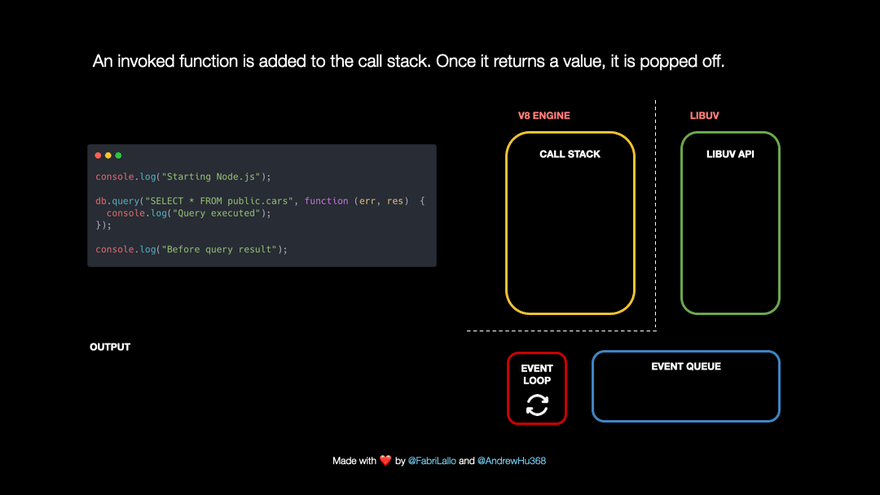
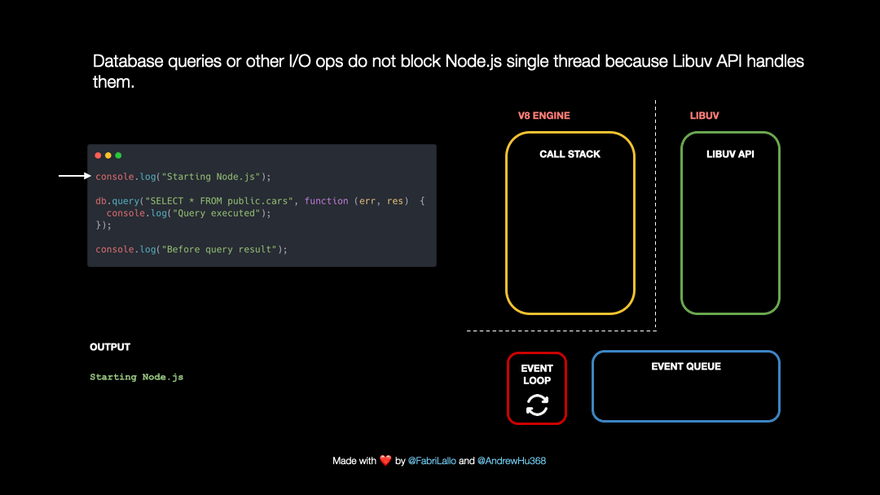
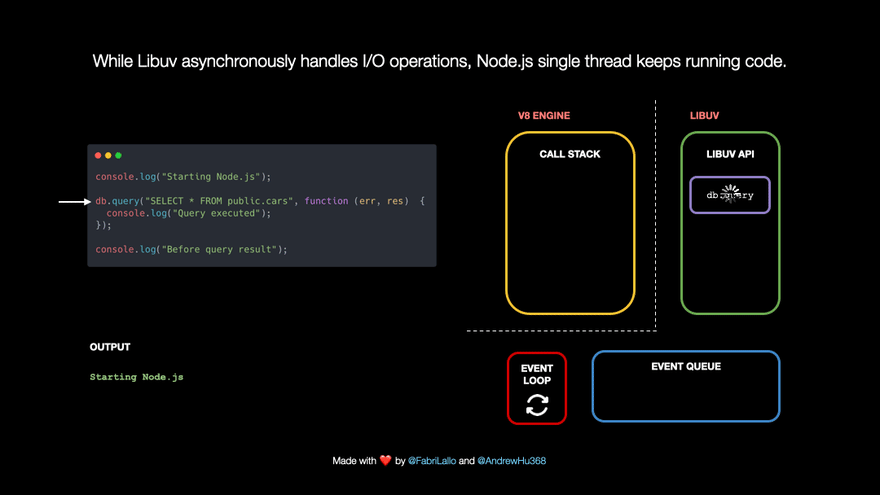
JS

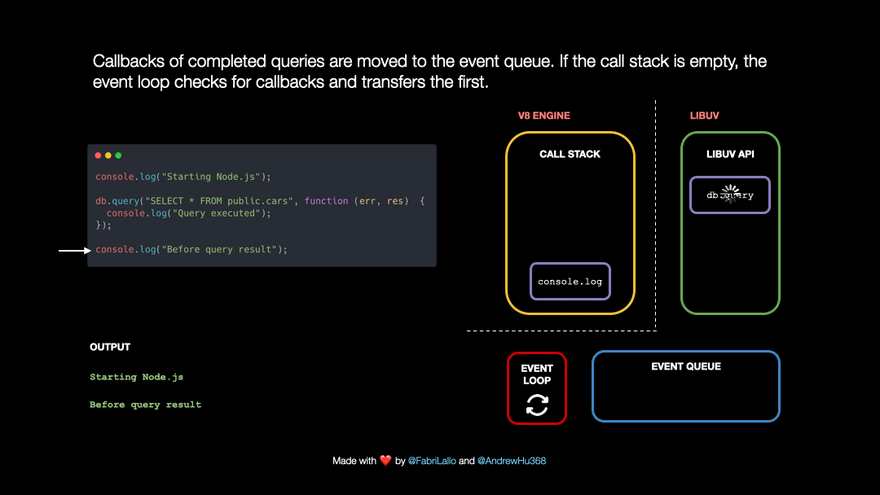
Async callbacks

Event loop









Pure function vs side effects

<https://blog.greenroots.info/what-are-pure-functions-and-side-effects-in-javascript>

call back

In JavaScript, you can also pass a function as an argument to a function. This function that is passed as an argument inside of another function is called a callback function.

// function

function greet(name, callback) {

console.log('Hi' + ' ' + name);

callback();

}

// callback function

function callMe() {

console.log('I am callback function');

}

// passing function as an argument

greet('Peter', callMe);

Prototype and proto

In JavaScript, when you create an object from a constructor function, it comes with a built-in property called `prototype`. This is like a blueprint for all the objects created from that constructor function. It defines the shared properties and methods that all instances of that object will have.

In the code above, the `getArea` method is contained in an object called `Circle.prototype`. When a new `Circle` instance is created, its `\_\_proto\_\_` property is set to `Circle.prototype`.

`\_\_proto\_\_` is a property of an object that points to its prototype. This property is used internally by the JavaScript engine to search for properties and methods on an object's prototype chain.

The difference between `\_\_proto\_\_` and `prototype` is simple: `\_\_proto\_\_` is a property of an object instance, while `prototype` is a property of a constructor function.



Currying

Currying is a function that takes one argument at a time and returns a new function expecting the next argument. It is a transformation of functions that translates a function from callable as f(a, b, c) into callable as f(a)(b)(c).



* Currying is a checking method to make sure that you get everything you need before you proceed
* It helps you to avoid passing the same variable again and again
* It divides your function into multiple smaller functions that can handle one responsibility. This makes your function pure and less prone to errors and side effects

Deep copy vs Shallow copy

https://www.freecodecamp.org/news/copying-stuff-in-javascript-how-to-differentiate-between-deep-and-shallow-copies-b6d8c1ef09cd/

IIFE



IIFE stands for Immediately Invoked Function Expression. It is a design pattern in JavaScript where a function is defined and executed immediately after its creation. IIFE is often used to create a private scope for variables, preventing them from polluting the global scope.

Function composition

Function Composition is a technique in which you combine two or more functions to produce a new function. The idea is to take the output of one function and use it as the input for another.

Mathematically, given two functions f and g, their composition is represented as f(g(x)). Here, g(x) is computed first, and its result is passed to f.



Compose and pipe

A group of numbers and symbols

Description automatically generated

A math formula with colorful letters and numbers

Description automatically generated

Pipe -> left to right

Compose -> right -> left

Throttling and Debouncing

**Throttling:** In throttling, the function is executed at a fixed interval. Even if the triggering event occurs more frequently, the function is invoked according to the defined interval.

**Debouncing:** In debouncing, the function is only executed after a specific delay since the last event's occurrence. If new events occur within the delay period, the timer is reset, and the function execution is further delayed.

Use case

**Throttling:** Throttling is suitable for scenarios where you want to limit the frequency of function calls, like handling scroll events or resizing events. It helps avoid overloading the system with frequent updates.

**Debouncing:** Debouncing is ideal when you want to wait for a pause in the events before triggering a function. This is useful for situations like search suggestions, where you want to wait for the user to finish typing before fetching suggestions.

### Implementation

**Throttling:** Throttling typically involves setting a fixed interval between function calls using timers or timestamps to track the last invocation time.

**Debouncing:** Debouncing involves starting a timer when an event occurs and resetting the timer whenever a new event occurs within the delay period. The function is executed when the timer expires after the last event.