

Csci 4131 Internet Programming
Spring 2024
Lecture 10
February 19th
Instructor: Dr. C

Logistics – Csci 4131 Lecture 10, February 19th

- ***Programming Homework Assignment 3 is due Sunday, March 3rd***
- ***Homework 3 requires a significantly more complex solution than the previous 2 homework assignments***
- ***If you haven't read the assignment requirements specification; obtained your Google API key (see email and announcement), started the Google Maps JavaScript API tutorial, and have some of the functionality working you are NOW behind.***
- **NO ZYBOOKS HW THIS WEEK!**

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)

Make sure to 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

Upcoming After HTTP: 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 window load event enables access to the elements in an HTML5 page AFTER they are loaded
- Statements outside of a function 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, document, or body onload event.
 - https://www.w3schools.com/jsref/event_onload.asp
- Or???
- Place the script(s) as the last item(s) in the document's body
- ***Are there other solutions?***
- **Of course!**

Lecture 10 Exercise 1:

- What is displayed when the following web page is loaded into a browser? **(Download the file `racecondition.html` from the lecture 10 materials in the week 6 module on Canvas)** [racecondition.html](#)

```
<!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 10, Exercise 1 Submission Item in the week 6 module on Canvas

Close your computer and thumbs up when you are done

- With Homework assignment 3, we turn our focus from on the the client (Browser), over to the server side:
- **First Web Services** - Google API's for maps, which we are about to cover and you are working on (HW3)
- **Next Up:** HTTP Protocol, Server Side Scripting

Simple Google Maps Examples

- Map
 - Map with Marker
 - Geocoding Example
-
- The files I have posted, require a google maps API key, so you have to get one and insert it to get them to work (see your email and the announcement on Canvas for a key)

Exercise 2 – Think/Pair/Update/Submit

1. Download the google maps example:

geocoder_ex.html

And the style file

style.css

From the Lecture 10 materials in the week 6 module on Canvas

And put them in the same directory

2. Get the Google maps API key from your email or the announcement on Canvas (or use your own key)
3. Edit the file **geocode_ex.html**, add the key. Save the updated file.
4. Now open the file in your browser, and if you have added the key correctly, you should see the map!!!!
5. **Submit your updated file to the Lecture 10, Exercise 2 Submission Link found with the Lecture 10 materials in the Week 10 module**

If you work with others, put your names and x.500 ids in a comment at the top of the file before submitting

An approach to putting icons to mark each location on your Schedule on a Google MAP

1. Define an location class. Add that class to each **location** on your Schedule (maybe wrap each location with an HTML element, like a paragraph or a span tag)
2. Write JavaScript to get each class –
`document.getElementsByClassName("location");`
3. Loop through the list of objects returned in step 2
 - i. If the location has not already been marked on the map, using the Geocoding library
 - a. 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 Contact name, Contact Info, picture and address with callback event of the marker

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

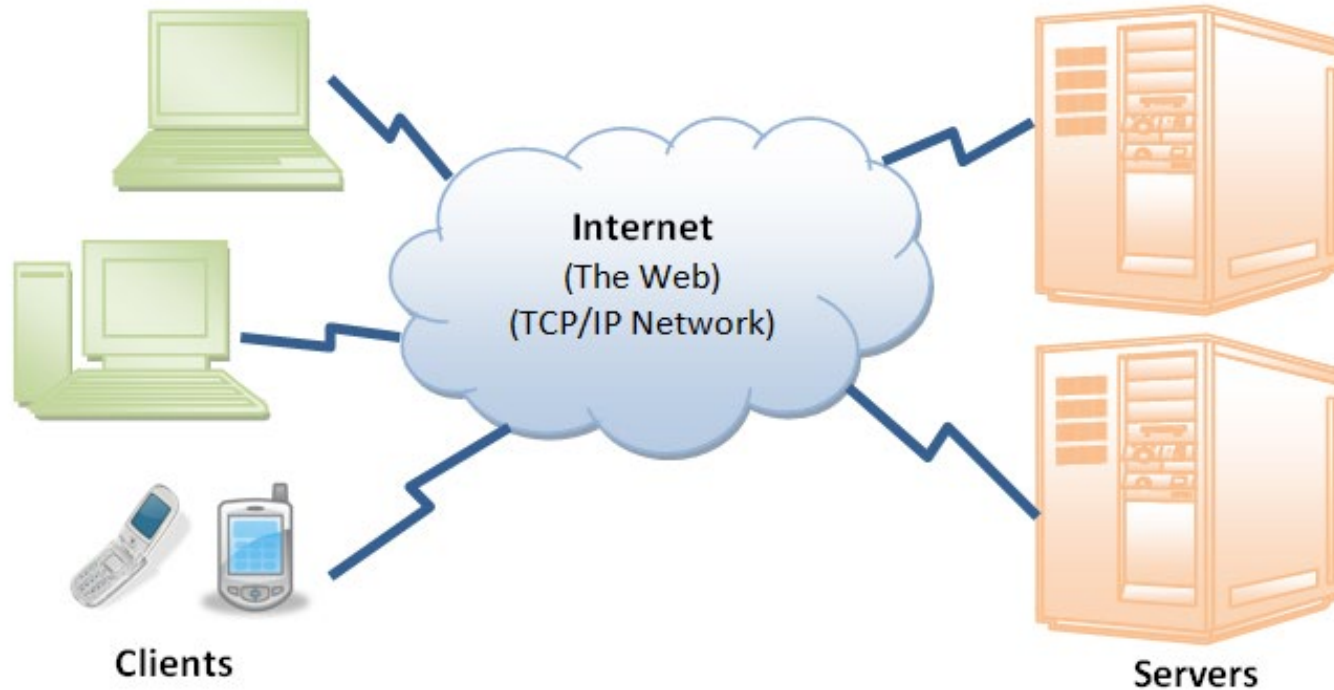
Note

- 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.

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/>

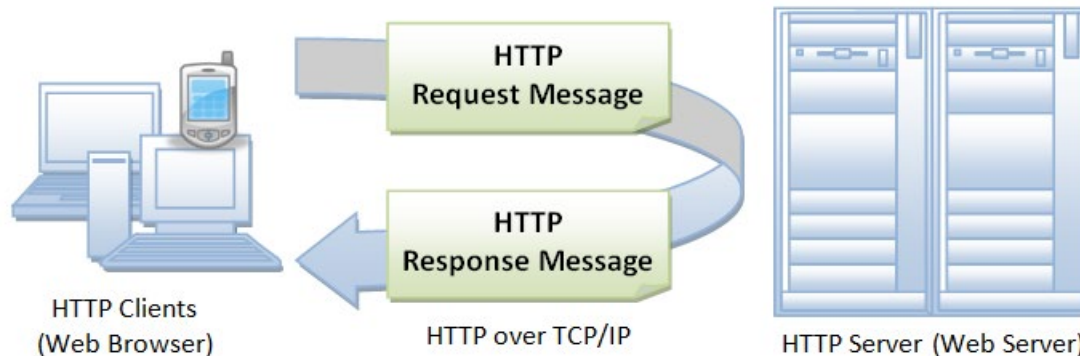
The Web



Massive distributed Client/Server Information System

HTTP – how info moves over the web

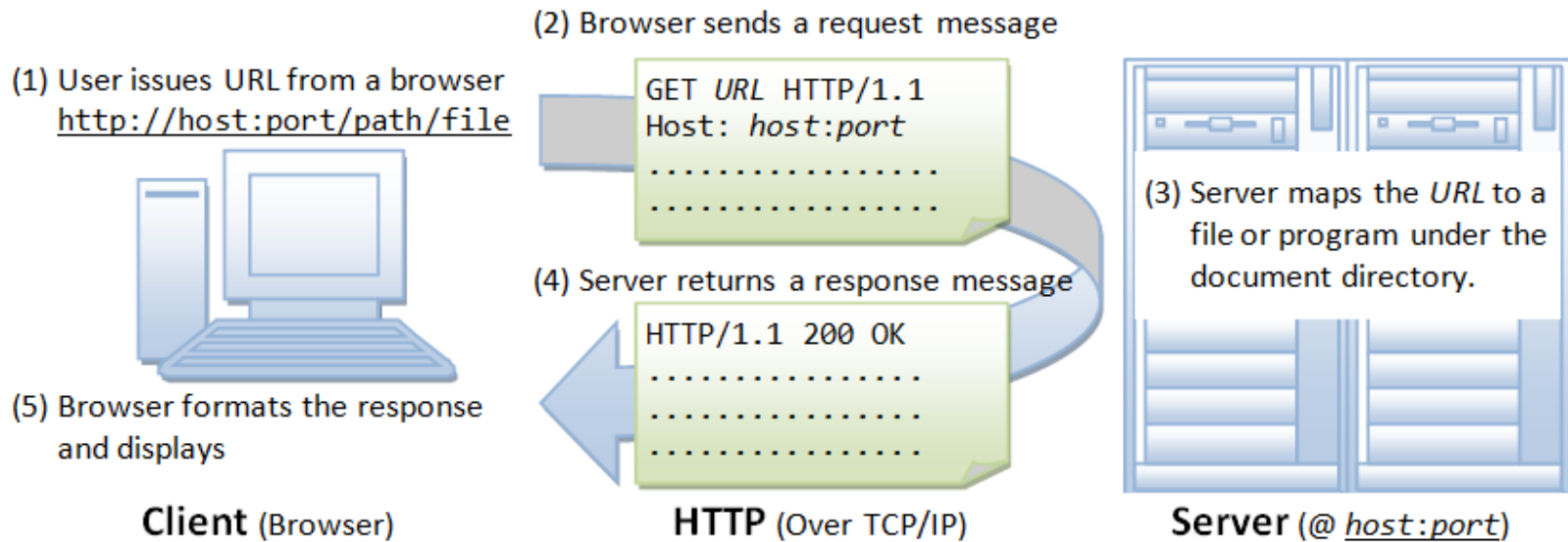
- HTTP (Hypertext Transfer Protocol) is perhaps the most popular application protocol used in the Internet (or The WEB).
- HTTP is an *asymmetric request-response client-server* protocol as illustrated.
 - An HTTP client sends a request message to an HTTP server.
 - The server, in turn, returns a response 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

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.

THINK/PAIR/SHARE

Write down an absolute URL

Write down a relative URL

Share em!!!!

Absolute URL

The specification of an absolute URL contains the following information:

- Scheme: Protocol to be used to access the resource: e.g., ftp, http, mailto (name others???)
- **hostname or domain name** of the server that contains the resource (resolves to an IP address)
- 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¶m2=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: **myContacts.html**
- 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, and thumbs up when done

Next Time

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

Next Time

- HTTP Protocol wrap-up
- Toward a simple HTTP server in Python...
- Introduction to Node.js???