Week 12

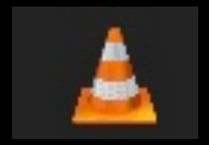
Javascript Foundation

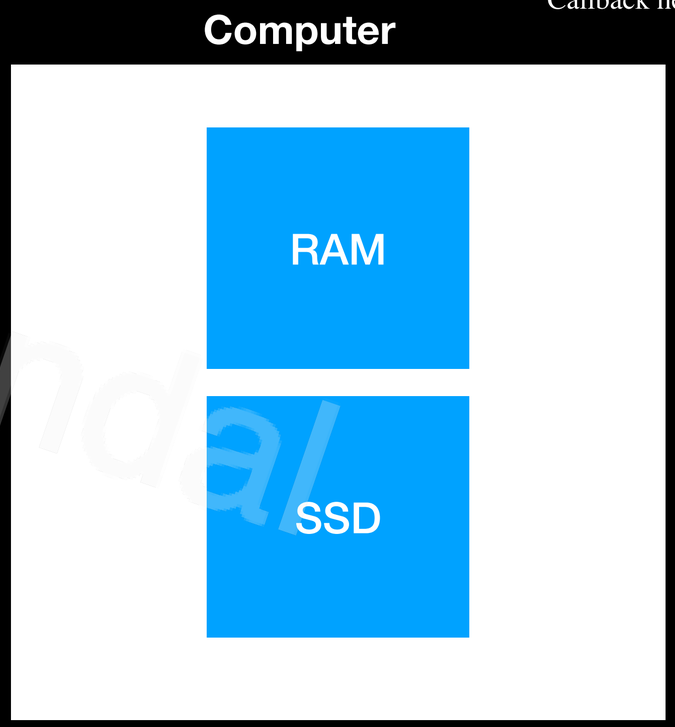
Week 12

Javascript Foundation

Why languages?
Interpreted vs compiled languages
Why JS >> Other languages in some use-cases
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, strings, booleans)
Complex primitives in JS (arrays, objects)
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callback queue, Asynchronous programming
Callback hell and Promises

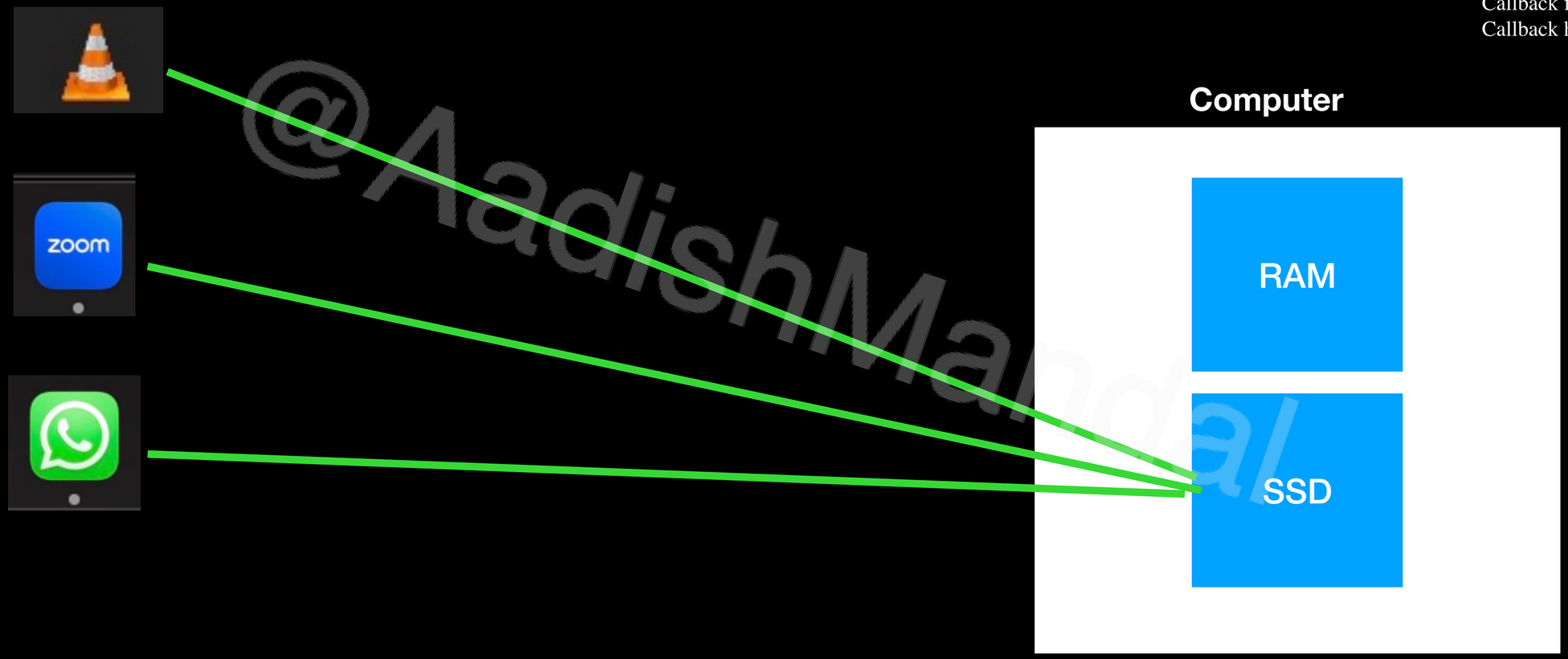
Why languages?





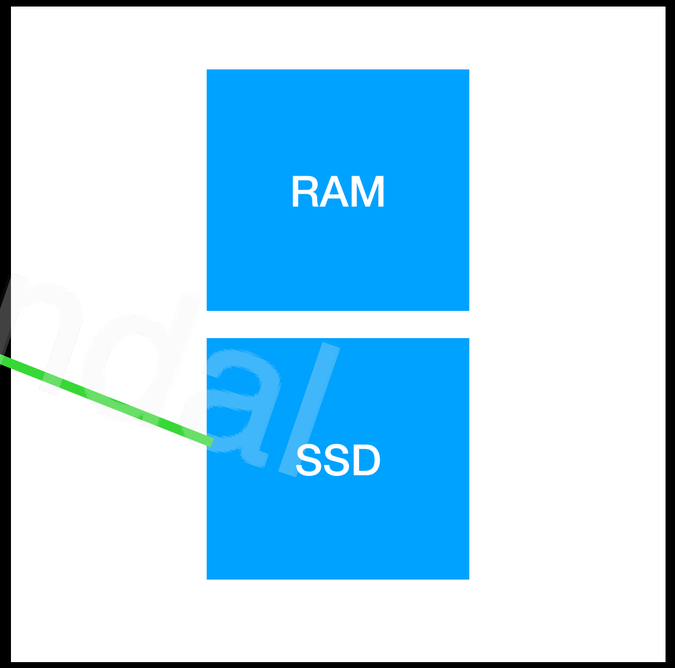
Why languages?
Interpreted vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, strin
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

Why languages?

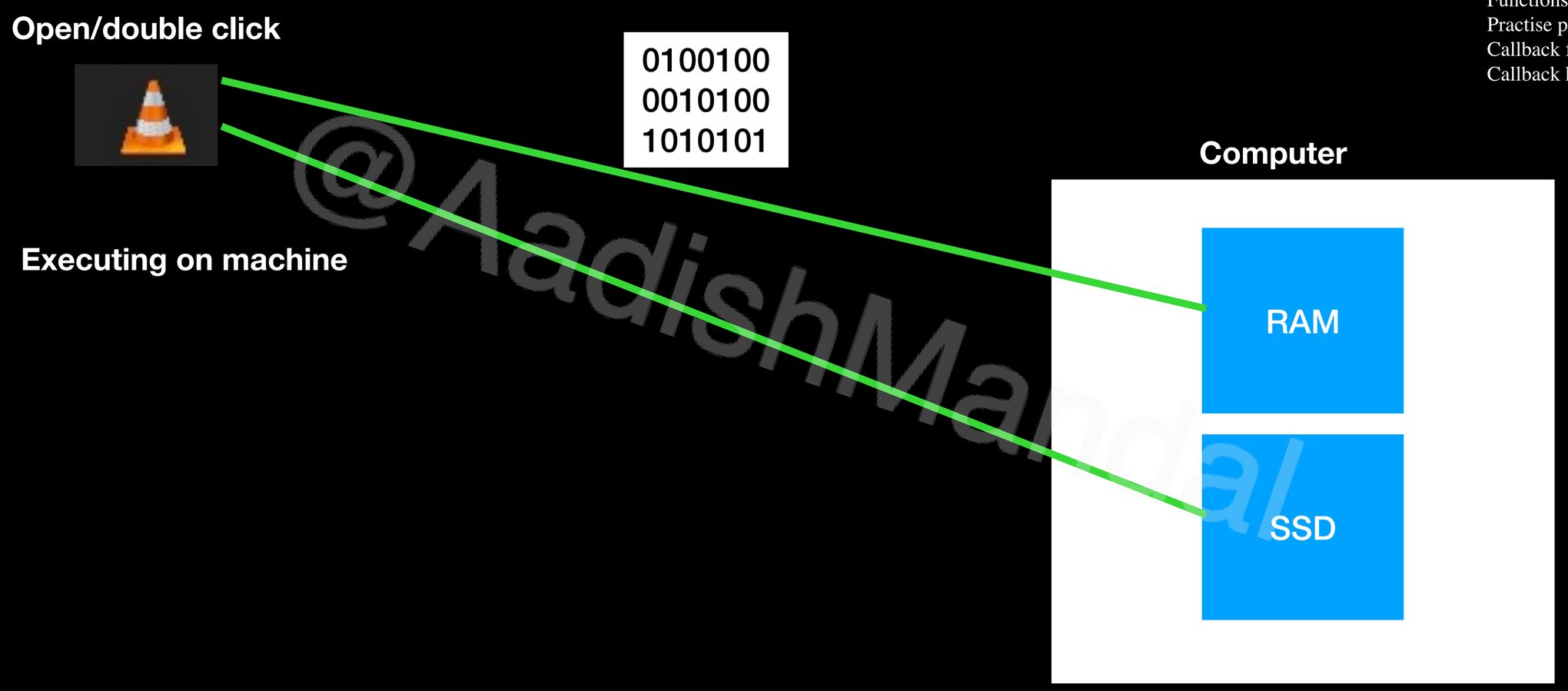


Why languages?

Statically present on machine



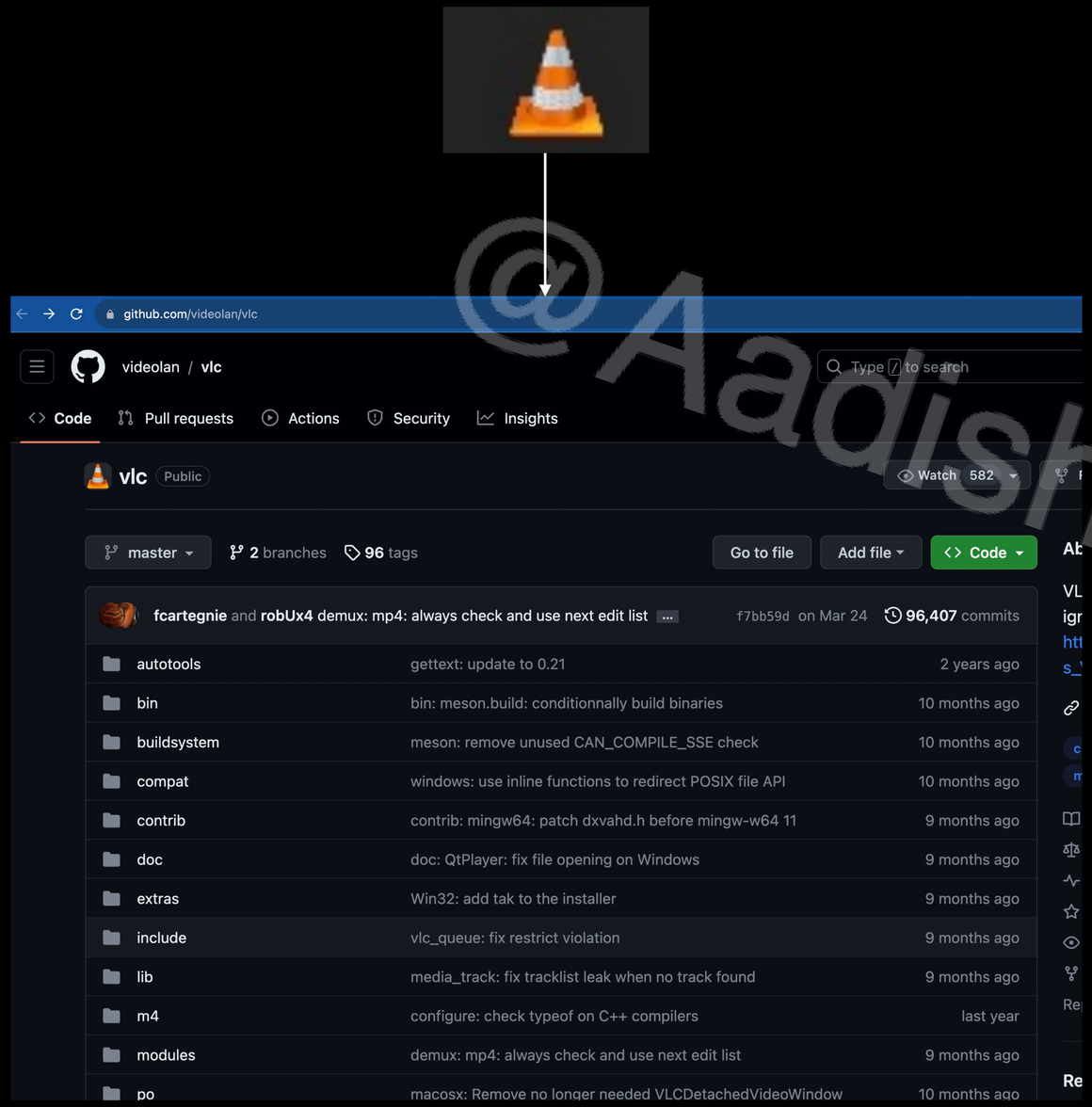
Why languages?



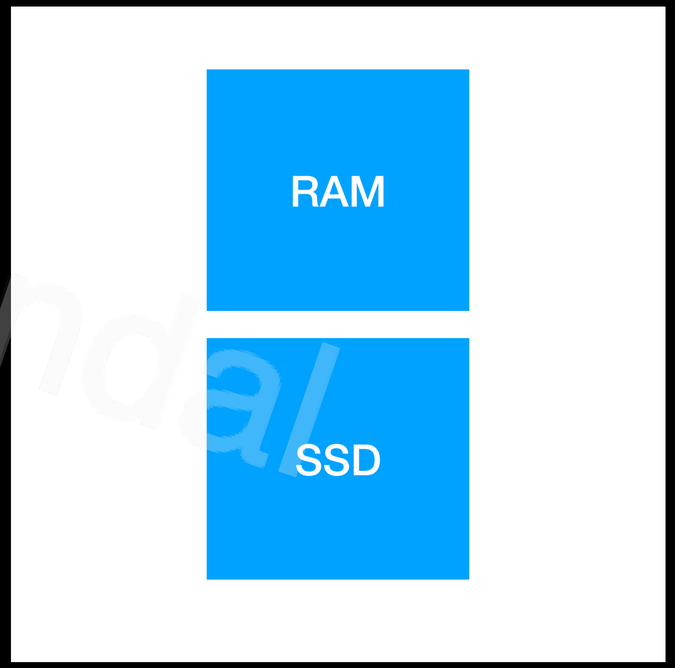
Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, strin
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

Why languages?

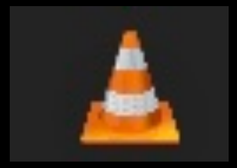
Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, strin
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

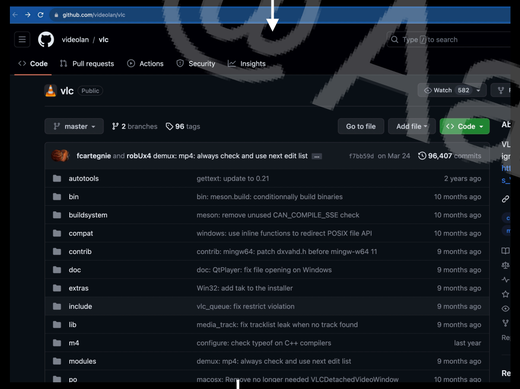


Computer



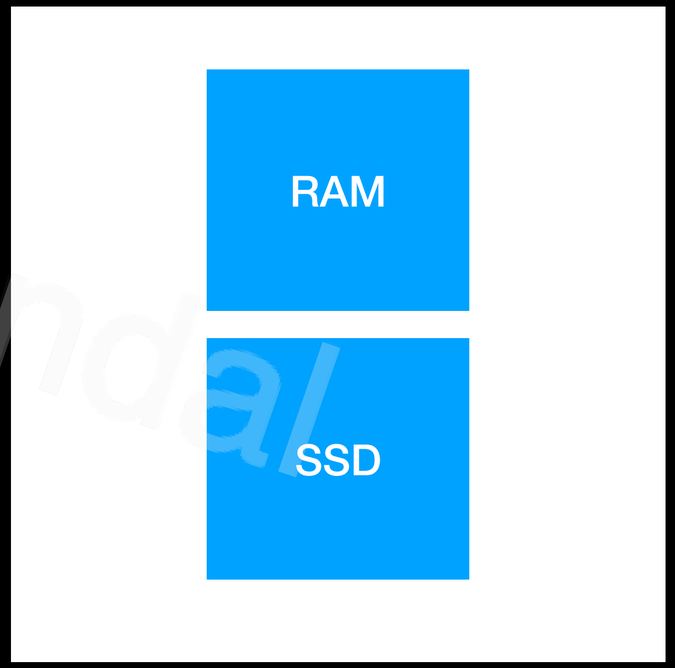
Why languages?





0100100
0010100
1010101

Computer



Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, strin
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

Why languages?
What have we learned?

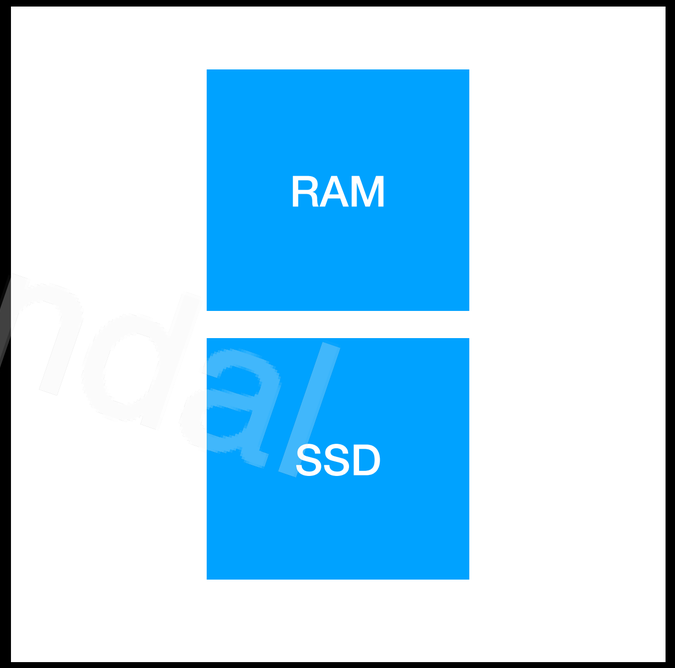
1. Languages are used to write applications
2. Developers write high level code in these languages
3. Every language has a compiler which converts the developer code into 01

C
C++
Java
Javascript
Golang
Rust

Compiler

0100100
0010100
1010101

Computer



Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, strin
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

Interpreted vs compiled languages

Compiler

What are compilers - Compilers convert high level developer friendly code into 0s and 1s

Interpreted vs compiled languages

Compiler

Lets see it in action - The C++ compiler is called g++

Step 1 - write code

```
#include <stdio.h>
using namespace std;


int main() {
  cout << "hello world" << endl;
  return 0;
}
```

Interpreted vs compiled languages

Compiler

Lets see it in action - The C++ compiler is called g++

Step 1 - write code

```cpp
#include <stdio.h>
using namespace std;


int main() {
  cout << "hello world" << endl;
  return 0;
}
```

Step 2 - Compile code

```
➜  100xdevs g++ a.cpp -o temp
```

Interpreted vs compiled languages

Compiler

Lets see it in action - The C++ compiler is called g++

Step 1 - write code

```cpp
#include <stdio.h>
using namespace std;


int main() {
  cout << "hello world" << endl;
  return 0;
}
```

Step 2 - Compile code

```
100xdevs g++ a.cpp -o temp
```

Step 3 - Run the code (put it in ram

```
→  100xdevs ./temp
hello world
```

Interpreted vs compiled languages

Compiler

But JS is different (interpreted)

Step 1 - write code

```
console.log("Hello world");
```

Interpreted vs compiled languages

Compiler

But JS is different (interpreted)

Step 1 - write code

```
console.log("Hello world");
```

Step 2 - Run code

```
→ 100xdevs node a.js
Hello world
```

Interpreted vs compiled languages

Compiler

Compiled languages

1. First need to compile, then need to run
2. Usually don't compile if there is an error in the code
3. Example - C++, Java, Rust, Golang

Interpreted Languages

1. Usually go line by line
2. Can run partially if the error comes later
3. Example - Javascript, Python

Interpreted vs compiled languages

Lets write some code

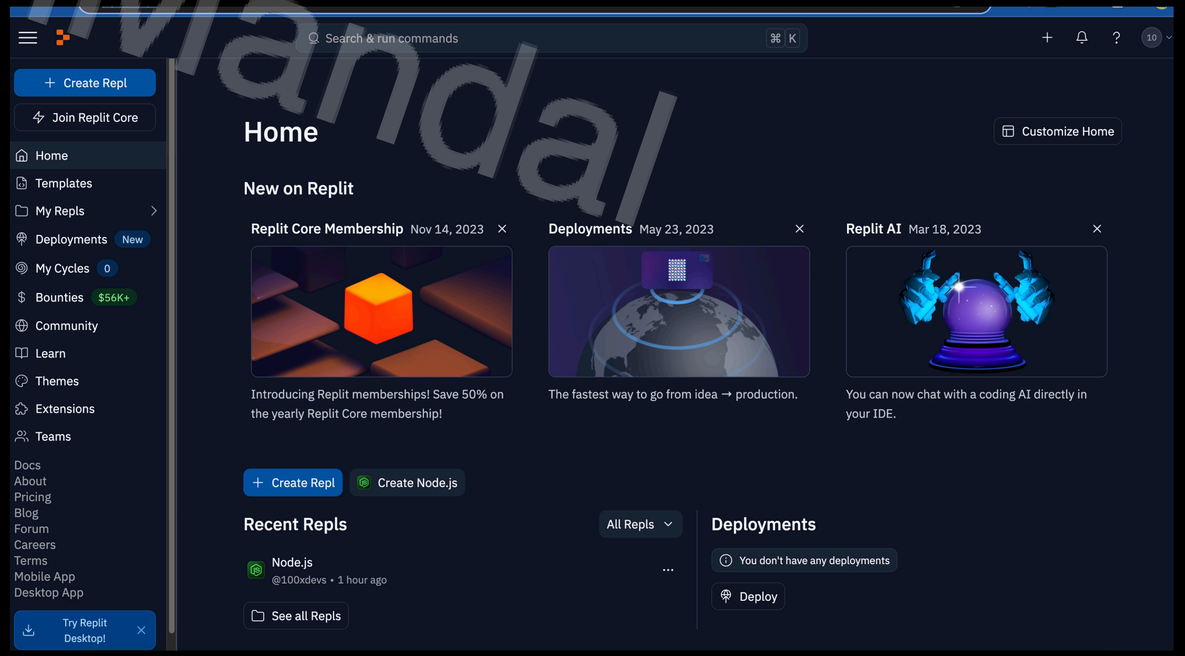
Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, strin
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

Please sign up on repl.it

Why repl.it? - It's lets you compile (or interpret?) javascript code without having it locally on your machine
lets try to run the hello world program

index.js

```
1  console.log("hi there");
2
```



Interpreted vs compiled languages

Lets write some code

Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, strin
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

Now run this code

index.js

index.js

```
1  console.log("hi there");
2  console.log(a);
```

Interpreted vs compiled languages

Lets write some code

Same code for C++


```cpp
#include <stdio.h>
#include <iostream>
using namespace std;


int main() {
  cout << "hello world" << endl;
  cout << a << endl;

  return 0;
}
```

```
➜  100xdevs g++ a.cpp -o temp
a.cpp:8:11: error: use of undeclared identifier 'a'
  cout << a << endl;
          ^
1 error generated.
➜  100xdevs
```

Interpreted vs compiled languages

What did we learn?

JS is an interpreted language
C++ is a compiled language
Interpreted languages go line by line while executing, can partially run until an error comes

Why is JS better than other languages

Browsers can only understand HTML/CSS/JS (not technically true)
Thanks to Node.js , Javascript can also be used for “Backend Development”

Static vs dynamic languages

C++

```cpp
#include <iostream>
using namespace std;

int main() {
    int number = 5;        // Declaration of an integer variable
    number = "Hello";      // This will cause a compile-time error

    cout << number << endl;
    return 0;
}
```

Benefits - More strict code

Javascript

```javascript
let number = 5;          // Variable initially holds a number
number = "Hello";        // Variable now holds a string

console.log(number);     // Outputs: "Hello"
```

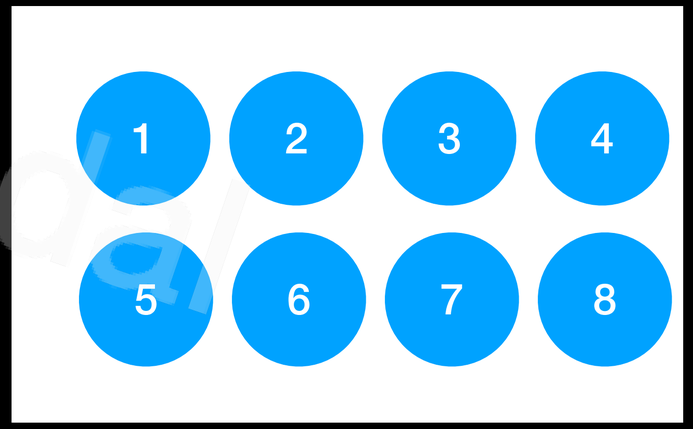
Benefits - Can move fast

Single threaded nature of JS

Hardware Overview:

Model Name:	MacBook Pro
Model Identifier:	MacBookPro18,2
Chip:	Apple M1 Max
Total Number of Cores:	10 (8 performance and 2 efficiency)
Memory:	32 GB
System Firmware Version:	7459.141.1

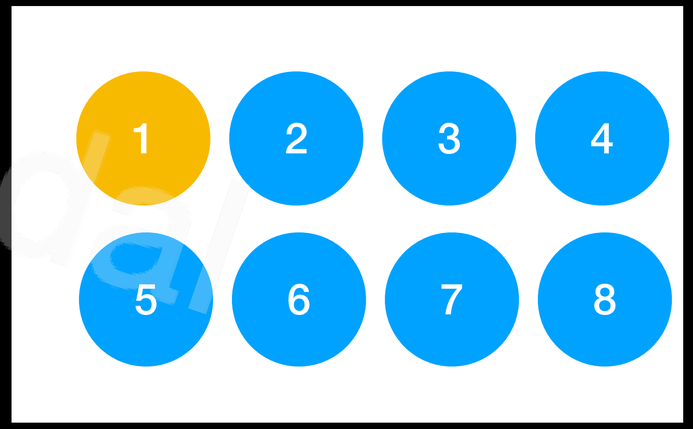
Mac Machine



Single threaded nature of JS

JS can only use one of these at a time
It is single threaded
This is why it is considered to be a bad language for scalable systems
There is a way to make it use all cores of your machine

Mac Machine



Single threaded nature of JS

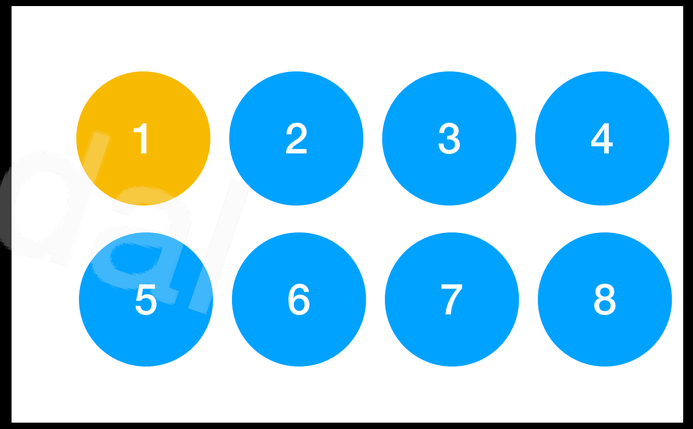
More practically, JS runs line by line and only One line runs at a time

index.js

index.js

```
1  console.log("hi there");
2  console.log(a);
```

Mac Machine



Simple primitives

Variables (let, var, const)
Data types - strings, numbers and booleans
If/else
Loops - For loop

Let's write some code -

1. **Write the program to greet a person given their first and last name**
2. **Write a program that greets a person based on their gender. (If else)**
3. **Write a program that counts from 0 - 1000 and prints (for loop)**

Complex primitives

1. Arrays
2. Objects

Let's write some code -

1. Write a program prints all the even numbers in an array
2. Write a program to print the biggest number in an arrya
3. Write a program that prints all the male people's first name given a complex object
4. Write a program that reverses all the elements of an array

Functions

Functions let you

1. Abstract out logic in your program
2. Take arguments as an input
3. Return a value as an output
4. You can think of them as an independent program that is supposed to do something given an input
5. Functions CAN take other functions as input - this will confuse you (callbacks)

Let's write some code -

1. Write a function that finds the sum of two numbers
2. Write another function that displays this result in a pretty format
3. Write another function that takes this sum and prints it in passive tense

Functions

Functions let you

1. **Abstract out logic in your program**
2. **Take arguments as an input**
3. **Return a value as an output**
4. **You can think of them as an independent program that is supposed to do something given an input**
5. **Functions CAN take other functions as input - this will confuse you (callbacks)**

https://gist.github.com/hkirat/898ac1da32b6b347a8c0c3e73e1c0666

index.js > ...

```javascript
function sum(num1, num2) {
    let result = num1 + num2;
    return result;
}

function displayResult(data) {
    console.log("Result of the sum is : " + data);
}

function displayResultPassive(data) {
    console.log("Sum's result is : " + data);
}

// You are only allowed to call one function after this
// How will you displayResult of a sum
```

Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some use-cases
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, strings, booleans)
Complex primitives in JS (arrays, objects)
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callback queue
Callback hell and Promises

Callback functions, event loops, callback queue

Synchronous vs Asynchronous functions

Synchronous

All the code we've written until now
All code running line by line (hence sync)

Asynchronous

Asynchronous functions in programming are those that allow a program to start a potentially long-running operation and continue executing other tasks without waiting for that operation to complete. This is particularly important in environments like web browsers or Node.js, where waiting for an operation to finish (like fetching data from a server or reading a large file) could make the application unresponsive.

Callback functions, event loops, callback queue

Synchronous vs Asynchronous functions

Synchronous

```
function sum() {
  let ans = 0;
  for (let i = 0; i<1000; i++) {
    ans = ans + i;
  }
  return ans;
}
```

Callback functions, event loops, callback queue

Synchronous vs Asynchronous functions

Asynchronous (setTimeout)

index.js > fetchData > ...

```javascript
function fetchData() {
  console.log('Requesting data from the ChatGPT server...');

  setTimeout(() => {
    console.log('Data received from the ChatGPT server: []');
  }, 3000);
}

fetchData();
```

Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, st
booleans)
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

Callback functions, event loops, callback queue

http://latentflip.com/loupe/

index.js > fetchData > ...

```javascript
function fetchData() {
  console.log('Requesting data from the ChatGPT server...');

  setTimeout(() => {
    console.log('Data received from the ChatGPT server: []');
  }, 3000);
}

fetchData();
```

Callback functions, event loops, callback queue

http://latentflip.com/loupe/

Better example

index.js > ...

```
1
2  setTimeout(function timeout() {
3      console.log("Click the button!");
4  }, 1000);
5
6  // Expensive operation (takes more than 1s)
7  let sum = 0;
8  for (let i = 0; i<10000000000; i++) {
9      sum = sum + 10;
10 }
11
```

Callback functions, event loops, callback queue

http://latentflip.com/loupe/

More examples?

Network calls
File system calls
Database calls
setInterval

Callback hell, Promises

Disclaimer - This is going to be overwhelming, especially for beginners
Please don't worry if you don't understand the next section, we don't need it for a while

Callback hell, Promises

https://gist.github.com/hkirat/502ea4573a045804be95083ce5af94dc

```javascript
// Function to simulate downloading data
function downloadData(callback) {
    setTimeout(function() {
        console.log("Data downloaded");
        callback("Downloaded Data");
    }, 1000);
}

// Function to simulate processing the downloaded data
function processData(data, callback) {
    setTimeout(function() {
        console.log("Data processed");
        callback("Processed " + data);
    }, 1000);
}

// Initiating the process
downloadData(function(downloadedData) {
    processData(downloadedData, function(processedData) {
        console.log("Final result: " + processedData);
    });
});
```

Callback hell, Promises

https://gist.github.com/hkirat/f7780b5061182b7281d37c23951e916d

```javascript
// Function to simulate downloading data, now returns a Promise
function downloadData() {
    return new Promise(function(resolve) {
        setTimeout(function() {
            console.log("Data downloaded");
            resolve("Downloaded Data");
        }, 1000);
    });
}

// Function to simulate processing the downloaded data, now returns a
function processData(data) {
    return new Promise(function(resolve) {
        setTimeout(function() {
            console.log("Data processed");
            resolve("Processed " + data);
        }, 1000);
    });
}

// Using Promises to handle the asynchronous operations
downloadData()
    .then(processData)
    .then(function(finalResult) {
        console.log("Final result: " + finalResult);
    })
    .catch(function(error) {
        console.error("An error occurred:", error);
    });
```


Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, st
booleans)
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

What's left?

Async await syntax in promises
Next week/offline video

Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, st
booleans)
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

Assignments

Why languages?
Scripting vs compiled languages
Why JS >> Other languages in some
Strict vs dynamic languages
Single threaded nature of JS
Simple primitives in JS (number, st
booleans)
Complex primitives in JS (arrays, obj
Functions in Javascript
Practise problem solving
Callback functions, Event loop, callba
Callback hell and Promises

For today -

1. Create a counter in Javascript (counts down from 30 to 0)
2. Calculate the time it takes between a setTimeout call and the inner function actually running
3. Create a terminal clock (HH:MM:SS)

There will be a video on how to install node.js and run tests locally for the main assignments for this week