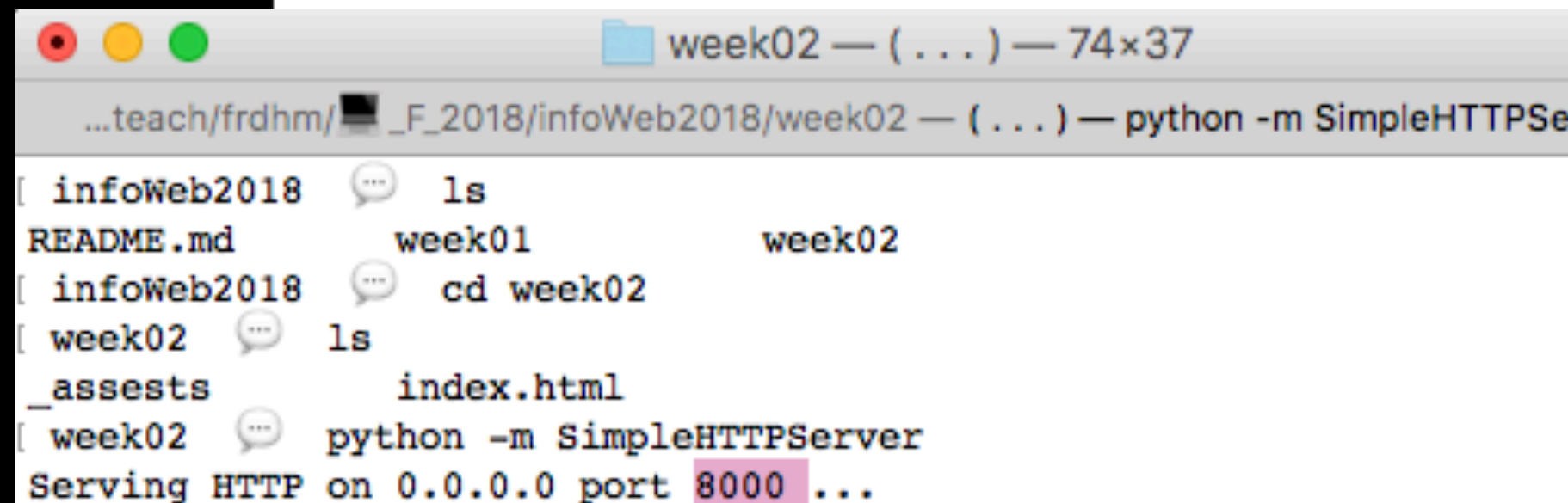
**"python -m SimpleHTTPServer"**  
– defaults to port 8000

if we wrote:

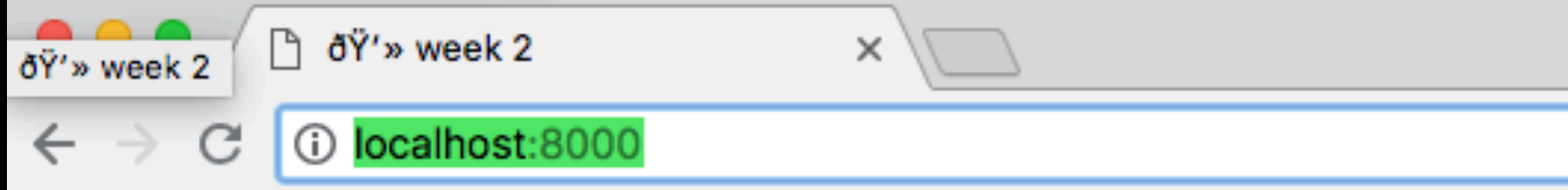
**"python -m SimpleHTTPServer 12345"**  
- we would go to port 12345



```
week02 — (...) — 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 ...
```



url is:

**localhost:8000**