Javascript and SQL

Non-Blocking in Javascript

- Anything that could block must be handled asynchronously
- Generally this means creating an anonymous function containing the things you want to do async
- The anon function gets evaluated at load, but the function is not executed until the async event triggers

setInterval like setTimeout except it repeats.
(How do you kill it?)

Anonymous function

Anonymous function

```
<html>

    <script language="javascript">
        let pollCount=0;
        function dotime() {
            setInterval(function() {
                fetch("http://165.106.10.170:30001/short")
                .then(function(response) {
                    pollCount++;
                    response.text().then(function(text) {
                        console.log(`Poll Count ${pollCount}`)
                        console.log(text);
                    });
                });
            }, 2000);  //polls every 2 seconds
        }
    </script>
    <body>
        <h1 onclick="dotime()">Short Polling</h1>
    </body>
</html>
```

Anonymous function

https://cs.brynmawr.edu/~gtowell/383/Lec10/short_poll.html
http://loin.cs.brynmawr.edu:30001/short_fetch.html

Same as last slide, but with async/await

And some other stuff

```
<html>

    <script language="javascript">
        let pollCount=0;
        let intervalOb = null
        function dotime() {
            if (intervalOb!=null) {
                clearInterval(intervalOb)
                intervalOb=null
            } else {
            intervalOb = setInterval(async function() {
                let response = await fetch("http://loin.cs.brynmawr.edu:30001/get-test")
                let text = await response.text()
                pollCount++;
                document.querySelector("#counter").innerHTML=`Poll Count ${pollCount}`
                document.querySelector("#texter").innerHTML=`From server ${text}`;
            }, 2000);  //polls every 2 seconds
        }
    }
    </script>
    <body>
        <h1 onclick="dotime()">Short Polling</h1>
        <div id="counter"></div>
        <div id="texter"></div>
    </body>
</html>
```

Most languages have blocking constructs

- Most languages have some form of parallel execution
  - "Thread"
  - So even if one thread is blocked thing can keep happening
    - For example, a simple hello world server in Go with a 15 second sleep block.
    - Problems with using threads to solve blocking?

- Android / iOS both disallow blocking in the main IO thread

```go
package main

import (
    "fmt"
    "net/http"
    "time"
)


func hw(w http.ResponseWriter, req *http.Request) {
    fmt.Println("Enter hello")
    time.Sleep(15*time.Second)
    fmt.Fprintf(w, "hello world")
    fmt.Println("Exit hello")
}


func main() {
    http.HandleFunc("/hello", hw)
    http.ListenAndServe(":30030", nil)
}
```

Explain what happens, Why?
when click on the span1 element

```
<html>
    <body>
        <script>

            function ra() {
                let ii=23;
                setTimeout( function() {

document.querySelector("#span1").innerHTML=`${ii}`;
                }, 5000)
                ii += 19;
            }
        </script>
        <span onclick="ra();" id="span1"
style="font-size:300%;color:red">Original 1</span>
        <br/>
    </body>
</html>
```

//q_1.html

If this is not what you want,
how do you change code?

```
<html>
    <body>
        <script>
            let c=0;
            function funny(fnc, delay) {
                console.log(`here ${c++}`)
                setTimeout(fnc, delay);
            }

            function ra(argg) {
                let ii=23;
                let gg = function() {

document.querySelector("#span1").innerHTML=`${ii}
${argg}`;
                    argg+=30;
                };
                funny(gg,1000);
                ii += 19;
                funny(gg, 2000);
            }
        </script>
        <span onclick="ra(12);" id="span1"
style="font-size:300%;color:red">Original 1</
span>
        <br/>
    </body>
</html>
```

//q_2a.html