Csci 4131 Internet Programming
Fall 2021
Lecture 5
September 22nd

Instructor: Dr. Dan Challou

Logistics – Csci 4131 Lecture 5, September 22nd

- HW Assignment 2 out, available in week 2 module on the Homepage of the class Canvas site (and in the assignments section). Due **THIS COMING** **Friday September 24th** at 11:59pm
- If you have not yet started on HW2 please start as soon as possible (like today!), it is a step-up in difficulty over HW1

- Zybooks assignments can be found in your zybook

- The weekly readings, tutorials, programming homework due dates and exam dates are in the item:
  - **Course Schedule: Weekly Class Readings and Tutorials, Exam Dates, and Programming Assignment Due Dates**

  in the Resources Module at the top of the Home

  Page on the class Canvas site

Remaining zyBooks assignments for this week

- zyBooks HW 3, due Saturday Sept 25th at 11:59pm

Still working the following readings and tutorials, but feel free to read ahead (google maps)

Sebesta, Ch 3 and 4, www.w3schools.com – CSS Tutorial, JavaScript Tutorial

https://www.w3schools.com/css/default.asp

http://www.w3schools.com/js/

HW 2 Updates (1)

- For embedding instgram in your Widgets page, can use the embed button to get the HTML, see the following link for instructions:

1.https://developers.facebook.com/docs/instagram/embed-button

- OR, you can use the URL format described on the link on the oEmbed page:

2.https://developers.facebook.com/docs/instagram/oembed

- The first approach is easier!
- I'll update the HW2 spec and post later today…

Homework 2 Updates (2)

- The link to the jQuery password -strength checker widget in Programming Assignment 2 does not include a progress bar!
- To get the regular, non-bonus credit, you just have to include the code (HTML, CSS, and jQuery) found at the link in the Homework 2 specification and use it to implement the password checker widget found at the link so it works correctly on your widgets page.
- To get the bonus points, you have to convert the jQuery that governs the behavior of the password checking widget to javaScript (it has to behave correctly without using any jQuery) and submit that as your solution.

Questions?

Agenda

- Last Time:
  - HW 2 Demo and Overview
  - Lecture 3 Exercise Review
  - CSS
  - Intro to the DOM
- Today
  - Lecture 4 Exercise Review
  - DOM
  - JavaScript

Questions?

Review Lecture 4, Exercise 1:

- Update the CSS file mystyle8.css used by Box_example.html to uses relative styling that enables the previous example seamlessly scales to the window size

  Rel_Box_Example.html

- You can use the online book, your phone or computer **for reference**
- **You can download the files: Box_Example.html and mystyle8.css from the Week 3 module on the class Canvas site**
- One key definition 1em (a relative unit of measure) = 16px (an absolute unit of measure) in most browsers
- Note, there is one styling command that you must set to a % to make sure the relative styling works – part of your task is to find it and fix it!!!! (Hint see: https://www.w3schools.com/html/html_responsive.asp)

The Document Object Model (DOM)

- **What is the HTML DOM?**
  - The HTML DOM is:
  - The data structure that holds the structure (HTML) specified by a Webpage and the style applied to it via CSS, and is what your browser uses to render (display) a web page.
  - A standard object model for HTML
  - A standard programming interface for HTML
  - Platform- and language-independent
- A W3C standard
- The HTML DOM defines the **objects and properties** of all HTML elements, and the **methods**(interface) to access them (via JavaScript).
- Summary:
  - **The HTML DOM is a standard for how to represent/store, get, change, add, or delete HTML elements.**
  - **See https://www.w3schools.com/js/js_htmldom.asp**

HTML DOM Nodes

- In the HTML DOM (Document Object Model), everything is a **node**:
- The document itself is a document node
- All HTML elements are element nodes
- All HTML attributes are attribute nodes
- Text inside HTML elements are text nodes
- Comments are comment nodes

Element Objects

- In the HTML DOM, each **Element object** represents an HTML element tag.
- Element objects can have **child nodes** of type element nodes, text nodes, or comment nodes.
- A **NodeList object** represents a list of nodes, like an HTML element's collection of child nodes.
- Elements can also have attributes. Attributes are attribute nodes.

– What Examples of **HTML** Attributes (which are specified in HTML tags like <img> </img>)

Can You Think Of ???

Attributes used in the img tag

- src
- alt

HTML Attributes vs DOM Properties

- **When writing HTML source code, you can define attributes on your HTML elements**.

- **Then, once the browser parses your code, a corresponding DOM node will be created. This node is an object, and therefore it has properties.**
- For instance, this HTML element:

  <input type="text" value="Name:">

  has 2 attributes.
- Once the browser parses this code, a HTMLInputElement object will be created, and this object will contain dozens of properties like: accept, accessKey, align, alt, ***attributes***, autofocus, baseURI, checked, childElementCount, childNodes, children, classList, className, clientHeight, etc.
- For a given DOM node object, properties are the properties of that object, and attributes are the elements of the attributes property of that object.

discussion courtesy of : http://stackoverflow.com/questions/6003819/properties-and-attributes-in-html

IN Summary

- Attributes – are what's written in HTML Elements.
  - https://www.w3schools.com/html/html_attributes.asp
- Properties (derived from HTML attributes) – are what's in DOM objects.
  - https://www.w3schools.com/jsref/dom_obj_attributes.asp

DOM Hierarchy

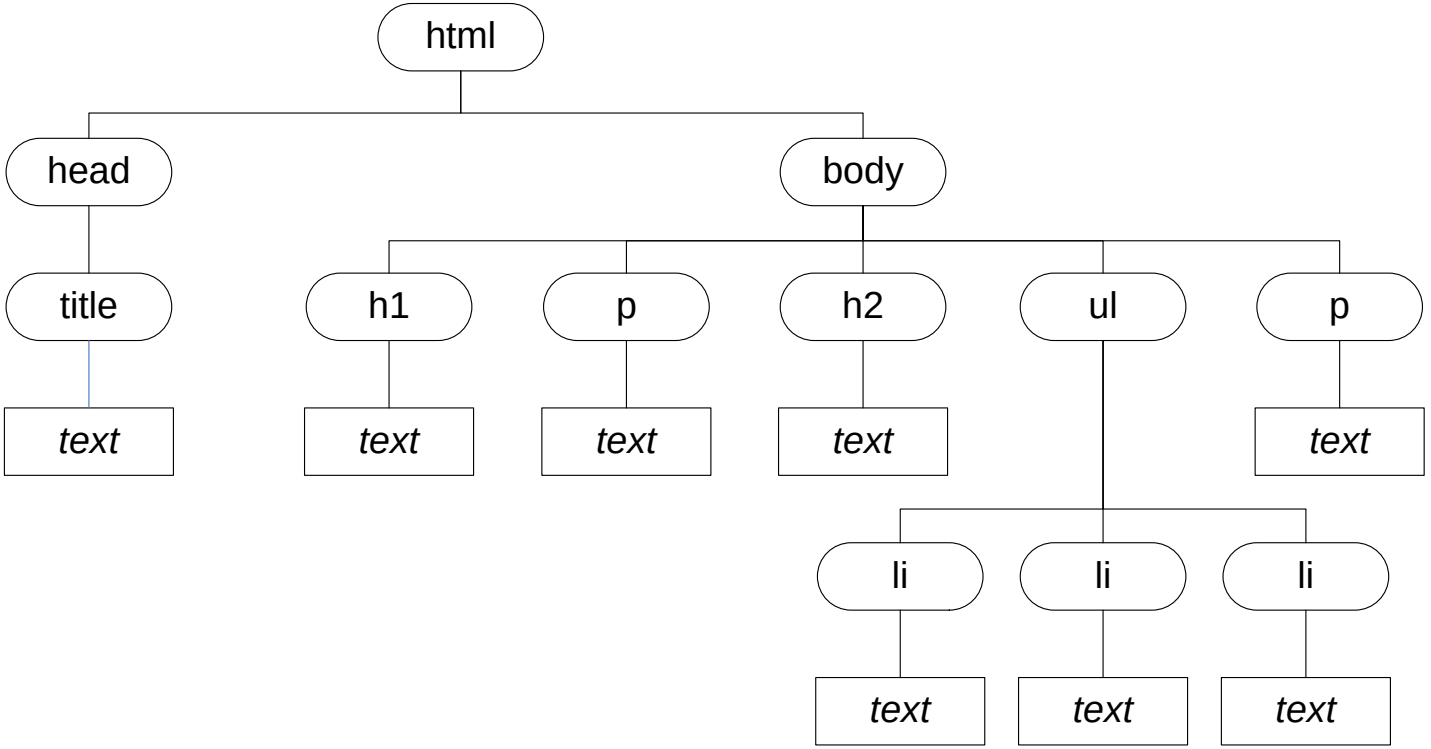
- https://www.w3schools.com/js/js_htmldom.asp

Consider the Following Web Page:

```
<!DOCTYPE html>
<html>
  <head>
                    <meta charset = "utf-8">
                    <title>An Example That Illustrates The Document Object Model</title>
  </head>
  <body>
                    <h1> Activities Available at the University of Minnesota </h1>
                    <p>  There are a multitude of things to do here at the University of
                              Minnesota.  You can attend sporting events, concerts, exercise, join a
                              club, get a work-study job, or just chill with your friends - and those are
                    just a few possibilites</p>
                    <h2> Entertaining, but under the radar activities</h2>
                    <ul>
                              <li>Attend a  Gopher track meet</li>
                              <li>Go to a Gopher baseball game</li>
                              <li>Play ultimate frisbee on the quad</li>
                    </ul>
                    <p>That is my list - yours may be quite different.
                      But, the point is, you should get involved in an activity
                      of your choice, research shows that it will
                      help your grades and enhance your college
                      experience </p>
  </body>
</html>
```

Dom_ex.html

Is this the DOM for the Web Page specified on the previous slide?



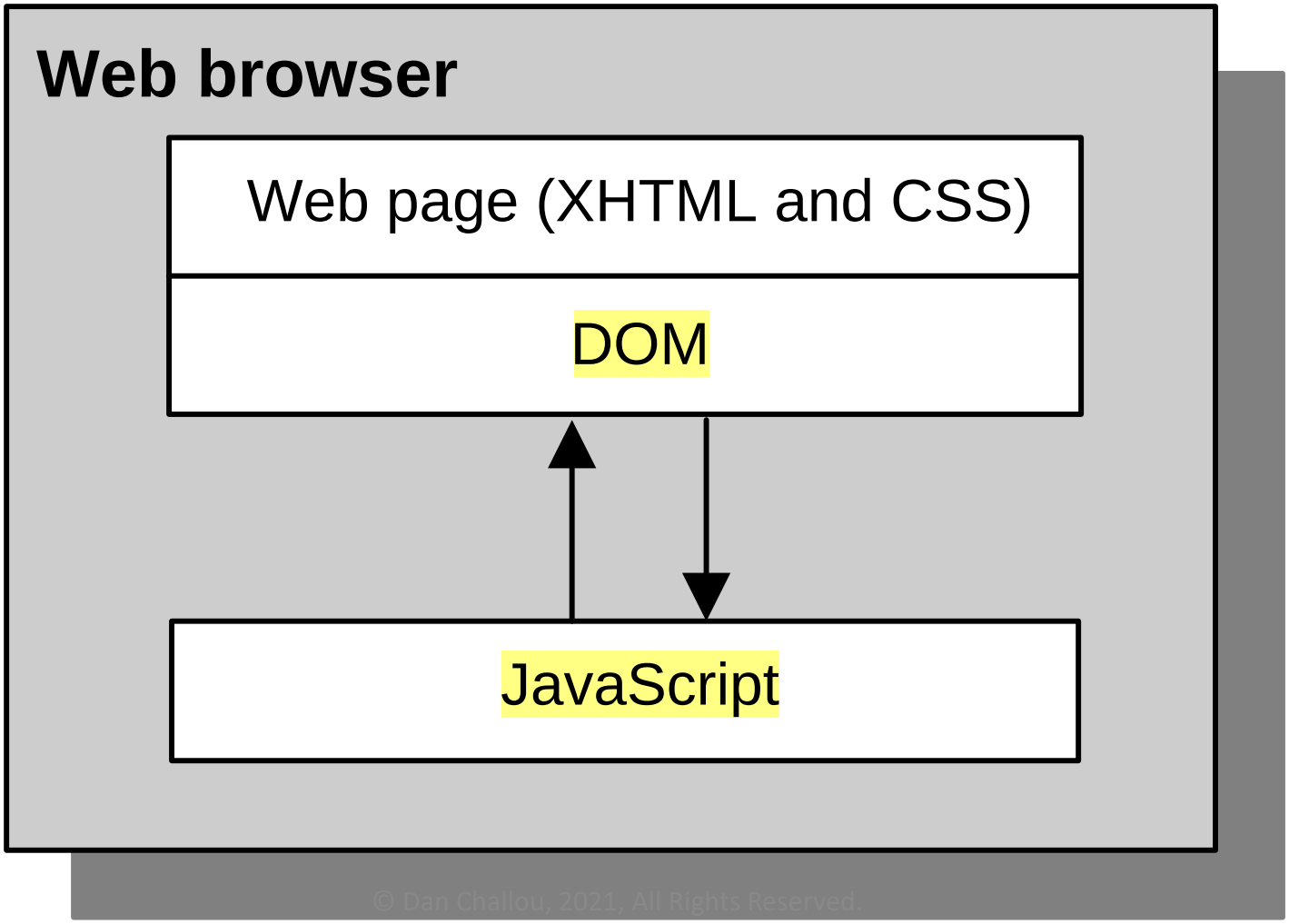
Please Answer with Yes hands up OR NO stand up

Questions?

The Document Object Model - Summarized

- The Document Object Model is a collection of nodes – **OBJECTS** - (in a tree) that store the data necessary render the current web page in a browser.
- The DOM for a web page is built as the page is loaded by the Web Browser
- Javascript can modify the web page in the browser by modifying the DOM
- When the DOM is changed, the web browser immediately displays the results of the change
- Many (Most?) JavaScript Applications manipulate the DOM based on user actions

How Web Technologies Interact in a Web Browser



JavaScript

- Is not Java, its syntax is closer to C, but it isn't C either
- JavaScript is dynamically typed (C, C++, Java – statically typed)
- It is closer in nature to functional languages (Scheme, Lisp, Closure, OCaml, Smalltalk,…)
- It uses objects, and enables you to create your own objects
- It does have classes (a relatively recent addition):

  https://javascript.info/getting-started
- It does support inheritance (is-a, has-a), but it excels at the latter
- See: http://www.crockford.com/javascript/javascript.html for an overview

What can in-browser JavaScript do?

- In-browser JavaScript can do everything related to webpage manipulation, interaction with the user, and the webserver.
- For example, in-browser JavaScript can:
  - Add new HTML to the page, change the existing content, modify styles.
  - React to user actions, run on mouse clicks, pointer movements, key presses.
  - Send requests over the network to remote servers, download and upload files (so-called AJAX and COMET technologies).
  - Get and set cookies, ask questions to the visitor, show messages.
  - Remember the data on the client-side ("local storage").

What Can't in-browser JavaScript do?

- JavaScript's abilities in the browser are limited for the sake of the user's safety. The aim is to prevent an malicious actor from accessing private information or harming the user's data.

What in-browser JavaScript CAN't do:

- JavaScript on a webpage may not read/write arbitrary files on the hard disk, copy them or execute programs. It has no direct access to OS system functions.
- Different tabs/windows generally do not know about each other. Sometimes they do, for example when one window uses JavaScript to open the other one. But even in this case, JavaScript from one page may not access the other if they come from different sites (from a different domain, protocol or port). (same origin policy)
- JavaScript can communicate over the net to the server where the current page came from. But its ability to receive data from other sites/domains is crippled.

The limits specified above do not exist if JavaScript is used outside of the browser, for example on a server. Modern browsers also allow plugin/extensions which may ask for extended permissions.

Consider the following JavaScript (intent: read in 2 numbers, compute & print sum and product)

```
<script>
    var number1; // first string entered by user
    var number2; // second string entered by user
    var sum; // sum of number1 and number2
    var product; // product of number1 and number2

    number1 = window.prompt( "Enter first integer" ); // 6 entered by user
    number2 = window.prompt( "Enter second integer" ); //5 entered by user

    sum = number1 + number2; // add the numbers
    product = number1 * number2; //multiply the numbers

    document.writeln( "<h1>The sum is " + sum + "</h1>" );
    document.writeln("<h1>The product is " + product + "</h1>" );
</script>
```

What is printed when the JavaScript above is executed, and 6 and 5 are input?

add_n_mult.html

Exercise 1 – **submit your answer via the Lecture 5, Exercise 1 submission item in the week 3 module on Canvas**

- The JavaScript function **parseInt(*argument*)** takes a string as an ***argument*** and returns an integer
- Example: **parseInt("23")** returns the integer number ***23***
- Use the function **parseInt** to rewrite the JavaScript given on slide 23 so that when 6 and 5 are entered:
  - The program returns 11 as the sum, and
  - 30 as the product
- **You can download the web page add_n_mult.html from the week 3 module on Canvas and edit it!**

Please close computer when done…

Questions?

Some Key Language Properties

- **JavaScript has two scopes – global and local**. Any variable declared outside of a function belongs to the global **scope**, and is therefore accessible from anywhere in your code. Each function has its own **scope**, and any variable declared within that function is only accessible from that function and any nested functions
  - https://www.w3schools.com/js/js_scope.asp
- **JavaScript Hoisting** – in JavaScript, a variable can be used before it has been declared
  - https://www.w3schools.com/js/js_scope.asp
- See and do the JavaScript tutorial:
  - https://www.w3schools.com/js/default.asp

The var keyword in JavaScript

- Global Scope: Not much difference between using:

var name = “Dan”;
and
name = “Dan”;

Must use **var** if you declare a variable and do not assign it a value:
var name; // ok
name; // might not be ok

Declaring a variable with the **var** keyword in a function makes it local to the function (*if you don’t use the var keyword to declare a variable in a function, the variable has global scope*).

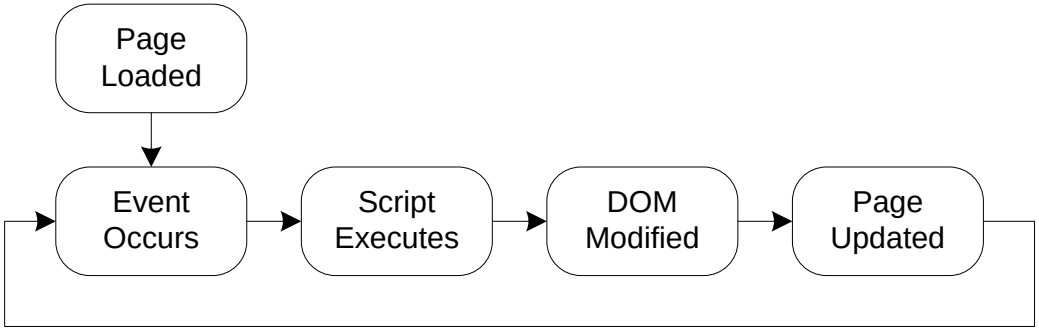
And learn how to find errors in, and debug, your JavaScript

- https://www.w3schools.com/js/js_errors.asp
- https://www.w3schools.com/js/js_debugging.asp

Examples of Plain old JavaScript functions (we'll get to closures later)

- functiondemo.html

DOM Scripting / Event Cycle



- DOM Scripting – uses JavaScript to manipulate the DOM
- DOM Scripting is event-driven. An event is typically such a clicking a mouse, moving your mouse over a DOM element or typing on the keyboard
- When the event occurs JavaScript code (an event handler) is executed to handle the event
- The event handler has full access to the DOM – it can change the properties of those elements
- When the DOM is modified, the browser detects those changes and updates the page
- When the event handler is finished, the web browser waits for another event to occur (and repeats the cycle when it does)
- DOM scripting should be done only to enhance a web page – this is known as progressive enhancement.

Simple Event Handling Example (**code along**) skipped this, did easier version

- eventDemo1.html

```
<html>
  <head>
    <meta charset = "utf-8">
    <title>Event Handler Demo</title>
    <script>
        // Dr Dan will write the script here
    </script>
 </head>
  <body>
            <p> <input type="button" value="Click to Trigger“ onclick=“display_click()” /> </p>
</html>
```

Are there other mechanisms available to set the behavior of the onclick event???

- Yep, one is to set the attribute(s) of HTML Element(s) to call JavaScript.
- HTML Elements (Buttons, Text Boxes, Paragraph Elements – have all sorts of attributes that can be set to respond to events like mouse events, keyboard events, etc.) See:

https://www.w3schools.com/tags/ref_eventattributes.asp

Example, using events on a DOM node. The events mouseover and mouseout are set to respond to those events on a DOM img element

- https://www.w3schools.com/js/tryit.asp?filename=tryjs_events_onmouseover
- What is the difference between the two approaches?
- **Different way of doing the same thing, can use JavaScript to add/remove behavior on attributes**

Other Methods JavaScript Can Use to Access/Change DOM Elements

- Use the methods the Document (of the DOM) provides

- https://www.w3schools.com/js/js_htmldom_document.asp
- https://www.w3schools.com/js/js_htmldom_elements.asp

Examples that use different mechanisms to display and hide items on a Webpage

- pop_up.html
- displayPixEx.html
- https://www.w3schools.com/howto/howto_js_toggle_hide_show.asp
- https://www.w3schools.com/css/css_display_visibility.asp
- https://www.w3schools.com/cssref/pr_class_visibility.asp

Questions?

Next Time

- JavaScript and Event handling revisited
- Animation via JavaScript
- Regular Expressions for checking form data
- JavaScript Closures
- JavaScript Event Handling Revisited
- Google Maps?