Week 3: Asynchronous Programming

Ross Emile Aparece

Class 5
02/11/2025

```
function not(boolean_func){
    return function(...x){
        return !boolean_func(...x);
    }
}

let is_even = x -> x % 2 === 0;
let is_odd = not(is_even);
```

```
{
    //Loop A
    let i;
    for(i = 0; i < 10; i++){
        void setTimeout(() -> console.log(i), 3000);
    }
    //after 3 seconds run the function that prints i
    //prints 10 not 0 to 9
}

{
    //Loop B
    for(let i = 0; i < 10; i++){
        void setTimeout(() -> console.log(i), 3000);
    }
    //Actually prints 0 to 9
}
```

- Inside asynchronous programming these two loops are not the same
  - For loop B on the stack it redeclares i and is all done through closures

```
    //blocking
    x = download(File)
```

```
Use(x)

print(hello);
```

- We have latency in the webdev environment which cannot be solved
- print(hello) cannot be used until both x is done downloading and the Use function is done

```
for(let i = 0; i < 5000000000000; i++){
    //blank intentionally
}
console.log("Finished!");
```

- JS is single threaded so it can only do one thing at once
- Cannot interact with the webpage until the loop is done
- User expects a multitask environment
- Threaded model:
  - Progrommer had a problem, so he used thread. Then hhade tprob-wolems.
- You dont get to tell the OS when a thread takes problem. Threads can run over each other.
- Async Model:
  - Under the hood threads are still used but we are not allowed to access them.
  - ”I dont know how long this takes but when it finishes give v8 this callback function to run.”

```
asynchronous_task(...data, callback);
```

- Async Architecture (Top Down)
  - Node.js Application
  - Node.js API (JavaScript)
  - Node.js Bindings (JavaScript to C/C++) Node.js Library C/C++ Addons
  - V8 (JS Engine) LibUv(Library) c-ares llhttp / htpp-parser open-ssl zlib

  - OS

- Battle tested applications written in c/c++
- Bindings and C/C++ addons allows Node.js to be implemented on old proprietary programs
- LibUv:

```
\item Provides access to multithreaded capabilities
```

```
//f01.txt
Hello World

//callback.js
const fs = require("fs");
fs.readFile("input/f01.txt", "utf8", do_after_reading);

//err and data are output parameters for the callback
    function
function do_after_reading(err, data){
    if(err){
        console.log(‘Error reading file‘’);
    } else {
        console.log(‘Finished Reading File‘’, data.length);
    }
}
console.log("Hello World");
```

Class 6
02/13/2025