

launch _code

LC101 2.3



Class Agenda

1. Announcement (first assignment)
2. Last Class Review
3. New Material
4. Studio (Flicklist 1)




Announcement

First Assignment is due this coming Monday!

HTML me something!

You will need to have your final product submitted on Github and your TF will check your project in person on Monday.

It's just about creating an HTML document, and using Git to push it to your remote github repository. Your actual project, won't contain much HTML.



Review

CSS - Inline, Internal, External (style="color: red")

Git - local repo vs. remote repo, branches



URL - Base Pieces

URL - Uniform Resource Locator

<https://www.launchcode.org/learn>

Protocol (https)

Host (www.launchcode.org) (your DNS changes this readable name into an IP address)

Path (/learn)



URL - optional pieces

<https://www.launchcode.org/learn?p=15#footer>

Protocol (https) - what type of connection we are making

Host (www.launchcode.org)

Path (/learn)

Query parameters (p=15)

Fragment (#footer)



URL - pieces defined

Protocol - how we are connecting

Host - Who we are connecting to

Path - what folder or resource we are requesting

*Query params - Additional information we are seeking, or the server needs

*Fragment - The specific part of the resource we need



HTTP

HTTP - Hyper Text Transfer Protocol, it's a set of rules your computer follows to communicate with other computers. It defines what requests, and responses look like, and the information they contain.

They use Request Lines, Response Lines, Methods, and Headers to pass information between Client and Server.



HTTP - Request

<http://www.launchcode.org/learn>

Request Line for this URL is:

GET /learn HTTP/1.1

This is what your computer sends to the host/server (www.launchcode.org)



HTTP - Response

Client makes request - GET /learn HTTP/1.1

Server response - HTTP/1.1 200 OK



Client - Server


HTTP defines how 2 computers communicate.
The client sends a request, and the server replies with a response.

Request - GET /learn HTTP/1.1

Response - HTTP/1.1 200 OK

Each request and response has a header
(additional information)

• The response header is what contains the actual HTML.



Disclaimer

There is a lot more to how HTTP works, how the internet works, how client server relationships work. Our prep work did a good job of introducing us to some of the basics, but there is still a lot we will not learn in this course.

I expect you to know: URL, HTTP request, and response, the basics of a server/client relationship.



Python - Flask

From the first couple of classes we learned how a client renders a website. The browser renders an HTML document to display content in a specific place. CSS can be used to change the style of the content.

We will now change our focus on how servers work using Python, and the Flask Framework.



Python - Flask

Flask is a web framework for Python.

You import flask like you would any package we have worked with so far using the import statement.

Flask is specifically set up to help you handle Requests from a client, and to respond in kind with an HTML document..



Let's See an Example

1. Make a new directory
2. Change into that directory
3. Create new virtual environment with Conda
4. Activate the new virtual environment
5. Create a new .py file that will contain the flask information



Conda Cheatsheet

Personally, I don't use Conda. However it is the same across all platforms so it makes a lot of sense for this class.

I will be referencing the Conda cheatsheet extensively...

https://conda.io/docs/_downloads/conda-cheatsheet.pdf



After the Studio

If you finish the studio early today, I would recommend doing a few things in preparation for the rest of this unit.

1. Go over the basics of HTML
2. Go over the Git commands we have used so far
3. Create a few conda environments and practice activating and deactivating them
4. Create a few new Flask projects, to get practice going through the setup steps



Studio - Flicklist

Over the next few classes we will be building a project with Flask, it's called Flicklist.

Essentially flicklist is going to be a list of movies we can access from our web browser using HTML, and Flask!



FlickList Setup

1. Make new directory
2. Change to new directory
3. Clone repo (git clone <https://github.com/LaunchCodeEducation/flicklist-flask.git>)
4. Create virtualenv with Conda (conda create -n flicklist)
5. Install flask into this new environment (conda install flask)
6. Work on your own through the rest of the studio, by checking out a different branch.

