



Web Scrapping With HTML & CSS

Data Boot Camp
Lesson 10.2



The Big Picture



Module 10

This Week: Web Scraping with HTML & CSS

This Week: Web Scraping with HTML & CSS

By the end of this week, you'll know how to:



Gain familiarity with and use HTML elements, as well as class and id attributes, to identify content for web scraping



Use BeautifulSoup and Splinter to automate a web browser and perform a web scrape



Create a MongoDB database to store data from the web scrape



Create a web application with Flask to display the data from the web scrape



Create an HTML/CSS portfolio to showcase projects



Use Bootstrap components to polish and customize the portfolio



This Week's Challenge

Using the skills learned throughout the week, create a web app that displays scraped images of Mars' hemispheres, complete with titles.

Module 10

Today's Agenda

Today's Agenda

By completing today's activities, you'll learn the following skills:

01

Flask templates

02

Saving data with PyMongo

03

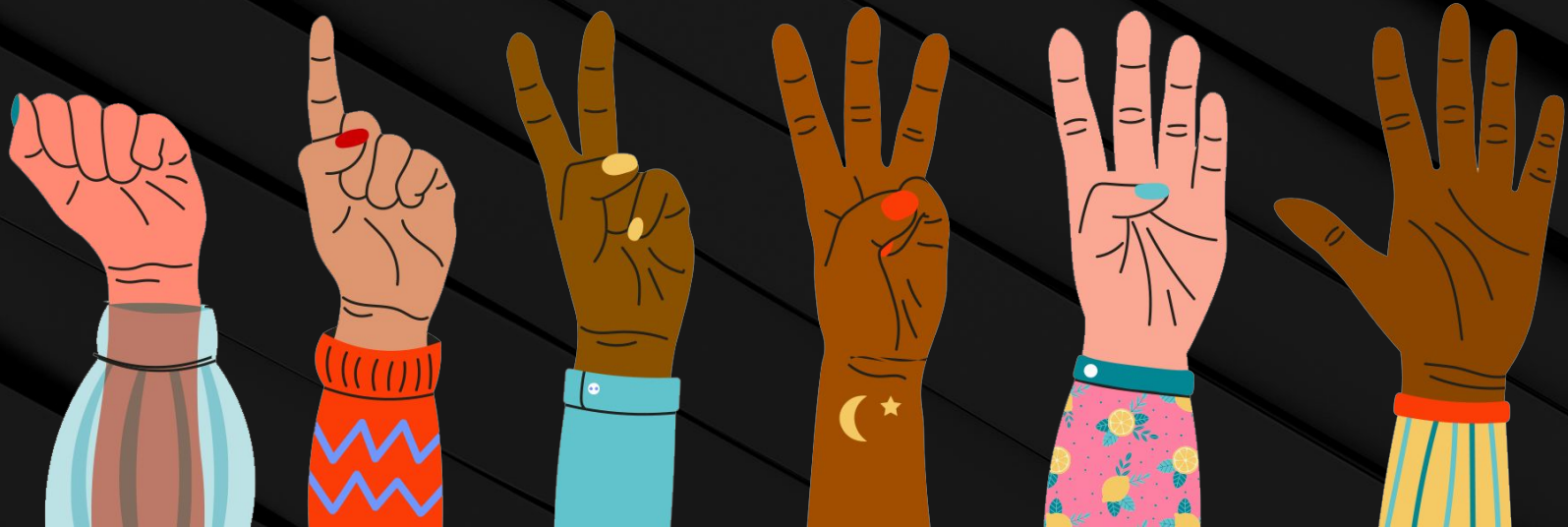
Creating a web page using this module's tools



**Make sure you've downloaded
any relevant class files!**

FIST TO FIVE:

How comfortable do you feel with this topic?



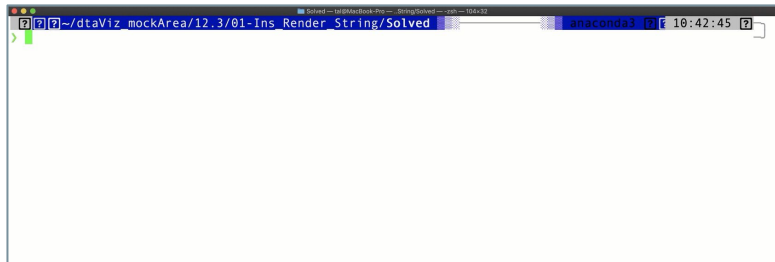


Instructor Demonstration

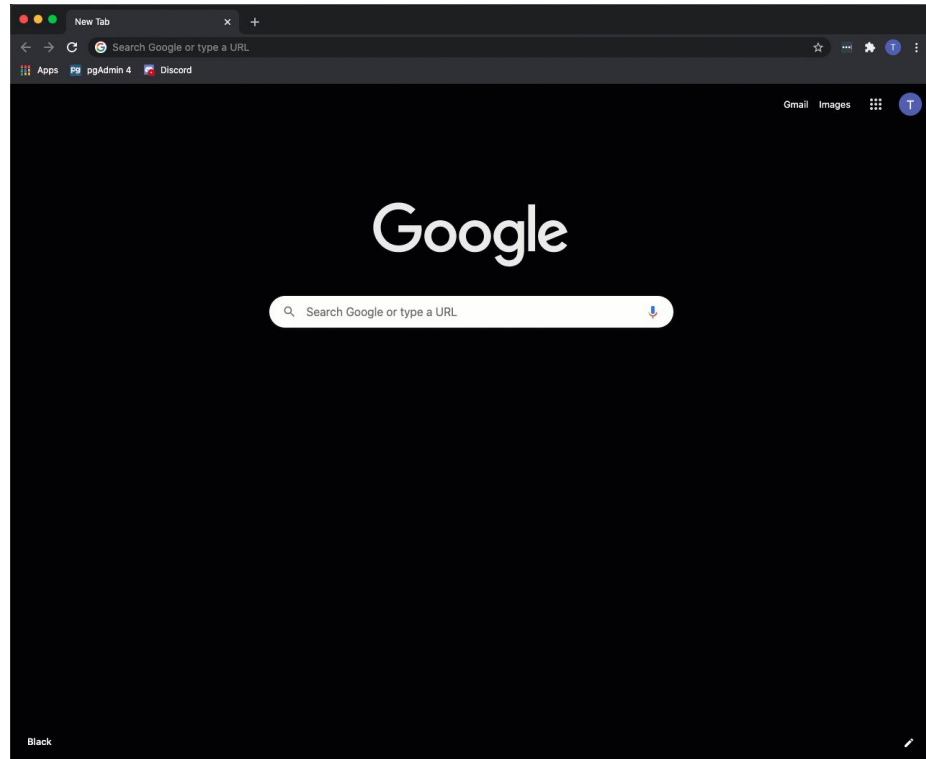
Intro to Flask Render

Intro to Flask Render

How it works—the basics of rendering a template with Flask!




1. In your CLI, navigate to the appropriate folder.
2. Run: `python app.py`
3. Open your browser and visit



Intro to Flask Render

/app.py
/templates
 /index.html



```
1 # import necessary libraries
2 from flask import Flask, render_template
3
4 # create instance of Flask app
5 app = Flask(__name__)
6
7
8 # create route that renders index.html template
9 @app.route("/")
10 def echo():
11     return render_template("index.html", text="Serving up cool text from the Flask server!!")
12
13
14 if __name__ == "__main__":
15     app.run(debug=True)
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8   <title>Templates 101</title>
9   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
10 </head>
11
12 <body>
13   <div class="container">
14
15     <div class="jumbotron text-center">
16       <!-- Render our data -->
17       <h1>{{ text }}</h1>
18     </div>
19
20   </div>
21 </body>
22
23 </html>
```



Activity: Rendering a String with Flask

In this activity, you will use Flask to render a welcome message on their page.

Suggested Time:
15 minutes





Let's Review



Instructor Demonstration

Rendering a List and a Dictionary

Rendering a List

/app.py
/templates
 /index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8   <title>Teams!</title>
9   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
10 </head>
11
12 <body>
13   <div class="container text-center">
14     <h1 class="jumbotron">Team Rosters</h1>
15     <div>
16       <ul style="list-style: none;">
17         {% for name in list %}
18         <li>{{ name }}</li>
19         {% endfor %}
20       </ul>
21     </div>
22   </div>
23 </body>
24 </html>
```

```
1 # import necessary libraries
2 from flask import Flask, render_template
3
4 # create instance of Flask app
5 app = Flask(__name__)
6
7
8 # create route that renders index.html template
9 @app.route("/")
10 def index():
11     team_list = ["Jumpers", "Dunkers", "Dribblers", "Passers"]
12     return render_template("index.html", list=team_list)
13
14
15 if __name__ == "__main__":
16     app.run(debug=True)
```

```
{% for name in list %}
  <li>{{ name }}</li>
{% endfor %}
```



Rendering a List and a Dictionary

Suggested Time:

20 minutes



Instructor Demonstration

Scrape, Save, and Render Data

Scrape, Save, and Render Data

/app.py
/scrape_craigslist.py
/templates
/index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Hot Finds</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>

<body>
  <div class="container">

    <div class="jumbotron text-center">
      <h1>Hot Finds On Craigslist</h1>
      <p><a class="btn btn-primary btn-lg" href="/scrape" role="button">Find An Awesome Deal!</a></p>
    </div>

    <!-- Craigslist Listings -->
    <div class="row" id="craigslist-listings">
      <div class="col-md-12">
        <div class="heading">{{listings.price}} {{listings.headline}}</div>
        <small>{{listings.hood}}</small>
      </div>
    </div>
  </div>
</body>
</html>
```

```
1 from splinter import Browser
2 from bs4 import BeautifulSoup
3
4
5 def init_browser():
6     # @NOTE: Replace the path with your actual path to the chromedriver
7     executable_path = {"executable_path": "usr/local/bin/chromedriver"}
8     return Browser("chrome", **executable_path, headless=False)
9
10
11 def scrape():
12     browser = init_browser()
13     listings = {}
14
15     url = "https://raleigh.craigslist.org/search/hhh?max_price=1500&availabilityMode=0"
16     browser.visit(url)
17
18     html = browser.html
19     soup = BeautifulSoup(html, "html.parser")
20
21     listings["headline"] = soup.find("a", class_="result-title").get_text()
22     listings["price"] = soup.find("span", class_="result-price").get_text()
23     listings["hood"] = soup.find("span", class_="result-hood").get_text()
24
25     return listings
```

```
1 from flask import Flask, render_template, redirect
2 from flask_pymongo import PyMongo
3 import scrape_craigslist
4
5 app = Flask(__name__)
6
7 # Use flask_pymongo to set up mongo connection
8 app.config["MONGO_URI"] = "mongodb://localhost:27017/craigslist_app"
9 mongo = PyMongo(app)
10
11 # Or set inline
12 # mongo = PyMongo(app, uri="mongodb://localhost:27017/craigslist_app")
13
14 @app.route("/")
15 def index():
16     listings = mongo.db.listings.find_one()
17     return render_template("index.html", listings=listings)
18
19
20
21 @app.route("/scrape")
22 def scraper():
23     listings = mongo.db.listings
24     listings_data = scrape_craigslist.scrape()
25     listings.update({}, listings_data, upsert=True)
26     return redirect("/", code=302)
27
28
29 if __name__ == "__main__":
30     app.run(debug=True)
```



Time to Code

Scrape and Render

Suggested Time:

30 minutes

Questions?

