More Javascript

Procedural Programming

- In Data Structures and most other CS classes:
  - Program Flow looks like:
    - start program
      - program runs continuously, vigorously and linearly
        - possibly less vigorous at times
          - sleep
          - waiting for things (file reading, etc)
      - program ends
- Multithreaded programs can look a little different

Event Driven Programming

- Program start:
  - define functions
  - initialize variables
  - start "event listeners"
  - LOOP:
    - On event
      - run code to handle event
    - END LOOP

- It is hard to really say that an event-driven program is really one program.
  - It is more that each event is its own program but all the programs live in a shared space.

- The program spends most of its time doing NOTHING but waiting for events
  - Almost all javascript programs are event driven

Javascript in HTML pages

- Javascript may be either directly in page between <script></script> or in an included file
- Javascript may be either in header or in body
- Javascript blocks are evaluated strictly in order in which they appear within html page

```
<html>
    <head>
        <script>
            console.log("hello head 1");
        </script>
        <script src="sim.js"></script>
        <script>
            console.log("hello head 2");
        </script>
    </head>
    <body>
        <script>
            console.log("hello body 1");
        </script>
       <script>
            console.log("hello body 2");
        </script>
    </body>
</html>
```

https://cs.brynmawr.edu/~gtowell/383/js1.html
https://cs.brynmawr.edu/~gtowell/383/sim.js

Namespaces in Javascript

- All Javascript runs in a single namespace
  - vars/functions created in one block are available in all other blocks
  - Error to use vars before they are declared
- Using backquotes

```
<html>
    <head>
        <script>
            let headCounter=1
            console.log("hello1 head %d",
headCounter++);
        </script>
        <script>
            console.log("hello2 head",
headCounter++, bodyCounter);
        </script>
    </head>
    <body>
        <script>
            let bodyCounter=1;
            console.log(`body part1 $
{headCounter++} ${bodyCounter++}`);
        </script>
       <script>
            console.log("body part2",
headCounter++, bodyCounter++);
        </script>
    </body>
</html>
```

https://cs.brynmawr.edu/~gtowell/383/js2.html

JS and body text

- Evaluation order in html page redux
  - It is not just JS, but everything that is evaluated in order.
    - All DOM is built in textual order
  - So, pages elements are unavailable to JS before they appear
- Conclusion: any JS that modifies a page -- immediately -- should be last

```
<html>
    <body>
        <script>
          document.querySelector("#span1").text="rep 1";
        </script>
        <span id="span1" >Original 1</span>
        <br/>
        <span id="span2" >Original 2</span>
        <script>
          document.querySelector("#span2").innerHTML="rep 2";
        </script>
    </body>
</html>
```

https://cs.brynmawr.edu/~gtowell/383/js3.html

JS and events

- Realistically, almost never do things immediately in JS.
  - other than initializations
- Instead execute a function when an event occurs.
- There are lots of events, these are common

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

full list = https://www.w3schools.com/jsref/dom_obj_event.asp

```
<html>
    <body onload="f();">
        <script>
            function f() {

document.querySelector("#span1").innerHTML="r
ep b";

document.querySelector("#span2").innerHTML="r
ep c";
            }
        </script>
        <span id="span1" >Original 1</span>
        <br/>
        <span id="span2" >Original 2</span>
    </body>
</html>
```

https://cs.brynmawr.edu/~gtowell/383/js4.html

Events

Can be anywhere

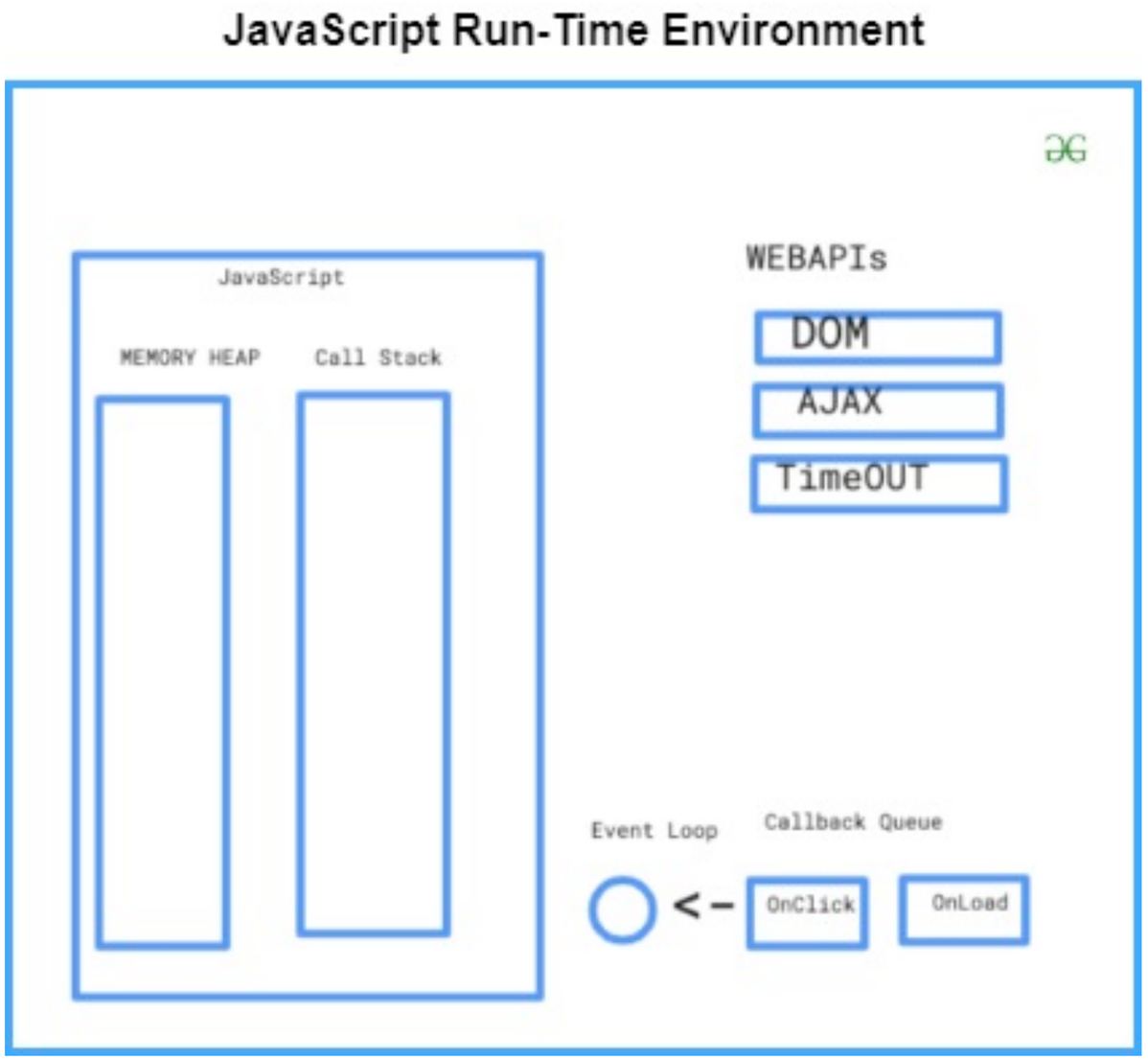
- The can be associated with almost any html element
  - Often this is a mistake, but it can make for very interactive html pages

```
<html>
    <body>
        <script>
            let hoverCount=1;
            function g() {

document.querySelector("#span1").innerH
TML=`first ${hoverCount++}`;

document.querySelector("#span2").innerH
TML=`second ${('c'+hoverCount++)}`;
            }
        </script>
        <span onmouseover="g();"
id="span1" style="font-
size:300%;color:red">Original 1</span>
        <br/>
        <span onclick="g();" id="span2"
style="font-
size:400%;color:blue">Original 2</span>
    </body>
</html>
```

https://cs.brynmawr.edu/~gtowell/383/js5.html

Javascript is single-threaded and non-blocking

Event Driven



https://www.geeksforgeeks.org/why-javascript-is-a-single-thread-language-that-can-be-non-blocking/

https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop

Single threaded == javascript only ever does one thing at a time

Non-blocking = JS never gets stuck waiting for an event (bad programmers can write a classic sleep loop)

Event-driven = after setup program waits for event and then responds per the program setup.

Event driven and timing

```
<html>
    <body>
        <script>
            let count=0;
            function replaceAfter(timeee) {

document.querySelector("#span1").innerHTML=`Started $
{count}`;
                for(let i=0; i<timeee; i++) {
                    for (let j=0; j<timeee; j++) {
                        for (let k=0; k<timeee; k++) {
                            let z = i*j*k;
                        }
                    }
                    console.log(i);
                }
                count++;

document.querySelector("#span2").innerHTML=`Finished $
{count}`;
            }
        </script>
        <span id="span1" style="font-
size:300%;color:red">Original 1</span>
        <br/>
        <span onclick="replaceAfter(1500);" id="span2"
style="font-size:400%;color:blue">Original 2</span>
    </body>
</html>
```

https://cs.brynmawr.edu/~gtowell/383/js6.html and js7.html

```
<html>
    <body>
        <script>
            let count=0;
            function replaceAfter(timeee) {

document.querySelector("#span1").innerHTML=`Started $
{count}`;
                setTimeout(function() {
                    count++;

document.querySelector("#span2").innerHTML=`Finished $
{count}`;
                }, timeee);
            }
        </script>
        <span id="span1" style="font-
size:300%;color:red">Original 1</span>
        <br/>
        <span onclick="replaceAfter(5000);" id="span2"
style="font-size:400%;color:blue">Original 2</span>
    </body>
</html>
```

- Javascript does not have sleep() because it would be blocking,
  - You can kind of simulate it (as at left)
  - DO NOT!!!

Timers ..... are tricky

```
<html>
    <body>
        <script>
            let count=0;
            function doit() {
                count++;

document.querySelector("#span2").innerHTML=`Finished $
{count}`;
            }
            function replaceAfter(timeee) {

document.querySelector("#span1").innerHTML=`Started $
{count}`;
                setTimeout(doit(), timeee);
            }
        </script>
        <span id="span1" style="font-
size:300%;color:red">Original 1</span>
        <br/>
        <span onclick="replaceAfter(1500);" id="span2"
style="font-size:400%;color:blue">Original 2</span>
    </body>
</html>
```

```
<html>
    <body>
        <script>
            let count=0;
            function doit() {
                count++;

document.querySelector("#span2").innerHTML=`Finished ${count}
`;
            }
            function replaceAfter(timeee) {

document.querySelector("#span1").innerHTML=`Started ${count}
`;
                setTimeout(function() { doit() }, timeee);
            }
        </script>
        <span id="span1" style="font-
size:300%;color:red">Original 1</span>
        <br/>
        <span onclick="replaceAfter(1500);" id="span2"
style="font-size:400%;color:blue">Original 2</span>
    </body>
</html>
```

https://cs.brynmawr.edu/~gtowell/383/js8.html

- the first argument is evaluated, and the result of that evaluation is placed in the timer queue.
  - So the left does NOT work.

Javascript Practice

- Write a javascript web page to
  - Have an 2 elements that respond to a mouse click
  - First element: On click replace the element click on with something else
  - Second element: on click replace content immediately with 10. After 1 second replace with 9. Do this until 0. (Ie make a countdown timer from 10).

- Doing More:
  - Create a span element containing text line "NOT hovering"
    - only when hover over change text to "Stop looking over my shoulder"

Port Registration v1

```
<body>
  <center>
    <h1>CS383 -- Node port regsitration</h1>
    <h2>Before you start using a port in Node ..</h2>
  </center>

<form action="reserveport" method="post">
  <table >
    <tr><td>Port</td>
        <td><input type="text" size="20" id="hello1"   name="port"></td></tr>
    <tr><td>Name</td>
        <td><input type="text"   size="50" name="uname"></td></tr>
    <tr><td>Student ID</td>
        <td><input type="text"   size="50" name="studid"></td></tr>
    <tr><td>Use of Port</td>
        <td><input type="text"   size="100" name="portuse"></td></tr>
   <tr> <td> <span style="font-size: 150%;"><input type="submit" value="RegisterPort" name="reserve"  style="background-color:green; width:250px; font-size: 150%;"> </span> </td>  <td>
    </td></tr>
</table>
</form>


<form action="checkport" method="post">
    <span style="font-size: 150%;"><input type="submit" value="Just Check" name="reserve"  style="background-color:magenta; width:250px; font-size: 150%;"> </span>
</form>

<a href="app5.js">Click here </a> to see the node.js code underlying this page

</body>
</html>
```

Port Registration v2

```
<body>
  <center>
    <h1>CS383 -- Node port regsitration</h1>
    <h2>Before you start using a port in Node, check here and register your use. </h2>
<br> <br>
  </center>

  <table >
    <tr><td>Port</td>
        <td><input id="portt" type="text" size="20" id="hello1"   name="port"></td></tr>
    <tr><td>Name</td>
      <td><input id="user" type="text"   size="50" name="uname"></td></tr>
    <tr><td>Student ID</td>
      <td><input id="student" type="text"   size="50" name="studid"></td></tr>
    <tr><td>Use of Port</td>
      <td><input id="usage" type="text"   size="100" name="portuse"></td></tr>
   <tr> <td> <span  onclick="doQuery();" style="background-color:green; width:250px; font-size: 150%;"> Submittt </span> </td>  <td>
    </td></tr>
</table>

    <tr> <td> <span onclick="doCheck();" style="background-color:magenta; width:250px; font-size: 150%;"> Check Port Usage </span> </td>
      </td></tr>
  </table>
<p></p>
<a href="app6.js">Click here </a> to see the node.js code underlying this page

<div id="gt2"></div>


</body>
</html>
```

Javascript for Port registration

Check Only

```
function doCheck() {
    let params = {
            method: "POST",
        headers: { 'Content-type': 'application/json' },
        body: JSON.stringify({ 'port': 1 })
    }
    let uurl = "checkport"
    console.log("Querying server ....")
    console.log(params)
    fetch(uurl, params)
        .then(function(response) {
            response.text().then(function(text) {
                data = JSON.parse(text);
                console.log(data);
                document.querySelector("#gt2").innerHTML = "";
            });
        });
}
```