6. Flask, HTML, CSS

Web development basics

Logistics ⚙️

- Did you install stuff for today? `pip install flask` is your friend.
- Every group has a project, I think?
- Gradescope problems?
- I’m pushing the Career Day back a week (didn’t get time to coordinate guest speakers), so next week we’ll talk about APIs!

Recap: Debugging

What is the *first* question I'm going to ask you when you ask me why your code isn't working?

(This is a sort of unfair question because I haven't actually shared this yet, but take a guess!)

Recap: Debugging

What is the *first* question I'm going to ask you when you ask me why your code isn't working?

What have you tried so far?

Recap: Python

Discuss: Python is easier/harder

- To write
- To debug
- To read
- To run (in terms of compute time)

Recap: Python

Discuss: Python is easier/harder

- To write
- To debug (???)
- To read (???)
- To run (in terms of compute time)

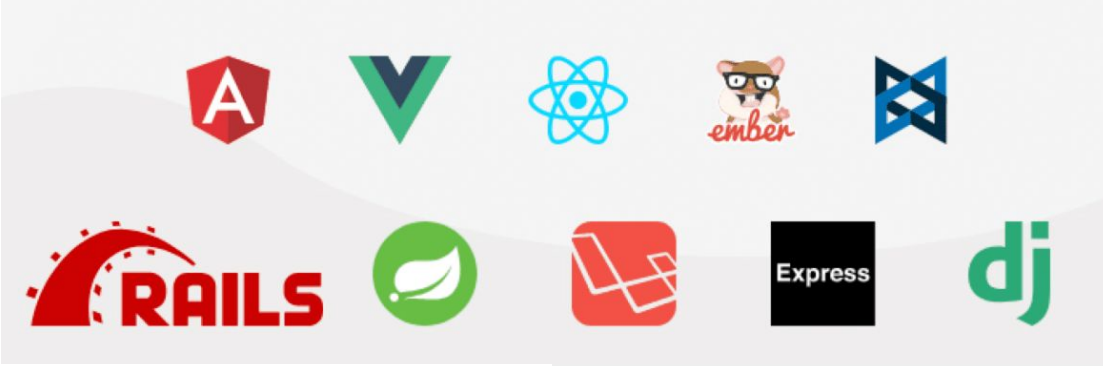
Key points: Python is interpreted, not compiled. Intuitive syntax and library support makes writing easy. Dynamic typing arguably makes reading and debugging harder, but ease of language arguably makes those things easier.

Agenda 📆

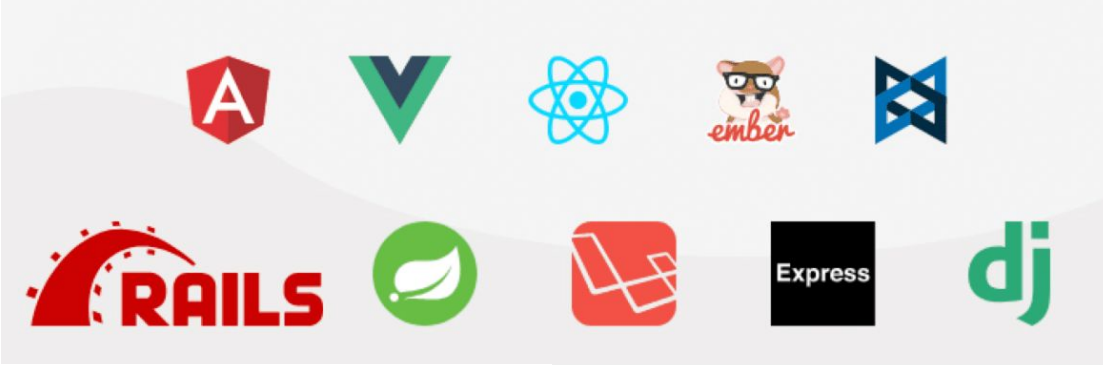
1. Introduction
2. Flask (Demo)
3. HTML (Demo)
4. CSS (Demo)
5. Website Challenge

Why do we care? 🤔

- We’ll be using this tech stack throughout the year.
- Knowing these simple technologies will give you a foundation for building most small-to-medium web projects.



Framework



Framework

Code that provides a structure which makes it easier for a developer to create an application

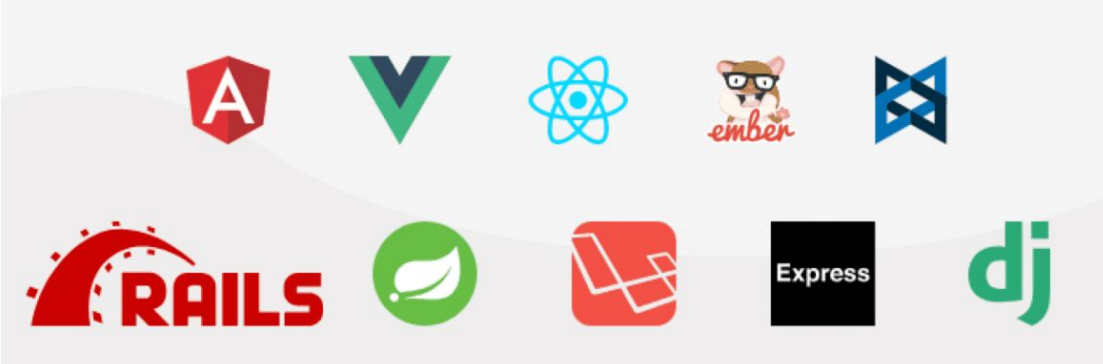
Why use a web framework?

Make it easier to write and scale web applications

- Work with HTTP request and responses
- Routing URLs to appropriate handler
- Interact with databases with less code
- User authorization
- Formatting output (JSON, HTML)

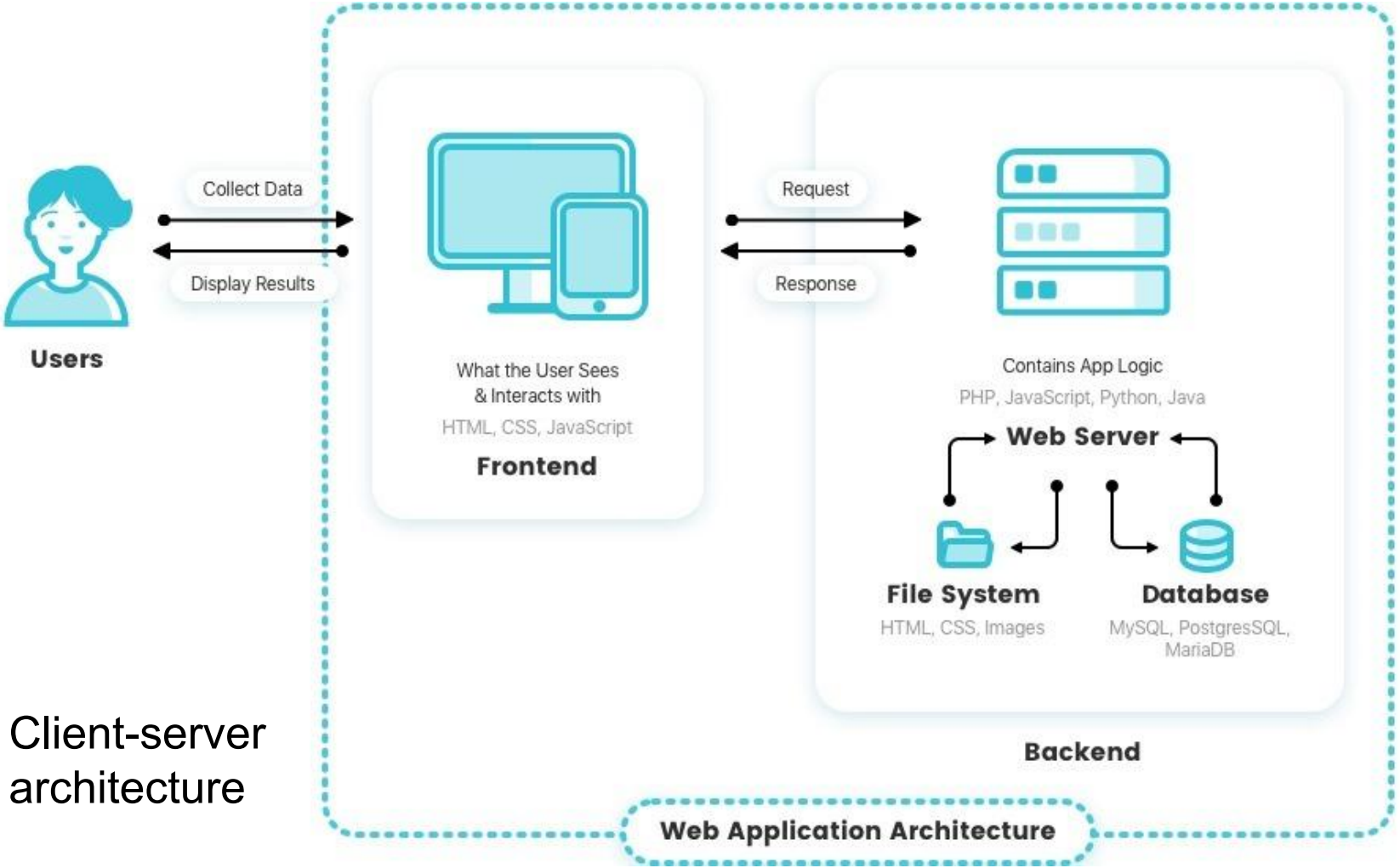
Trade-offs: Which web framework to use?

- Ease - What programming language do you know best?
- Flexibility - How “opinionated” is the design?
- DIY vs. batteries included - Does it come with helpful libraries?
- User friendliness - How is the documentation? How many people use it?

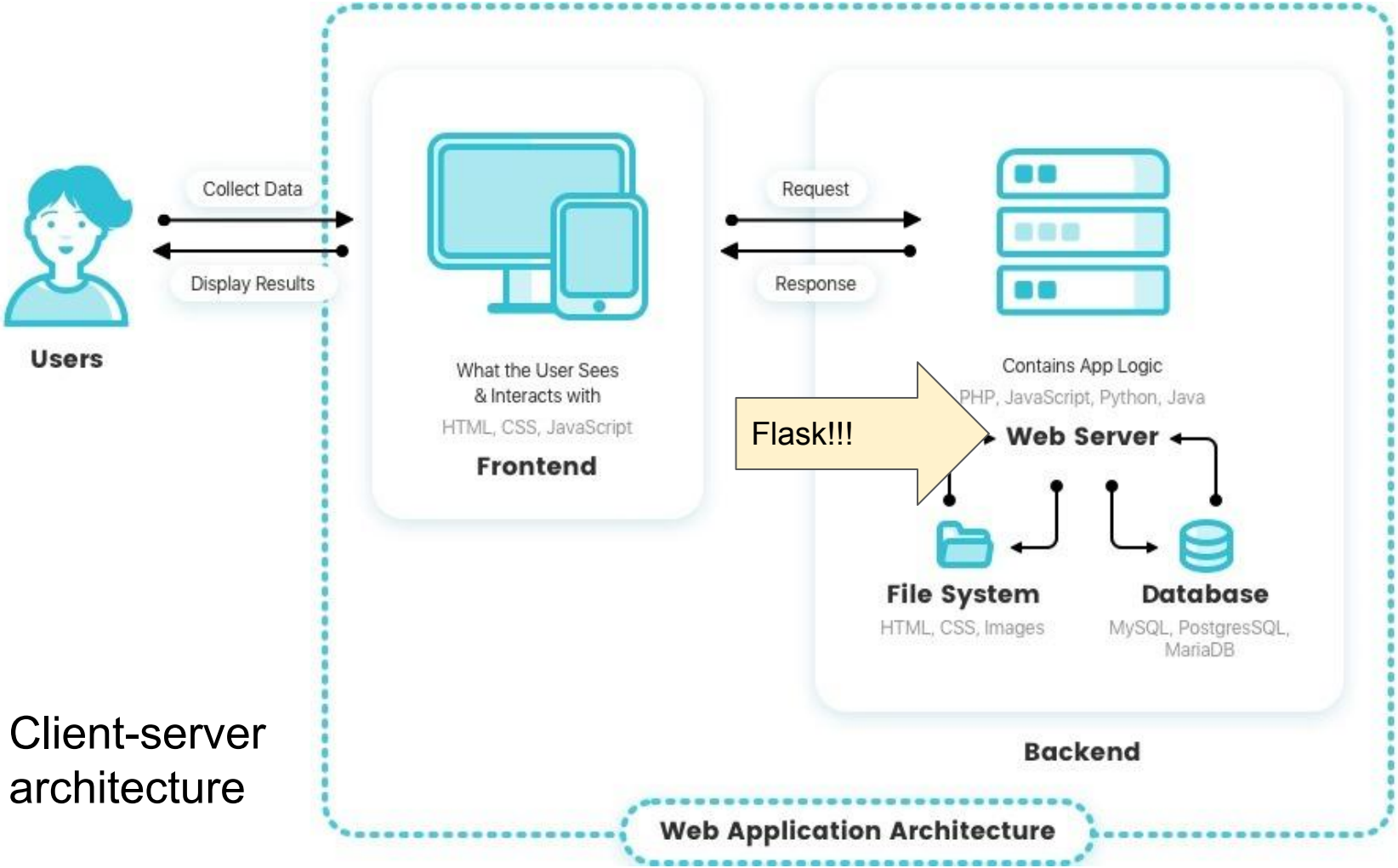


What is Flask?

- Web framework that uses Python
- Helps with routing and fetching, and HTML/CSS for layout
- Almost makes development too easy…
  - “Hello, world” web app is about 10 lines of Python code, and nothing else!!
- Advertises itself as a “micro-framework”
  - The micro- prefix here just means it abstracts over very little (meaning you can decide what to abstract)
  - Core functionality + many extensions



Client-server architecture



Client-server architecture

Installing Flask (what it should look like)

```
vocstartsoft:~/environment/ $ sudo pip install flask #try pip3 if pip does not work!
Collecting flask
  Downloading
https://files.pythonhosted.org/packages/9b/93/628509b8d5dc749656a9641f4caf13540e2cdec85276964ff8f43bbb1d3b/F
lask-1.1.1-py2.py3-none-any.whl (94kB)
    100% |████████████████████████████████| 102kB 7.8MB/s
Collecting click>=5.1 (from flask)
  Downloading
https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257104c434fe0b07e5a195a6847506c074527aa599ec/C
lick-7.0-py2.py3-none-any.whl (81kB)
    100% |████████████████████████████████| 81kB 7.7MB/s
Collecting Werkzeug>=0.15 (from flask)

...

Installed /usr/local/lib/python2.7/dist-packages/MarkupSafe-0.23-py2.7.egg
Finished processing dependencies for Flask
vocstartsoft:~/environment/ $
```

Installing Flask (one of these ways should work…)

- `pip install flask`
- If permission error: **`sudo pip install flask`**
- Replace `pip` with `pip3`
- If pip command not found
  - `which pip` or `which pip3` > Use output directory
  - Replace `pip` with output directory in above command
- Need help installing/using pip? Check these docs
  - https://pip.pypa.io/en/stable/installation/

It’s demo time! 

Flask

Using Flask Framework

```
# lect6.py
import flask

app = flask.Flask(__name__)

@app.route('/') # Python decorator
def index():
    return "Hello, world!"

app.run()
```

Run python lect6.py

```
# lect6.py
import flask

app = flask.Flask(__name__)

@app.route('/') # Python decorator
def index():
    return "Hello, world!"

app.run()
```

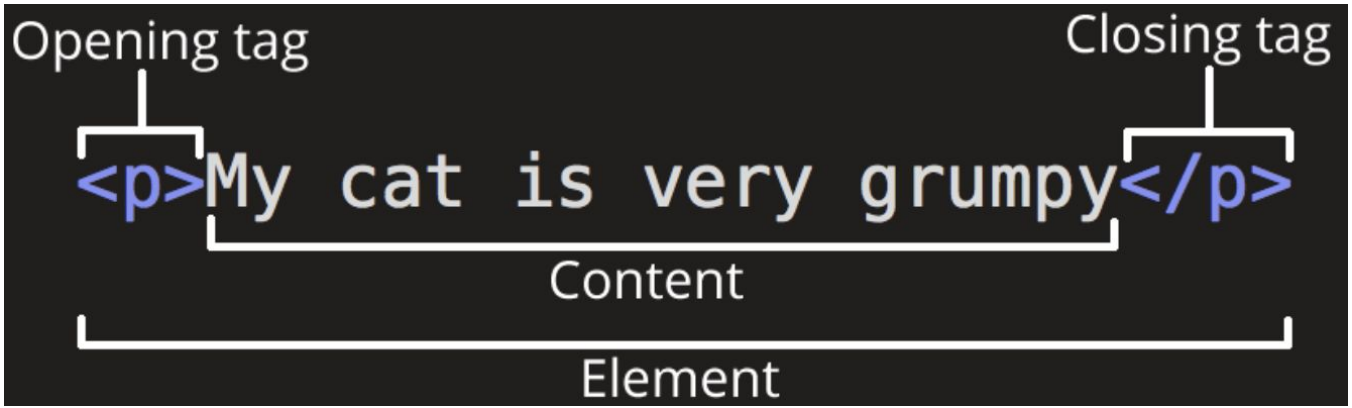
```
vocstartsoft:~/environment/scratch $ python lect6.py
 * Serving Flask app "lect3" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a
production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

HTML

Hyper**t**ext **M**arkup **L**anguage

- Code that is used to structure a web page and its content.
- Not a programming language!
- Content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables.

Anatomy of an HTML **element** 🦴



Anatomy of an HTML **document** 🦴

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

→ **doctype:** Dumb, but needed

→ **html:** Root element

→ **head:** Everything that's **not** content

→ **meta:** The character set for content

→ **title:** That title in your browser tab

→ **body:** All the actual content

Lot’s of useful HTML elements - Text

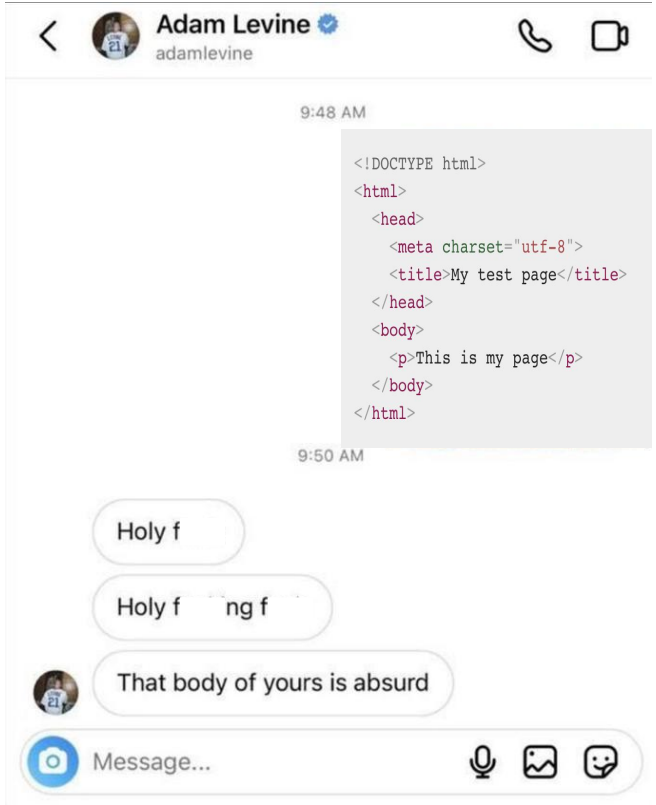
```
<b>Bold text</b>

<i>Italicized text</i>

<ul>
  <li>Aang</li>
  <li>Katara</li>
  <li>Sokka</li>
</ul>

<a href="https://www.google.com">
  Click here
</a>
```

Bold text

Italicized text

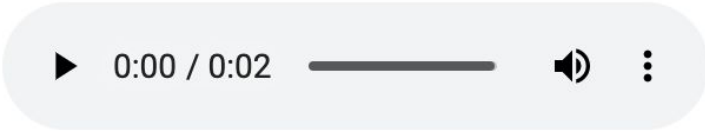
- Aang
- Katara
- Sokka

Click here



Lot’s of useful HTML elements - other

```
<audio src="/media/cc0-audio/t-rex-roar.mp3" />
```




```
<img
src="https://www.example.com/images/dinosaur.jpg">
```

```
<div>Thing 1</div>

<div>Thing 2</div>
```

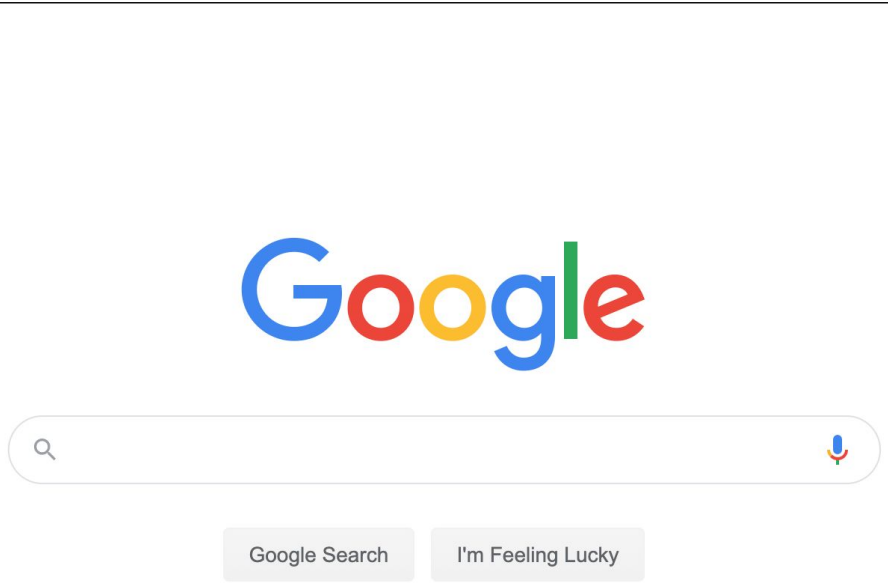
“Templates”

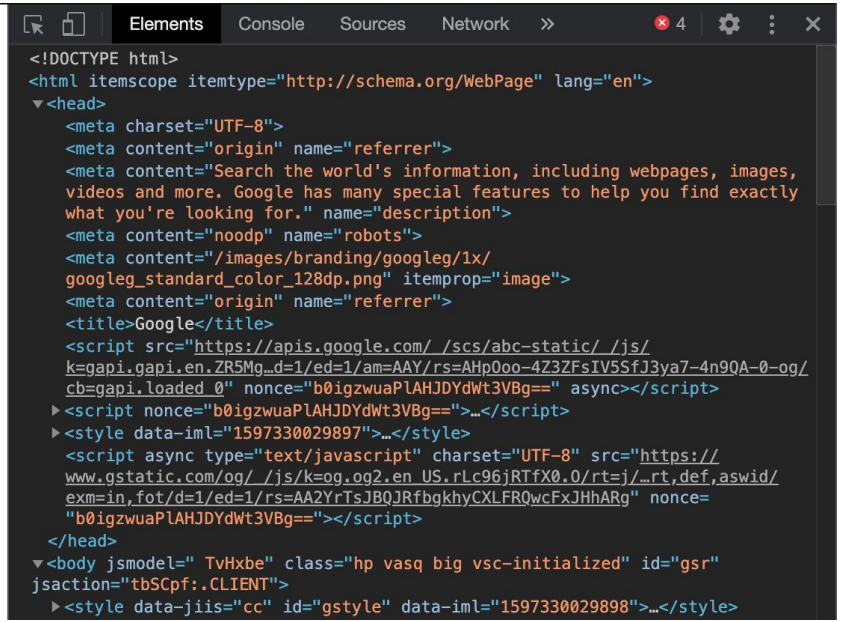
- Templates enable you to put your HTML into nicely formatted files
  - No need to stuff all HTML into a crazy string
- Flask has a built-in templating language called Jinja2
- Instead of putting HTML in your python file, just tell Flask to render a template
- Docs here: https://jinja.palletsprojects.com/en/3.0.x/templates/

It’s demo time! 

HTML via Flask

HTML can get complex..





This doesn’t seem efficient...

```
import flask
import random
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return '<html itemscope="" itemtype="http://schema.org/WebPage" lang="en"><head><meta content="Search the world's information, including webpages, images, videos and more. Google has many special features to help you find exactly what you're looking for." name="description"><meta content="noodp" name="robots"><meta content="/logos/doodles/2017/bessie-colemans-125th-birthday-5751652702224384-hp.gif" itemprop="image"><link href="/images/branding/product/ico/googleg_lodp.ico" rel="shortcut icon"><meta content="Bessie Coleman’s 125th birthday! #GoogleDoodle" property="og:description">...'

app.run()
```

The power of micro-frameworks

- Remember how we discussed that Flask is a micro-framework? As in, it abstracts over very little (AKA gives the user control over what to abstract)?
- Jinja2 is a great example of this abstraction.
- Flask allows a user to determine how they want to render HTML, whether it be manually or using a template engine such as Jinja2, Mako, etc.

With that, let’s get started and make our app with templates!

App.py can stay simple...

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    debug=True
)
```

```html
<!-- templates/index.html -->
<h1>Hello, templates!</h1>
```

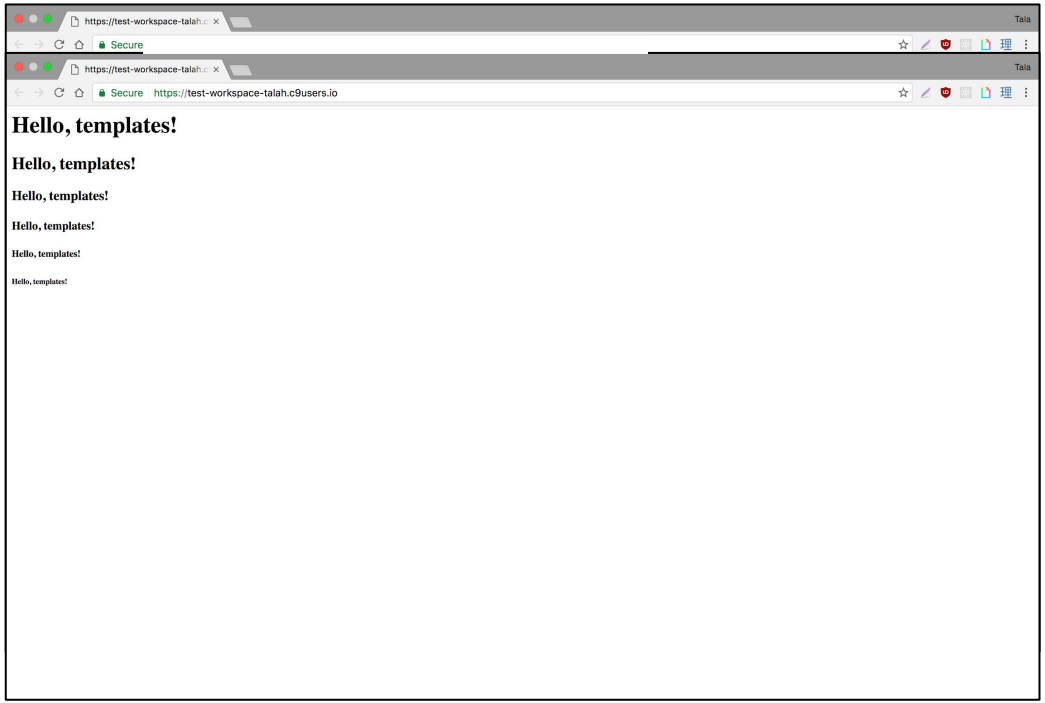
This is the important difference!

Index.html can slowly grow...

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    debug=True
)
```

```html
<!-- templates/index.html -->
<h1>Hello, templates!</h1>
<h2>Hello, templates!</h2>
<h3>Hello, templates!</h3>
<h4>Hello, templates!</h4>
<h5>Hello, templates!</h5>
<h6>Hello, templates!</h6>
```

As the web page grows!

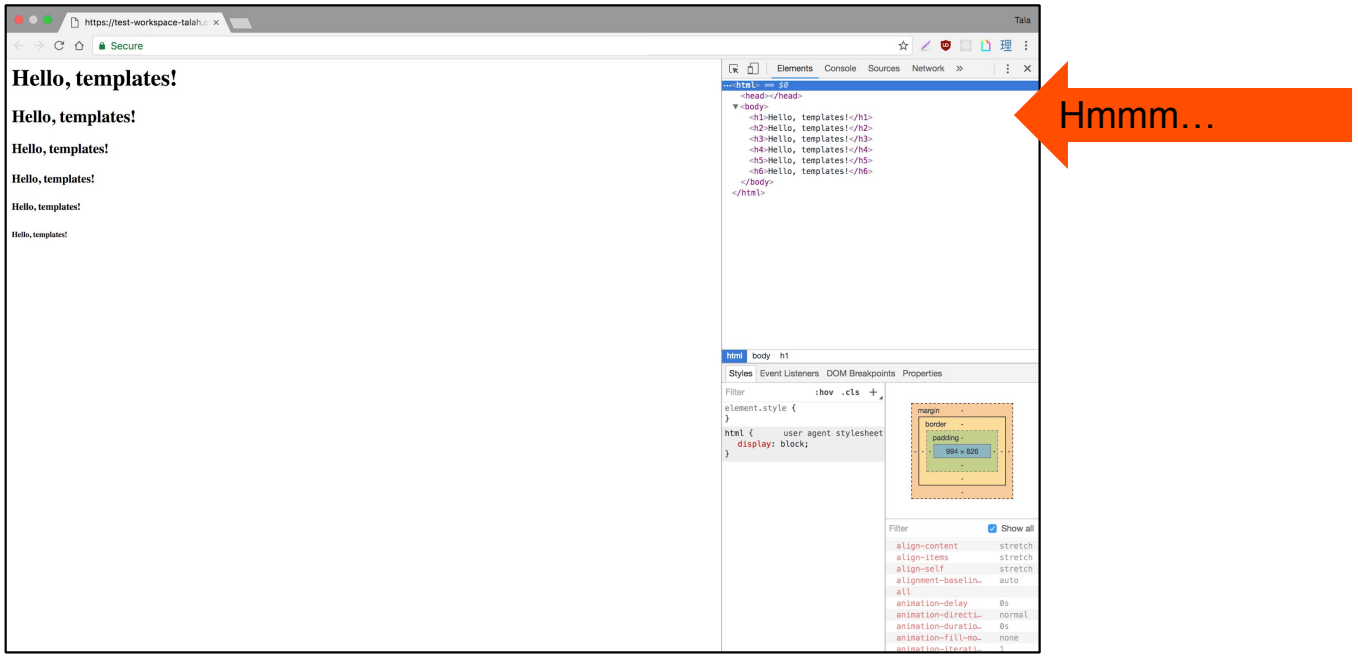


“Bad” HTML

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    debug=True
)
```

```html
<!-- templates/index.html -->
<h1>Hello, templates!</h1>
<h2>Hello, templates!</h2>
<h3>Hello, templates!</h3>
<h4>Hello, templates!</h4>
<h5>Hello, templates!</h5>
<h6>Hello, templates!</h6>
```

Browser is lax with syntax



But we should still do better in our HTML

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    debug=True
)
```

```html
<!-- templates/index.html -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, templates!</h1>
    <h2>Hello, templates!</h2>
    <h3>Hello, templates!</h3>
    <h4>Hello, templates!</h4>
    <h5>Hello, templates!</h5>
    <h6>Hello, templates!</h6>
  </body>
</html>
```

How do we get a Python variable to show up in our HTML file?

Passing data down… the hard way.

```
# rand.py
import flask
import random
import os

app = flask.Flask(__name__)

@app.route('/random')
def index():
    r = random.randint(1, 20)
    return '<h1>' + str(r) + '</h1>'

app.run(
    debug=True
)
```

Passing data down… the easy way!

```python
# app.py
import flask, random, os

app = flask.Flask(__name__)

@app.route('/')  # we'll use the default page
def index():
    num = random.randint(1, 20)
    return flask.render_template(
        "index.html",
        random_number=num # if this is confusing, look up 'python kwargs'
    )

app.run(
    debug=True
}
```

```html
<!-- templates/index.html -->
<html>
  <head>
  </head>
  <body>
    <h1>{{ random_number }} </h1>
  </body>
</html>
```

Passing **more** data down… the easy way!

```python
# app.py
import flask, random, os

app = flask.Flask(__name__)

@app.route('/')  # we'll use the default page
def index():
    num_one=random.randint(1, 20)
    num_two=random.randint(1, 20)
    return flask.render_template(
        "index.html",
        random_num_one=num_one,
        random_num_two=num_two
    )

app.run(
    debug=True
)
```

```html
<!-- templates/index.html -->
<html>
  <head>
  </head>
  <body>
    <h1>{{ random_num_one }} and {{ random_num_two }} </h1>
  </body>
</html>
```

Static vs. Dynamic web pages

Static web pages are provided by the server as originally stored. No additional processing is done.

- For us, this happens when your view function is just render_template without any additional parameters.
- “hello, templates!” page was a static web page

Dynamic web pages are provided by the server after processing is done to create a response.

- This can include server fetching data from a database, talking to other services, or even generating random numbers
- Our random number page was a dynamic web page

Another way of thinking about the difference: **is the HTML the same every time?**

CFU: Static vs Dynamic

Which of the following are static or dynamic webpages?

1. A page that displays the current Unix time (seconds since January 1, 1970)
2. A page that shows an HTML file containing five hard-coded book titles
3. A page that displays a random number, and nothing else (obviously a bad website idea but whatever)

CFU: Static vs Dynamic

Which of the following are static or dynamic webpages?

1. A page that displays the current Unix time (seconds since January 1, 1970)
2. A page that shows an HTML file containing five hard-coded book titles
3. A page that displays a random number, and nothing else (obviously a bad website idea but whatever)

1. Dynamic
2. Static
3. Dynamic

Static Resources

Different types of data that are provided by the server as they're stored.

For example - images, CSS files

Static images

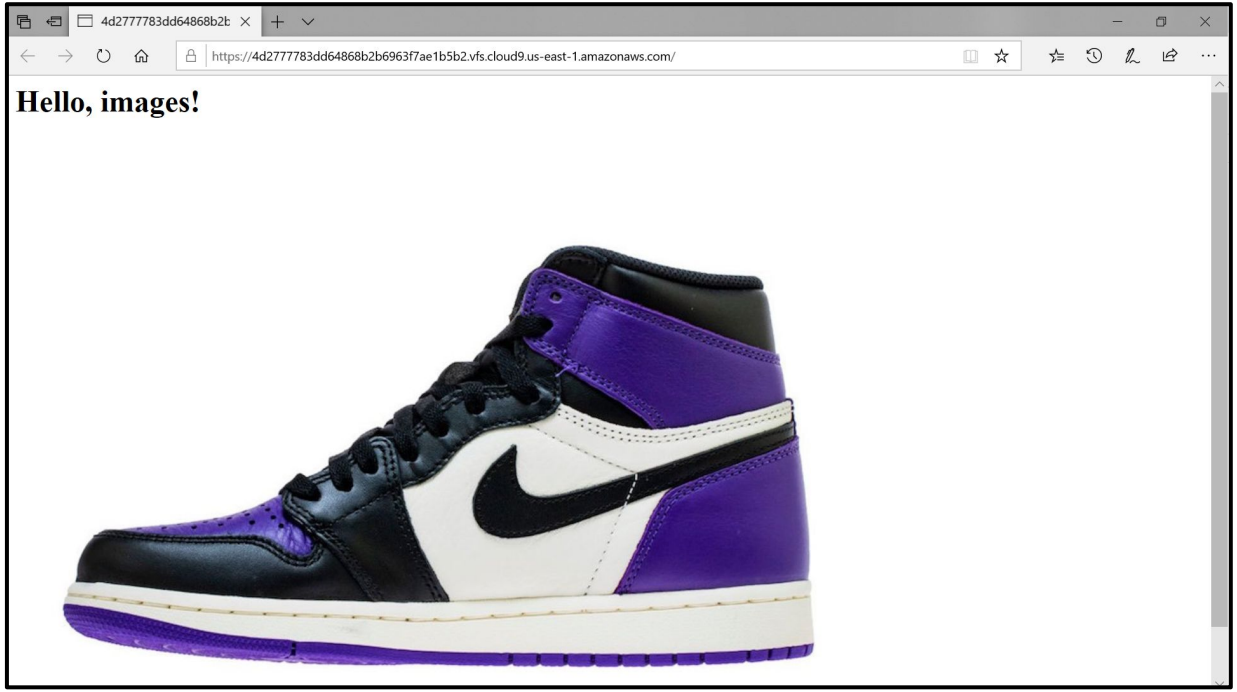
```
VM:~/lect5/static $ wget https://www.kicksonfire.com/wp-content/uploads/2018/09/AIR-JORDAN-1-3-1.jpg #
downloads a picture from a URL
--2019-08-25 21:49:49--  https://www.kicksonfire.com/wp-content/uploads/2018/09/AIR-JORDAN-1-3-1.jpg
Resolving www.kicksonfire.com (www.kicksonfire.com)... 151.139.244.25
Connecting to www.kicksonfire.com (www.kicksonfire.com)|151.139.244.25|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 62965 (61K) [image/jpeg]
Saving to: ‘AIR-JORDAN-1-3-1.jpg’

AIR-JORDAN-1-3-1.jpg
100%[=====================================================================>]  61.49K
--.-KB/s    in 0s

2019-08-25 21:49:52 (307 MB/s) - ‘AIR-JORDAN-1-3-1.jpg’ saved [62965/62965]

vocstartsoft:~/lect5/static $ ls
AIR-JORDAN-1-3-1.jpg
```

Static images

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    debug=True
)
```

```html
<!-- templates/index.html -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, images!</h1>
    <img src="/static/AIR-JORDAN-1-3-1.jpg" />
  </body>
</html>
```

Voila!


4d2777783dd64868b2b
https://4d2777783dd64868b2b6963f7ae1b5b2.vfs.cloud9.us-east-1.amazonaws.com/
Hello, images!

This equals That

```
<!-- templates/index.html -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, images!</h1>
    <img src= "https://www.kicksonfire.com/
wp-content/uploads/2018/09/AIR-JORDAN-1-3-1.jpg"
/>
  </body>
</html>
```

```
<!-- templates/index.html -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, images!</h1>
    <img src="/static/AIR-JORDAN-1-3-1.jpg" />
  </body>
</html>
```

HTML: Passing the Torch 🔥

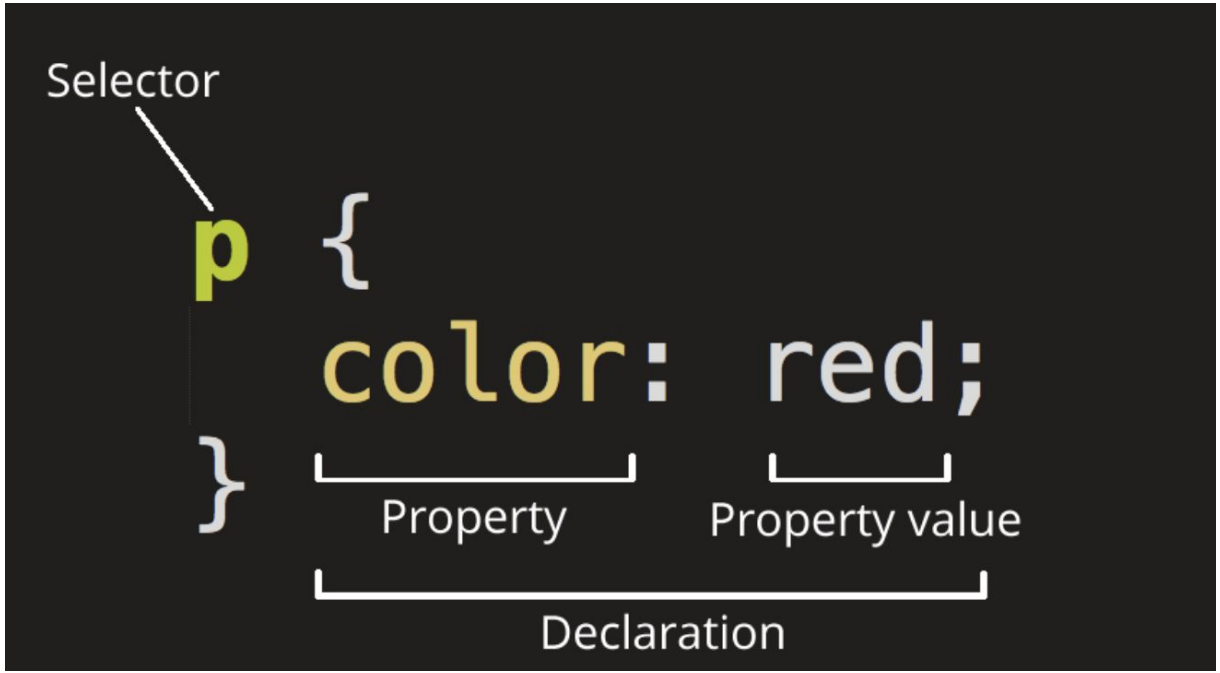
- I'm not going to teach you how to use HTML, so you're going to have to do research on your own.
- But HTML is pretty simple to pick up, here are some resources:
  - **https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics**
  - https://developer.mozilla.org/en-US/docs/Learn/HTML
- Post on Discord or come to Office Hours if you need help!

CSS

Cascading **S**tyle **S**heets

- Describe how the content on a web page (as described by HTML) should look.
- It can include location, positioning, color, and opacity.

Anatomy of a CSS rule set 🦴



Special selectors 🤩

Custom class selector

```html
<ul>
  <li>Item one</li>
  <li class="special">Item two</li>
  <li>Item <em>three</em></li>
</ul>
```

```css
.special {
  color: orange;
  font-weight: bold;
}
```

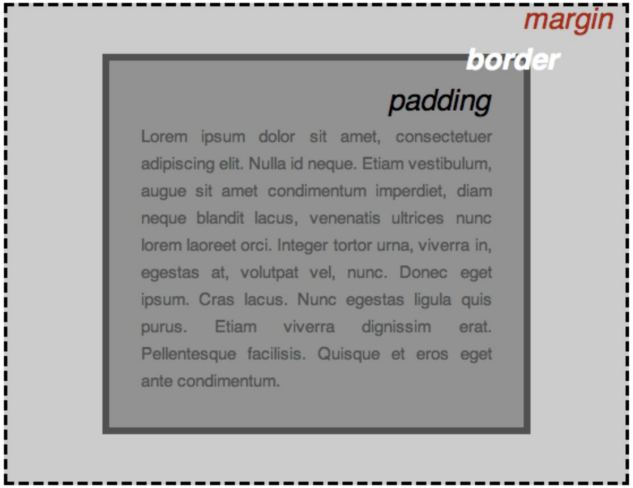
- Item one
- **Item two**
- Item *three*

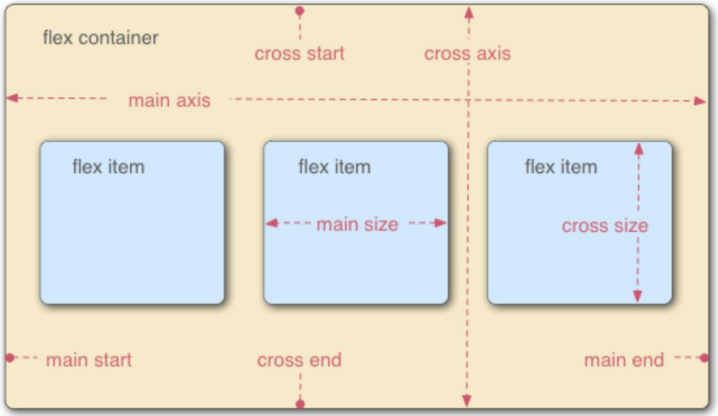
State selectors

```css
a:hover {
  text-decoration: none;
}
```

Lot’s of useful CSS properties...

- Background-color
- Color
- Font-size
- Font
- Text-align
- Border
- Margin
- Padding
- Display
- Height
- Width





It’s demo time! 

CSS

https://replit.com/@jomart-gsu/VengefulWelloffSlash#index.html

Styling in an HTML file

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0'),
    debug=True
)
```

```html
<!-- templates/index.html -->
<html>
  <head>
    <style>
          body {
            font-family: Helvetica;
        color: white;
        background-color: black;
        font-size: 96pt;
        text-align: right;
        padding-top: 1em;
      }
    </style>
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

Styling in a style sheet (file)

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0'),
    debug=True
)
```

Create a static folder

```html
<!-- templates/index.html -->
<html>
  <head>
    <link rel="stylesheet" href={{ url_for('static', filename='style.css')
}} />
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

```css
/* static/style.css */
body {
    font-family: Helvetica;
    color: white;
    background-color: black;
    font-size: 96pt;
    text-align: right;
    padding-top: 1em;
}
```

Creating our Static Folder

```
vocstartsoft:~/environment/ $ mkdir lect4-1

vocstartsoft:~/environment/ $ cd lect4-1

vocstartsoft:~/environment/lect4-1 $ touch app.py

vocstartsoft:~/environment/lect4-1 $ mkdir static

vocstartsoft:~/environment/lect4 $ ls

app.py  static/

vocstartsoft:~/environment/lect4 $ cd static

vocstartsoft:~/environment/lect4/static $ touch theme.css

vocstartsoft:~/environment/lect4/static $ ls

theme.css

vocstartsoft:~/environment/lect4/static $ up

vocstartsoft:~/environment/lect4/ $
```

Styling in a style sheet (file)

```python
# app.py
import flask
import os

app = flask.Flask(__name__)

@app.route('/')
def index():
    return flask.render_template("index.html")

app.run(
    port=int(os.getenv('PORT', 8080)),
    host=os.getenv('IP', '0.0.0.0'),
    debug=True
)
```

```html
<!-- templates/index.html -->
<html>
  <head>
    <link rel="stylesheet" href="/static/theme.css" />
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

```css
/* static/theme.css */
body {
    font-family: Helvetica;
    color: white;
    background-color: black;
    font-size: 96pt;
    text-align: right;
    padding-top: 1em;
}
```

CSS: Passing the Torch 🔥

- Most web sites put their stylesheets into separate .css files
- **Always** move to a style sheet instead of in-line
- CSS3 has stuff like animations, filters like blur, and rounded corners
- Resources for CSS (no more in-class learning):
  - **https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics**
  - https://developer.mozilla.org/en-US/docs/Learn/CSS
  - https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Introduction
  - http://learnlayout.com/

CSS Challenge

- Let’s practice!
- https://replit.com/@replit/HTML-CSS-JS?v=1#index.html (or Replit -> Templates -> HTML, CSS, JS)
  - Make whatever kind of webpage you want - it can be a list of
  - Play around with any CSS attributes you’re curious about. See what happens when you tweak different properties in the reference docs, and see what questions you have
  - Have a final product ready to go at the end of class so we can demo.

Bye 👋

- Before next class, do HW6.
  - This will actually be the beginning of a sequence of homeworks that add up to a project!
- **Game Jam** via Programming Club is **today, 3-6pm, in Student Center East 217**!
- Unrelated, but the **voter registration deadline is October 11th**!