Csci 4131 Internet Programming
Fall 2021
Lecture 9
October 6th

Instructor: Dr. C

Logistics – Csci 4131 Lecture 9, October 6th

- ***Programming Homework Assignment 3 is due this coming Friday, October 8th***
- ***Homework 3 is requires a significantly more complex solution than the previous 2 homework assignments***
- ***If you haven't read the assignment requirements specification; signed up for google maps APIs; obtained your Google API key, started the google maps JavaScript API tutorial, and have some of the functionality working* you are NOW *SERIOUSLY* behind.**
- **Remaining Zybooks – small HW due Saturday 10/9**

Reading & Tutorials

Current

Google Maps / JavaScript API:
https://developers.google.com/maps/documentation/javascript/tutorial

Google Maps Geocoding
https://developers.google.com/maps/documentation/javascript/geocoding,

Google Maps Places API
https://developers.google.com/maps/documentation/javascript/places

Google Directions Service
https://developers.google.com/maps/documentation/javascript/directions

Google Click on Points of Interest (used to fill location field on Form when points of interest are selected/clicked on the map next to it):
https://developers.google.com/maps/documentation/javascript/examples/event-poi

Optional:

Sebesta - Chapter 5,6; and JavaScript tutorials (see course schedule in the module at the top of the home page on the Class Canvas site)

Homework 3- sign up for Google Maps

https://developers.google.com/maps/documentation/javascript/get-api-key

You must enable billing and give google a credit or debit card number

You get a 200 dollar credit for their services

You should use, at most very little of the credit for this assignment or follow-up assignments (20 dollars or less)

Email the class help email (csci4131help-s21@umn.edu) immediately if you have an issue with signing up, or if you somehow manage to incur charges to your account for work you do in this course.

Make sure to sign up for all the google API's and review Google's documentation and Examples on the following Services / APIs:

- Google Maps
- The Geocoding Library – for markers
- The Places Service – for searching for places
- The Directions Service
  - The Directions Display Object
  - The Directions Renderer Object

- Note, w3schools has tutorials to get you started as well
  - https://www.w3schools.com/graphics/google_maps_intro.asp

Starting Today

Necessary Preparations for enabling your understanding of HTTP: Getting Setup for Python 3.x – which you will use to do HW4

- https://www.python.org – to download python to your machine so you can develop and run HW Assignment 4 – note, our target machines are ubuntu, and our examples have been developed and tested on the cse labs machines. That is where we will be testing **and GRADING** your programs.

- https://docs.python.org/3/ - documentation on newest version of Python
- https://docs.python.org/3/howto/sockets.html
- https://docs.python.org/3/howto/sockets.html

- - documentation on the socket library – which will help you understand EchoClient and EchoServer python programs (which I'll post on Canvas for your review and refactoring for HW Assignment 4)

Learning or Refreshing your Python

- https://docs.python.org/3/tutorial/
- https://www.learnpython.org/
- https://www.linkedin.com/learning/topics/python?u=42740356
- (need your x500 and umn password to login to linked in learning – lots of Python Tutorials there)

Upcoming: Node.js

Introduction to Node.js (Building a Webserver in JavaScript)

https://www.w3schools.com/nodejs/

https://codeburst.io/the-only-nodejs-introduction-youll-ever-need-d969a47ef219

Questions?

JavaScript Events Revisited and Wrapped up

(Chapters 5,6 Sebesta)

- DOM events
  - Enable scripts to respond to user interactions and modify the page accordingly

- Events and event handling (via JavaScript)
  - make web applications more dynamic and interactive

ref: http://www.quirksmode.org/js/introevents.html

Summary: Event Handling

- An **event handler** is a function that responds to an event.
- Assigning an event handler to an event on a DOM node is called **registering an event handler**
- **addEventListener** method can be called multiple times on a DOM node to register more than one event-handling method for different events associated with that node.
- Remove an event listener by calling **removeEventListener** with the same arguments that you passed to **addEventListener** to register the event handler.

Caveats

- The load event enables access to the elements in an HTML5 page AFTER they are loaded
- Statements outside of functions in a script (JavaScript) loaded in a document's head section execute when the script loads (***that is, before all the elements in the body are loaded***)
- For example, if JavaScript executed by a call in the head section to **getElementById** for an HTML element in the body, then **getElementById** will return null

- **That is a race condition!**

Review Exercise 1, Lecture 8

- So, is there a potential race condition with this revised version of myClock?
  - myclockRace.html
- If so what, and how would you fix it???

Solution(s)?

- Make sure you don't have statements that reference DOM elements outside of script functions that you load in the header, and then associate the functions with the objects and events by writing a function that is called when the window onload event.
- Or???
- Place the script(s) as the last item(s) in the document's body
- Are there other solutions?
- Of course, but we won't cover them in detail here

Lecture 9 Exercise 1:

- **What is displayed when the following web page is loaded into a browser? (Download the file racecondition.html from the lecture 9 materials in the week 5 module on Canvas)**

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Race Condition</title>
        <script language=javascript>
                var textobj = document.getElementById("stuff");
                textobj.value = "Hello World";
        </script>
</head>
<body>
        <input type="text" id="stuff">
</body>
</html>
```

Exercise 1, Continued:

- Refactor (fix/rewrite) the page so the text:

  Hello World

  is displayed in the text box when the browser loads the page

Submit your answer via the Lecture 9, Exercise 1 Submission Item in the week 6 module on Canvas

Close your computer or raise your hand when you are done

Think/ Pair/Share : What is displayed when the web page below is loaded by a browser?

Try to figure out the answer without running it, raise your hand when you are done!

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Closure Example</title>
        <script language=javascript>

                function mult(x) {
                        return function (y) {
                                return x * y;
                        }
                }

                var m3 = mult(3);
                alert(m3(4));
        </script>
</head>
<body>
</body>
</html>
```

Why???

Here is what happens:

1. When the **mult** function is called, it returns an *anonymous* function:

```
function (y) {
        return x * y;
}
```

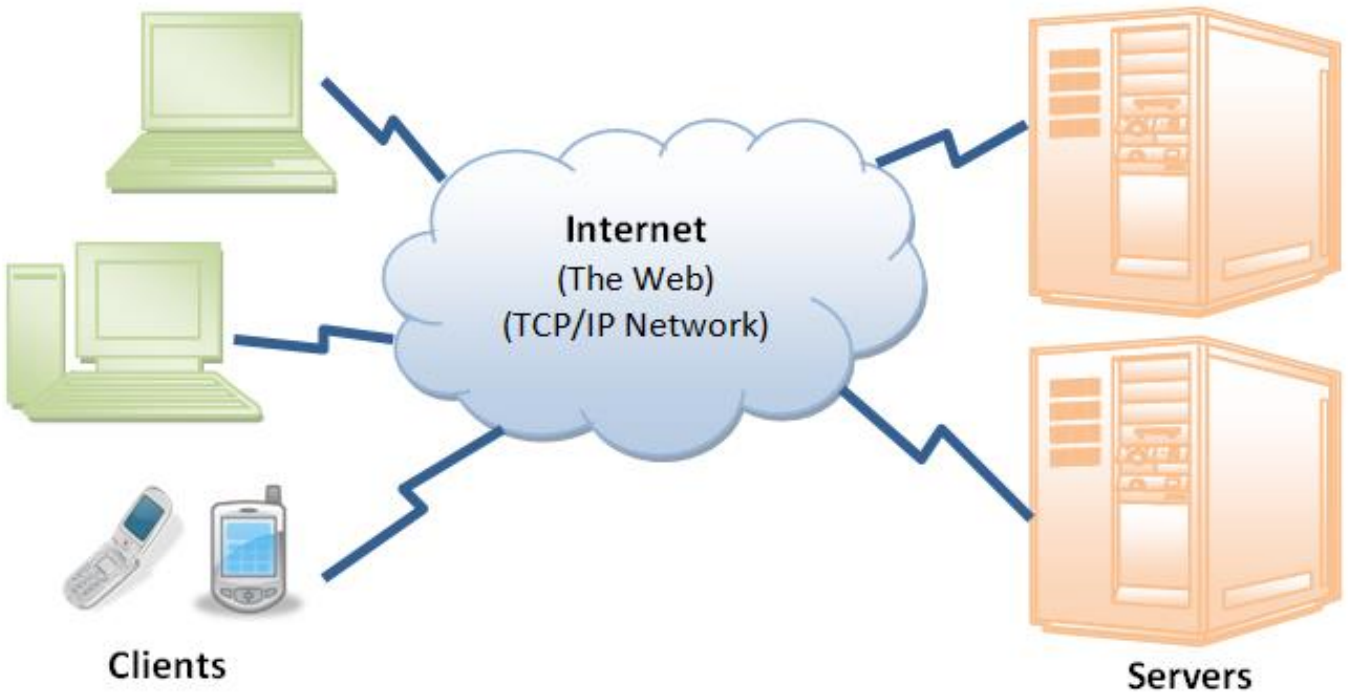
2. The *anonymous* function "closes" the context and remembers what the value of the parameter **x** was when the **mult** function was called (i.e., it **binds x** to the value **3**)
3. The *anonymous* function returned by the **mult** function is bound to the identifier **m3**). Note, the *anonymous* function retains the value **x** bound to **3** – the *anonymous* function will always retain the binding **x = 3** that was created when the **anonymous** function was created by **mult**.
4. After the assignment: var m3 = mult(3); the identifier will refer to a function that multiplies the value of the parameter y times 3 and returns the result
5. So when m3 is called with a value of 4 (m3(4)), it will multiply 4 * 3 and return the result: **12**
6. So **12** is displayed an alert box.

- With Homework assignment 3, we turn our focus from just the client (Browser), over to the server side:
- **First Web Services** - Google API's for maps, which we have covered and you are wrapping up
- **Next Up**: HTTP Protocol, Server Side Scripting (an small Python Web Server)

HTTP Protocol

- Foundation of the WWW
- Purpose – impart a deeper understanding of how it works, along with a better understanding of browser and web server function
- **Reading:**
  - **https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html**
  - RFC 2616 (HTTP 1.1)
    https://tools.ietf.org/html/rfc2616
    http://www.w3c.org/Protocols/

The Web



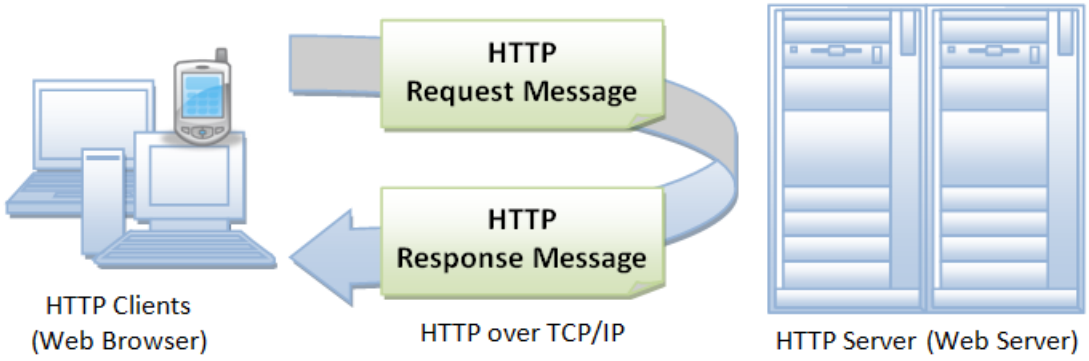
Massive distributed Client/Server Information System

HTTP – how info moves over the web

- HTTP (Hypertext Transfer Protocol) is perhaps the most popular application protocol used in the Internet (or The WEB).

- HTTP is an *asymmetric request-response client-server* protocol as illustrated.
  - An HTTP client sends a request message to an HTTP server.
  - The server, in turn, returns a response message.

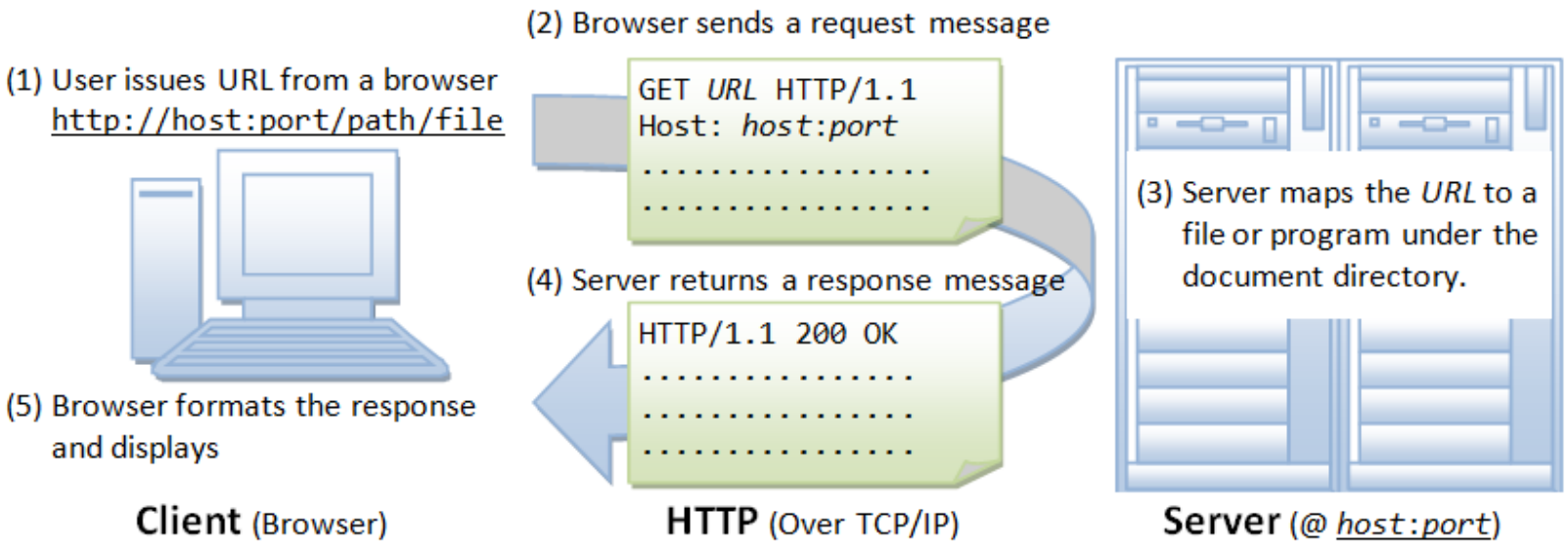
- HTTP is a *pull protocol*: the client *pulls* information from the server (instead of server *pushing* information to the client).



RFC2616: "The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems.

It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers

How the HTTP Protocol Works



Server

1) Listens for a connection
2) Sets up client connection, or closes the connection if the client is unwanted
3) Receive Request - read an HTTP request message from the network
4) Process Request – interpret the request message and take action
5) Access resource – access the resource specified in the message
6) Construct response – create the HTTP response message with the right headers
7) Send response – send response back to the client
8) Log Transaction – place notes about the completed transaction in a log file

Do you have access to a Web Server that will serve up your web pages and run web applications Here at the U-of-M?

- You Betcha!!
- https://web.archive.org/web/20201118170800/https://cseit.umn.edu/knowledge-help/homepages-cs-cselabs
- NOTE, in the writeup, http should be replaced by https everywhere http is referenced

Recall URL's

- http://www.w3.org/Addressing/URL/Overview.html

- URL's are a naming scheme for referencing resources in the Internet. See RFC 1738.
- The URL enables resources to be accessed without knowing the specifics of their underlying protocol, such as FTP or HTTP.

It is a location-dependent scheme. i.e., a URL name becomes invalid if the resource is relocated to another host or moved to a different part of the file system.

Recall URL's

- Recall, a URL has the following syntax:

  protocol://*hostname*:*port*/*path-and-file-name*

- There are 4 parts in a URL:
  - *Protocol*: The application-level protocol used by the client and server, e.g., HTTP, FTP, and telnet.
  - *Hostname*: The DNS domain name (e.g., www.test101.com) or IP address (e.g., 192.128.1.2) of the server.
  - *Port*: The TCP port number that the server is listening for incoming requests from the clients.
  - *Path-and-file-name*: The name and location of the requested resource, under the server document base directory.

URL

A URL can be in one of the two forms:

- **Absolute or Complete URL**
  - –Specifies the complete access path for the named resources in the Internet.
- **Relative or Partial URL**
  - –Meaningful only in the context of some other URL.
  - –Used when the referenced resource is on the same host machine as the referring resource.

Anybody have an example of using a Relative URL???

Absolute:

http://challouPC/c/Users/challou/Desktop/Csci4131Fall21/lectures/L9_Oct6.ppt

Absolute URL

The specification of an absolute URL contains the following information:

- Protocol to be used to access the resource: e.g., ftp, http, mailto (name others???)
- **DNS name or IP address** of the server that contains the resource.
- If needed, specification of the server's port number.
- The directory path within which the resource is contained.
- The **name of the file** representing that resource.
- Some specific named component within the resource, such as a **named "anchor" within an HTML document**.
- **Query parameters** to be passed to the resource.

URLs Continued

- URL scheme is based on DNS (Domain Name System – translates names into IP addresses). It uses DNS to identify the Internet host of the resource.
- URLs belong to a more general class of naming scheme called Uniform Resource Identifiers (URI).
- URI also defines a location independent naming scheme called Uniform Resource Names (URNs). See URI:

http://www.w3.org/hypertext/WWW/Addressing/Addressing.html

Example of a URL with a Query string in it

https://en.wikipedia.org/w/index.php?title=Query_string&action=edit

Source:
https://en.wikipedia.org/wiki/Query_string

Example

- Example of a URL with encoded query with it.
- http://www.cs.umn.edu/admissions/application.cgi?param1=value1&param2=value2
- Parameters (param 1 and param 2) to be passed to the Common Gateway Interface (CGI) program
- This (name, value) query will be passed to the CGI program as the QUERY_STRING environment variable

Examples continued

- Example of URL with inclusion of single parameter to be passed to the resource.

- http:// www.cs.umn.edu /admissions/application.php?**someValue**

- **SomeValue** - Command line parameter to be passed to the PHP script

Think/Pair/Share: What is the URL of the following file?

- The file: **index.php**
- In the domain: **cs.umn.edu**
- Accessed by the: **http protocol**

- Also, is this a complete or partial/relative URL?

Please jot your answers down in a text file, and raise hand when done

URL Port Numbers

- Recall, a URL has the following syntax:
  protocol://*hostname*:***port***/*path-and-file-name*

  BUT the Port number can be omitted if the server is running on the default port for that service.

- For example,:
  HTTP servers (port 80), (also default port for localhost)
  HTTPS (port 443),
  FTP (port 21),
  Telnet (port 23).

- Protocol can be:
  –http, telnet, ftp, mailto, others

Examples of URLs that use protocols other than HTTP / HTTPS

Telnet URL

- **telnet hostName[:port]**

- For example: to Telnet to csel- kh1262-01.cselabs.umn.edu

  % telnet csel-kh1262-01.cselabs.umn.edu

- HTML:

 Telnet to csel-kh1262-01.cselabs.umn.edu

FTP URL

ftp host[port]

For example to ftp to csel-kh1262-01.cselabs.umn.edu

```
% ftp csel-kh1262-01.cselabs.umn.edu
```

HTML:

```
<a href="ftp csel-kh1262-01.cselabs.umn.edu">
ftp to csel-kh1262-01.cselabs.umn.edu </a>
```

Next Time

- Review Exercise
- URL's revisited and wrapped up
- The HTTP Protocol in detail

Backup

Simple Google Maps Examples

- Map
- Map with Marker
- Geocoding Example

Approach to Putting Icons to Mark each Location on your Schedule on a Google MAP

- Define an location class. Add that class to each location on your events list (maybe wrap each location with an HTML element, like a paragraph or a span tag)
- Write JavaScript to get each class – document.getElementsByClassName("location");
- Loop through the list of objects
  - If the location has not already been marked on the map, using the Geocoding library
    - create a marker with the address from the list as its address, and the callback function should create marker with the custom icon and associate an info window with the Event name and address with callback event of the marker

For the rest, read the Google Maps References and do their examples / tutorials

Note

- You are free to use any code we provide in this course, as long as you do not redistribute it or place your results in directories or folders that are accessible by anyone other than you. Moreover, you should note your source for the code with a link (if at all possible) in a comment or at least a comment.
- You can reuse code from Google as well, but please credit the code with a comment with a link to the page where you obtained the code.