Csci 4131 Internet Programming
Fall 2021
Lecture 8
October 4th

Instructor: Dr. Dan Challou

© 2021, Daniel J. Challou. All rights reserved. Do not copy, share or redistribute without the express written consent of the author

Logistics – Csci 4131 Lecture 8, October 4th

- Homework 2 Grades are posted on Canvas
- Questions or issues – please follow the procedure documented in the syllabus (start by heading to an office hour with one of the TA's or emailing the help email:

  csci4131help-f21@umn.edu

Remember fastest ways to get questions answered or issues addressed are (in order of most rapid response to less rapid response) are

1) go to an in person or virtual office hour
2) email the help email,
3) post to the discussions on Canvas

We'll eventually respond to emails sent to our x.500 email addresses – but that can take quite a bit more time (a day to a few (3-4) days – or more)

➢ **zyBooks Lecture Prep due Wed 10/6 and zyBooks HW 5 due 10/9 – check your zyBook for specifics and due date / time!**

© 2021, Daniel J. Challou. All rights reserved. Do not copy, share or redistribute without the express written consent of the author

Logistics – Csci 4131 Lecture 8, Oct 4th

- ***The homework 3 specification is available in the week 4 module***
- ***Homework 3 is*** **due this Friday 10/8** ***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, and started designing, implementing and testing HW 3 functionality*** **- you are NOW** ***SIGNIFICANTLY*** **behind.**

© 2021, Daniel J. Challou. All rights reserved. Do not copy, share or redistribute without the express written consent of the author

Dr C’s Office Hour Time Changed Today (10/4)

- Office hour will begin at 3:45 and end at 4:45
- Have to fill in for my 2081 Lab!

- I Apologize for the late notice!!!!

© 2021, Daniel J. Challou. All rights reserved. Do not copy, share or redistribute without the express written consent of the author

Reading & Tutorials

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)

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

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-f21@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.

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

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

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

Questions?

© 2021, Daniel J. Challou. All rights reserved. Do not copy, share or redistribute without the express written consent of the author

Agenda

- Last Time
  - Lecture 6 Exercise Review
  - JavaScript
    - Automation Wrap-up
    - Regular Expressions
    - JavaScript Closures
- Today
  - More JavaScript
    - JavaScript Closures
    - Introduction to Google Maps
    - Event-handling wrap-up

© 2021, Daniel J. Challou. All rights reserved. Do not copy, share or redistribute without the express written consent of the author

Questions?

© 2021, Daniel J. Challou. All rights reserved. Do not copy, share or redistribute without the express written consent of the author

Review Lecture 7, Exercise 1

- Download the file **phoneNumExEmptyForm.html** from the week 4 module on the Class Canvas site
- Use the regex from our Interactive Exercise to create an input form that accepts a SYNTACTICALLY VALID telephone number of the form: **xxx-xxx-xxxx**

^[1-9][0-9]{2}-[0-9]{3}-[0-9]{4}$
^[1-9]\d{2}-\d{3}-\d{4}$

Try with and without ^ and $
by updating the file **phoneNumExEmptyForm.html**

- Demo – phoneNumEx2.html
- Hint: Set the **pattern** attribute (that is **pattern="regex"**) in a text input field used to enter the phone number to your regular expression (in double quotes), for example:
  "insert regex from Interactive Exercise on previous slide"

HTML input element (tag) has a pattern element – check zyBooks, w3 Schools, (or Sebesta)

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

Questions?

© 2021, Daniel J. Challou. All rights reserved. Do not copy, share or redistribute without the express written consent of the author

Recall Closures (in Computer Programming)

source:
https://en.wikipedia.org/wiki/Closure_(computer_programming)

- In programming languages, **closures** (also **lexical closures** or **function closures**) are techniques for implementing lexically scoped name binding in languages with first-class functions.
- Operationally, a closure is a record storing a function together with an environment: a mapping associating each free variable of the function (variables that are used locally, but defined in an enclosing scope) with the value or reference to which the name was bound when the closure was created.
- A closure—unlike a plain function—allows the function to access those *captured variables* through the closure's copies of their values or references, even when the function is invoked outside their scope.

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

Sources offering clear discussion on JavaScript Scope and Closures

- Your zyBooks – Chapter 8, Section 5
- https://robertnyman.com/2008/10/09/explaining-javascript-scope-and-closures/

- Take Away
  - Closures are expressions, usually functions, which work with variables ***set at the time the function is called*** within a certain context.
  - More specifically, inner function(s) that bind/use local variables of the outer function that creates and returns them are closures.

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

Recall Example: What is Displayed in the Alert Box??? *Please formulate your best answer without running the code*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset = "utf-8">
    <title>Example of simple function closure</title>
            <script>
                        function addN (x) {  // this returns a closure
                            return function (y) {
                                return x + y;
                            };
                        };

                        var add3 = addN(3);  // addN(3) creates a function  with x bound to 3
                                             // and returns it, and sets the identifier add3 to refer to it
                        var result = add3(5);
                        alert(result);
            </script>
  </head>
  <body>
  </body>
</html>
```

simpleclosure.html

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

Why is this a bad closure?!

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset = "utf-8">
        <title>Example of incorrect function closures</title>
        <script> // what does the DOM look like after this page is loaded????
            function addLinks () {
                for (var i=1, link; i<6; i++) {
                    link = document.createElement("a");
                    link.innerHTML = "      Link" + i + "<br><br>";
                    link.onclick = function () {
                            alert("This is link: " + i);
                        };
                    document.body.appendChild(link);
                } // end for
            } // end addLinks
            window.onload = addLinks;
        </script>
    </head>
    <body>
    </body>    BadClosure.html
</html>
```

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

Side by side

- Incorrect Closure
- BadClosure.html
- Correct Closure
- cclosure_v2.html

© 2021, Daniel J. Challou. All rights reserved. Do not copy, share or redistribute without the express written consent of the author

So, how do we figure out what went, or is going wrong in JavaScript

- Use debugging constructs!
- Use a debugger!
- http://www.w3schools.com/js/js_debugging.asp
- https://developers.google.com/web/tools/chrome-devtools/javascript/breakpoints
- Chrome Debugging Tutorial:

https://developer.chrome.com/docs/devtools/javascript/

- Firefox Debugging Tutorial:

https://developer.mozilla.org/en-US/docs/Tools/Debugger

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

Questions?

© 2021, Daniel J. Challou. All rights reserved. Do not copy, share or redistribute without the express written consent of the author

- With Homework assignment 3, we turn our focus from just the client (Browser), over to the server side (Web Services - Google API's for maps, HTTP Protocol, Server Side Scripting)

- Remember, a video demo and overview is available in the Media Gallery in the file named:

  Csci 4131 Fall 2021 Homework 3 Functional Demonstration

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

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 an insert it to get them to work

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

Approach to Putting Icons to Mark each Location on your Contacts list 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

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

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

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

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.

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

Questions?

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

More on JavaScript Events

- DOM events
  - Enable scripts to respond to user interactions and modify the page accordingly
- Events and event handling (via JavaScript)
  - help make web applications more dynamic and interactive

Take a look at any web page with JavaScript in it: in nearly all cases there will be an event that triggers the script. The reason is very simple. JavaScript is meant to add inter*activity* to your pages: the user does something and the page reacts.

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

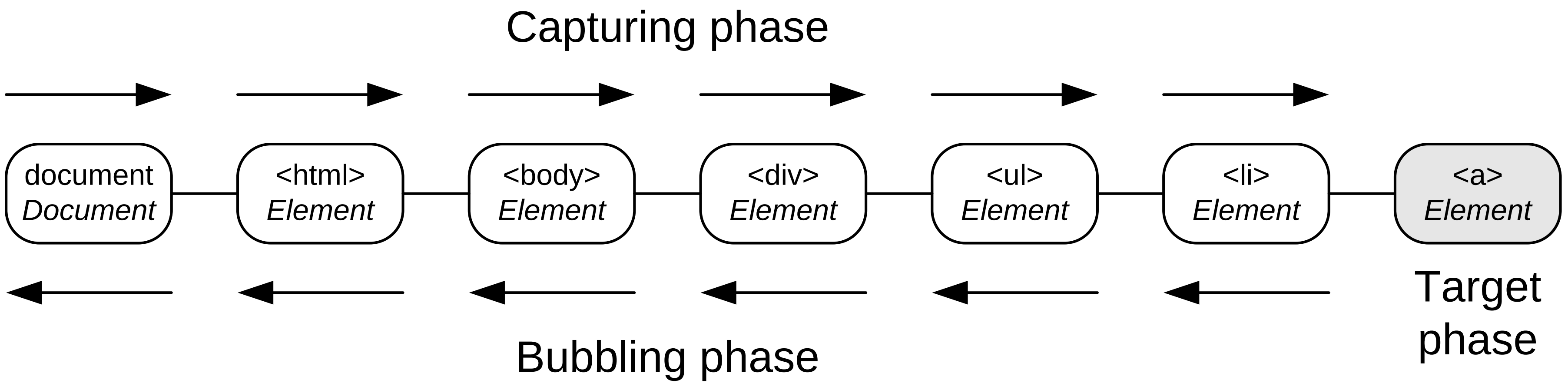
© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

Three Phases of Event Dispatch

- Capturing – the event travels downward from the document object to the target element
- Target – the event triggers on the target element
- Bubbling – the event travels upward from the target element to the document object

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

Event Dispatch in the DOM Tree



The bubbling phase of event dispatch is optional depending on the event type

Internet Explorer doesn’t support the capturing phase of event dispatch

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

Event Bubbling

- The process whereby events fired on *child* elements "bubble" up to their *parent* elements
- When an event is fired on an element, it is first delivered to the element's event handler (if any), then to the parent element's event handler (if any), then to its parent's parent, etc.
  - *If you are handling an event in a child element alone, you should cancel the bubbling of the event in the child element's event-handling code by using the `cancelBubble` property of the event object*

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

When is the event captured? – it is settable with addEventListener!

- Either Capture Phase (on the way down)
  - Or
- Bubble Phase (on the way up)

- Use to set it - addEventListener(event, eventhandler, true or false);
  - Default is false, captured during bubbling phase
  - If true, captured during capture phase

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

Examples

- During Bubble Phase:

  bubbleEx.html

- During Capture Phase

  captureEx.html

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

Common Types of Events

- HTML event – An event that is triggered by the HTML or XHTML page
- Mouse event – An event that is triggered by the user’s mouse
- Keyboard event – An event that is triggered by the users keyboard.

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

Overview of HTML Event Types

- Load – triggers when the browser loads all the content in the document. Works the same as window.onload event
- Unload – triggers when browser removes a document from the window
- Submit – triggers when a form is submitted
- Reset – trigger when a form is reset
- Select – triggers when a user selects text in field
- Change – triggers when content of an element is changed
- Focus – triggers when an element gains focus
- Blur – triggers when an element loses focus

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

HTML Event We Have Seen: Window Object's Load Event

- The `window` object's `load` event fires when the window finishes loading successfully (i.e., all its children are loaded and all external files referenced by the page are loaded)
- *Every* DOM element has a `load` event, but the `window` object's load event is the most commonly used

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

Event Details

Event	Bubbles	Cancellable	Valid For
load	**N**	**N**	**Body**
unload	N	N	Body
submit	Y	Y	Form
reset	Y	Y	Form
select	Y	N	Input, textarea
change	Y	N	Input, select, textarea
focus	Y	N	a.area, input, select, textarea, button
blur	Y	N	a.area, input, select, textarea, button

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the
express written consent of the author

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.

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

Caveat and Exercise 1

- 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 (i.e., before all the elements in the body are loaded)
- If such a statement attempted to execute **getElementById** for an HTML element in the body, then **getElementById** would return null

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

- ***Put your answer in the Lecture 8, Exercise 1 Submission Item on Canvas in the week 5 module***

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author

Next Time

- Closure wrap-up / Event Wrap-up
- Introduction to the HTTP Protocol

© Dan Challou, 2021, All Rights Reserved.
Do not share or reproduce without the express written consent of the author