# More Javascript

# Do more with DBs

Interested?
contact emmalinetaylor@yahoo.com.

# Next assignment (#3)

- Posted on class website

- Alternate assignment for one/two people
  - become a DBA / Sysadmin.
    - Figure out why the class server occasionally hangs
      - For details talk to me

# 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
- Lots of different ways to log

```
<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="replaceme
nt 1";
        </script>
        <span id="span1" style="font-
size:300%;color:red">Original 1</span>
        <br/>
        <span id="span2" style="font-
size:400%;color:blue">Original 2</span>
        <script>

document.querySelector("#span2").innerHTML="repl
acement 2";
        </script>
    </body>
</html>
```

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

# JS and events

- Realistically, almost never do things immediately in JS. Use functions and events.
  - other than declare global vars
  - and create functions
- 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
eplacement b";

document.querySelector("#span2").innerHTML="r
eplacement c";
            }
        </script>
        <span id="span1" style="font-
size:300%;color:red">Original 1</span>
        <br/>
        <span id="span2" style="font-
size:400%;color:blue">Original 2</span>
    </body>
</html>
```

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

# Events

## Can be anywhere

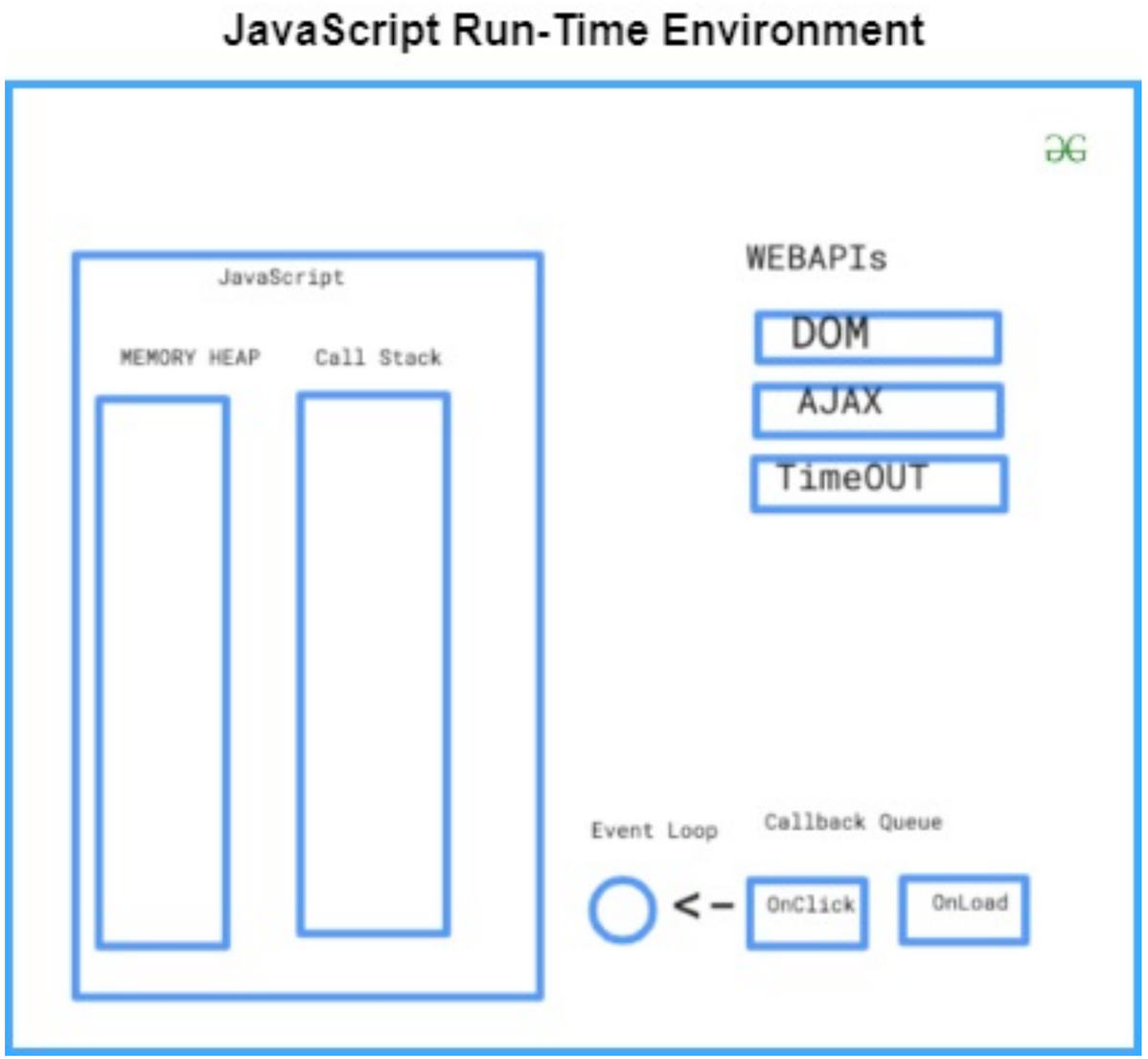
- The can be associated with almost any html element
  - Often this is a mistake, but it can make for very interactive html pages

```
<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 things at a time

Non-blocking = JS ne ver 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.

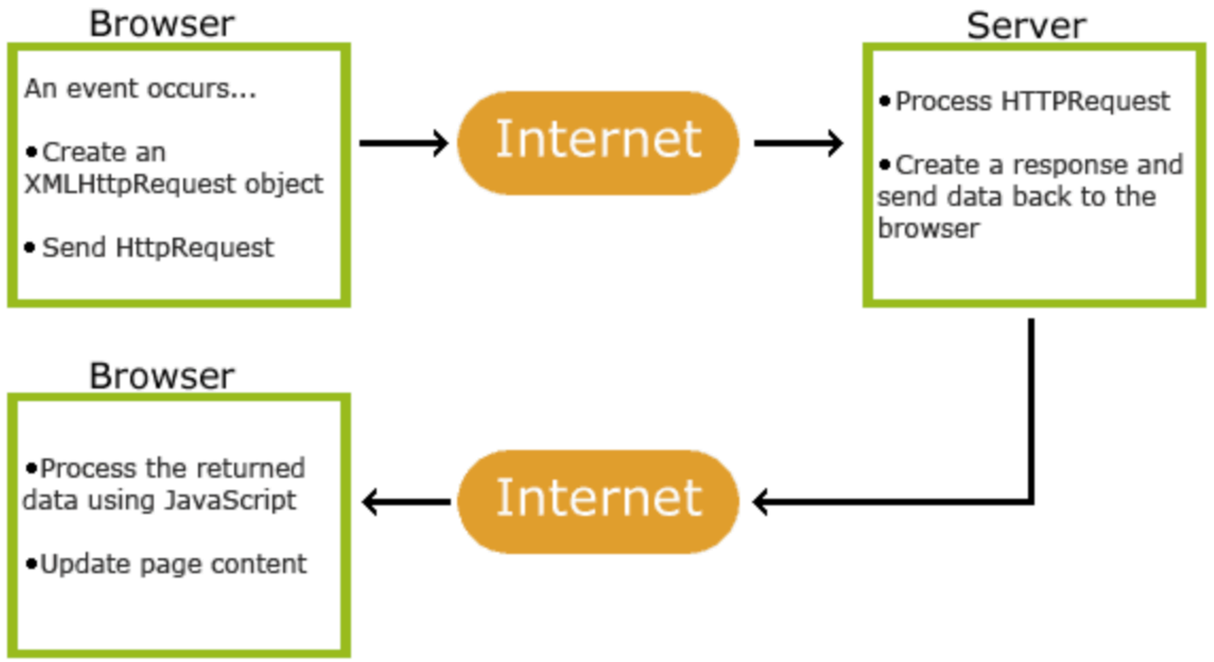
# Client-Server communications

- Whole page reload
  - forms (which can be built on the fly in JS)
- JS based
  - fetch
- Continuing Needs
  - Polling
  - Long Polling
  - Server sent events
  - Web sockets

# AJAX

## *A*synchronous *J*avaScript *A*nd *X*ML.

1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript



# 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).
- For experts:
  - Create a span element containing text line "NOT hovering"
    - only when hover over change text to "Stop looking over my shoulder"