CSE 503

Introduction to Computer Science for Non-Majors

Dr. Eric Mikida
epmikida@buffalo.edu
208 Capen Hall

Day 22
Web Servers (Part 1)

Announcements

- Lab #3 due Monday @ Midnight

A Fun Little Trick

Any guess as to what the following does?

```
for (let elem of document.all) {
    elem.style.fontFamily = "courier";
}
```

A Fun Little Trick

Any guess as to what the following does?

```
for (let elem of document.all) {
    elem.style.fontFamily = "courier";
}
```

Let's test it on a real website...Since most websites we look at use HTML and JavaScript, we can also run our own JavaScript on those websites by using our browsers Developer Console (just like the console in replit).

https://www.w3schools.com/cssref/css_websafe_fonts.asp

What is a Web Server?

How do you get food at a nice restaurant?

What is a Web Server?

How do you get food at a nice restaurant?

Do you go back into the kitchen and get it?

What is a Web Server?

How do you get food at a nice restaurant?

Do you go back into the kitchen and get it? ***NO***

What is a Web Server?

How do you get food at a nice restaurant?

Do you go back into the kitchen and get it? ***NO***

You interact with the menu and waiter/waitress (front-end)

What is a Web Server?

How do you get food at a nice restaurant?

Do you go back into the kitchen and get it? ***NO***

You interact with the menu and waiter/waitress (front-end)

Does the waiter/waitress then go back and make your food?

What is a Web Server?

How do you get food at a nice restaurant?

Do you go back into the kitchen and get it? ***NO***

You interact with the menu and waiter/waitress (front-end)

Does the waiter/waitress then go back and make your food? ***NO***

What is a Web Server?

How do you get food at a nice restaurant?

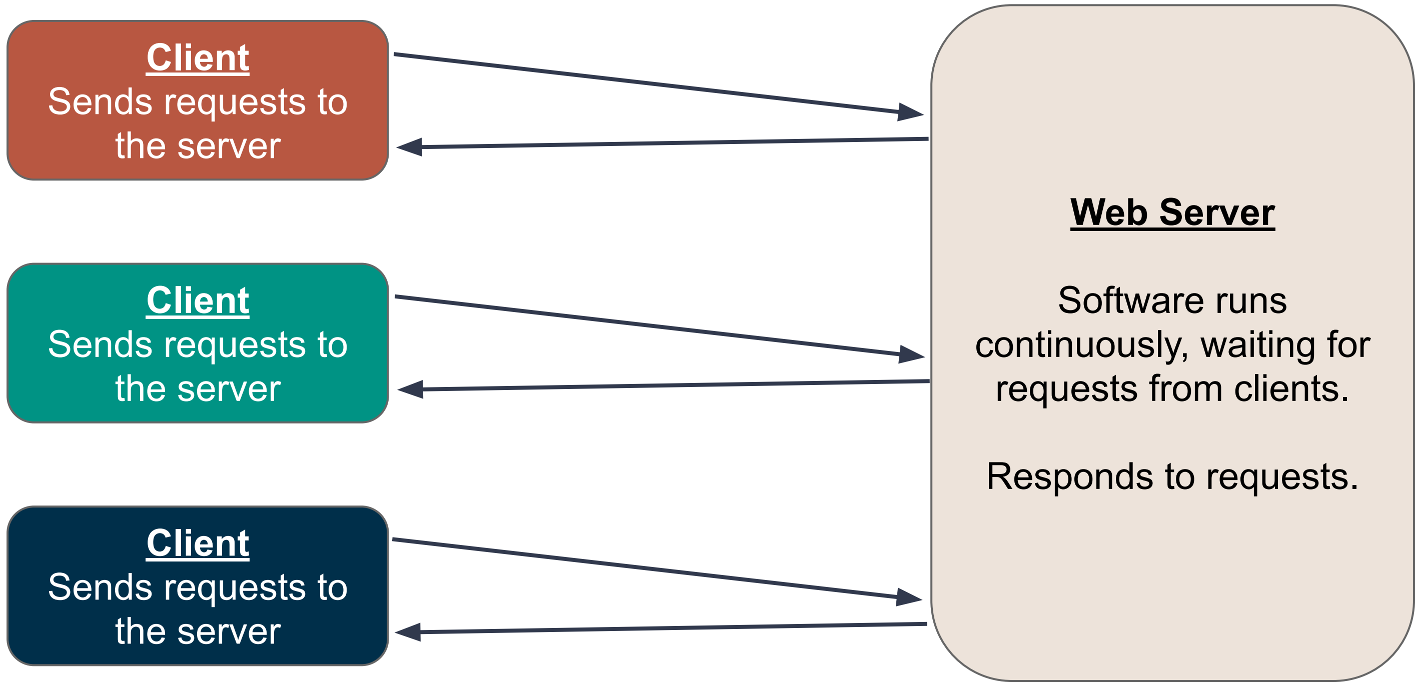
Do you go back into the kitchen and get it? ***NO***

You interact with the menu and waiter/waitress (front-end)

Does the waiter/waitress then go back and make your food? ***NO***

They put in a request for the kitchen (the back-end) to make your food

Web Server


Client
Sends requests to the server
Client
Sends requests to the server
Client
Sends requests to the server
Web Server
Software runs continuously, waiting for requests from clients.
Responds to requests.

How Does This Relate to Us?

- **Python** is great for reading, processing, and manipulating data
- **JavaScript** is great for building a front-end, and displaying results
- Our project consists of two main parts:
    1. A **Python Web Server** back-end which reads, processes and manipulates our data efficiently
    2. A **JavaScript** front-end which communicates with the web server, displays our results, and allows for users to interact with our data

How Does This Relate to Us?

- **Python** is great for reading, processing, and manipulating data
- **JavaScript** is great for building a front-end, and displaying results
- Our project consists of two main parts:
  1. A **Python Web Server** back-end which reads, processes and manipulates our data efficiently
  2. A **JavaScript** front-end which communicates with the web server, displays our results, and allows for users to interact with our data

The next few lectures will be about how to setup a web server in Python, and how to facilitate communication between the Python and JavaScript parts of our project.

HTTP

- Communication with the web server happens via **HTTP requests**
- The server sends back an **HTTP response**

HTTP

- Communication with the web server happens via **HTTP requests**
- The server sends back an **HTTP response**
- Requests follow the following format:

```
<protocol>://<server>/<path>?<query string>
```

HTTP

- Communication with the web server happens via **HTTP requests**
- The server sends back an **HTTP response**
- Requests follow the following format:

```
<protocol>://<server>/<path>?<query string>
```

Protocol: HTTP or HTTPS

HTTP

- Communication with the web server happens via **HTTP requests**
- The server sends back an **HTTP response**
- Requests follow the following format:

```
<protocol>://<server>/<path>?<query string>
```

Protocol: HTTP or HTTPS

Server: The domain name for the server, ie **www.buffalo.edu**

HTTP

- Communication with the web server happens via **HTTP requests**
- The server sends back an **HTTP response**
- Requests follow the following format:

```
<protocol>://<server>/<path>?<query string>
```

Protocol: HTTP or HTTPS

Server: The domain name for the server, ie www.buffalo.edu

Path: Name for the resource being requested

HTTP

- Communication with the web server happens via **HTTP requests**
- The server sends back an **HTTP response**
- Requests follow the following format:

```
<protocol>://<server>/<path>?<query string>
```

Protocol: HTTP or HTTPS

Server: The domain name for the server, ie **www.buffalo.edu**

Path: Name for the resource being requested

Query String: Provide additional info with key-value pairs (not always used)

HTTP

- Communication with the web server happens via **HTTP requests**
- The server sends back an **HTTP response**
- Requests follow the following format:

```
<protocol>://<server>/<path>?<query string>
```

Protocol: HTTP or HTTPS

Server: The domain name for the server, ie www.buffalo.edu

Path: Name for the resource being requested

Query String: Provide additional info with key-value pairs (not always used)

Example: `https://engineering.buffalo.edu/computer-science-engineering.html`

Making an HTTP request in Python

```python
import urllib.request

protocol = "https"
server = "engineering.buffalo.edu"
path = "computer-science-engineering.html"
url = protocol + "://" + server + "/" + path

response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)
```

Making an HTTP request in Python

```
import urllib.request

protocol = "https"
server = "engineering.buffalo.edu"
path = "computer-science-engineering.html"
url = protocol + "://" + server + "/" + path

response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)
```

Import a library for making HTTP requests

Making an HTTP request in Python

```
import urllib.request

protocol = "https"
server = "engineering.buffalo.edu"
path = "computer-science-engineering.html"
url = protocol + "://" + server + "/" + path

response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)
```

Create our request
(could be done in one line)

Making an HTTP request in Python

```
import urllib.request

protocol = "https"
server = "engineering.buffalo.edu"
path = "computer-science-engineering.html"
url = protocol + "://" + server + "/" + path

response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)
```

Make the request, and store the response

Making an HTTP request in Python

```
import urllib.request

protocol = "https"
server = "engineering.buffalo.edu"
path = "computer-science-engineering.html"
url = protocol + "://" + server + "/" + path

response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)
```

Read and decode the response
(initially it's just 0s and 1s)

Making an HTTP request in Python

```
import urllib.request

protocol = "https"
server = "engineering.buffalo.edu"
path = "computer-science-engineering.html"
url = protocol + "://" + server + "/" + path

response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)
```

Do whatever we want with the response! (It's just a string)

Query Strings

Query strings are a set of key-value pairs to pass extra information in your request

Keys and values are separated by "=", multiple key-value pairs separated by "&"

Query Strings

Query strings are a set of key-value pairs to pass extra information in your request

Keys and values are separated by "=", multiple key-value pairs separated by "&"

Examples

https://www.youtube.com/watch?v=wL9E2QKP2us

- Query String: "v=wL9E2QKP2us"
- key "v" with value "wL9E2QKP2us"

https://duckduckgo.com/?q=internships&t=h_&ia=web

- key "q" with value "internships"
- key "t" with value "h_"
- key "ia" with value "web"

Brief Aside on Web Scraping

The Internet, as most people know it, is designed for human consumption

What if we want to write software that reads data from the Internet?

Brief Aside on Web Scraping

The Internet, as most people know it, is designed for human consumption

What if we want to write software that reads data from the Internet?

- A web scraper is software that ***reads data from HTML***
- Many libraries exist to make this easier
- We won't explore this in CSE503, though it can be a fun area to explore on your own

Web APIs - An Alternative to Scraping

The Internet, as most people know it, is designed for human consumption

What if we want to write software that reads data from the Internet?

Web APIs - An Alternative to Scraping

The Internet, as most people know it, is designed for human consumption

What if we want to write software that reads data from the Internet?

- Web APIs are hosted by web servers at urls, but instead of sending HTML they send raw data
- Designed for programmatic consumption
- Typically send data as JSON

Web APIs - ISS Example

```
import urllib.request

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)
```

Web APIs - ISS Example

```
import urllib.request

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)
```

Make a request to the open-notify API

Documentation: http://open-notify.org/Open-Notify-API/ISS-Location-Now/

Web APIs - ISS Example

```python
import urllib.request

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content = response.read().decode()
print(content)
```

Returns a JSON string... →

```
{
  "message": "success",
  "iss_position": { "longitude": "135.1326",
                    "latitude": "19.9913"},
  "timestamp": 1666207804
}
```

JSON

So…What's a JSON string? How can we use it? Does it look familiar?

What's a JSON String?

JSON (JavaScript Object Notation) is a data format that can be represented as strings

- We can send these strings to communicate data across the Internet
- All programming languages can read strings
- Doesn't matter what language was used for client or server program
- They can all "speak" JSON since its just strings
- More flexible than CSV

JSON

6 different data types:

1. String: Any value in "double quotes"
2. Number: Any value not in quotes "true", "false", and "null" will be interpreted as a number.
3. Boolean: Either "true" or "false" without the quotes
4. Null: The word "null" without the quotes
5. Array: A comma-separated list of values surrounded by [brackets]
6. Object: A comma-separated list of key-value pairs surrounded by {braces}

Closely resembles JavaScript and Python syntax that we've seen, except it is a string.

See also: https://www.json.org or http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf.

JSON Example

```
[{"title":"God Am (Live 1996)", "artist":"Alice in
Chains","ratings":[5,4],"youtubeID":"74P4W_okEqA"},{"t
itle":"Fade to
Black","artist":"Metallica","ratings":[5,2],"youtubeID
":"WEQnzs8wl6E"}]
```

JSON in Python

```
import urllib.request
import json

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content_string = response.read().decode()

content = json.loads(content_string)
```

JSON in Python

```
import urllib.request
import json

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content_string = response.read().decode()

content = json.loads(content_string)
```

Import json library

JSON in Python

```
import urllib.request
import json

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content_string = response.read().decode()

content = json.loads(content_string)
```

Make a request just like before

JSON in Python

```
import urllib.request
import json

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content_string = response.read().decode()

content = json.loads(content_string)
```

Use the `json.loads()` function to convert the string into python type (in this case a dictionary)

JSON in Python

```
import urllib.request
import json

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content_string = response.read().decode()

content = json.loads(content_string)

????        Do whatever we want with the data!!!
```

JSON in Python

```
import urllib.request
import json

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content_string = response.read().decode()

content = json.loads(content_string)

print(content)
```

We can print it

JSON in Python

```
import urllib.request
import json

url = "http://api.open-notify.org/iss-now.json"
response = urllib.request.urlopen(url)
content_string = response.read().decode()

content = json.loads(content_string)

print(content['iss_position']['longitude'])
print(content['iss_position']['latitude'])
```

Or do anything else we may want to do