Server-side

Node.JS

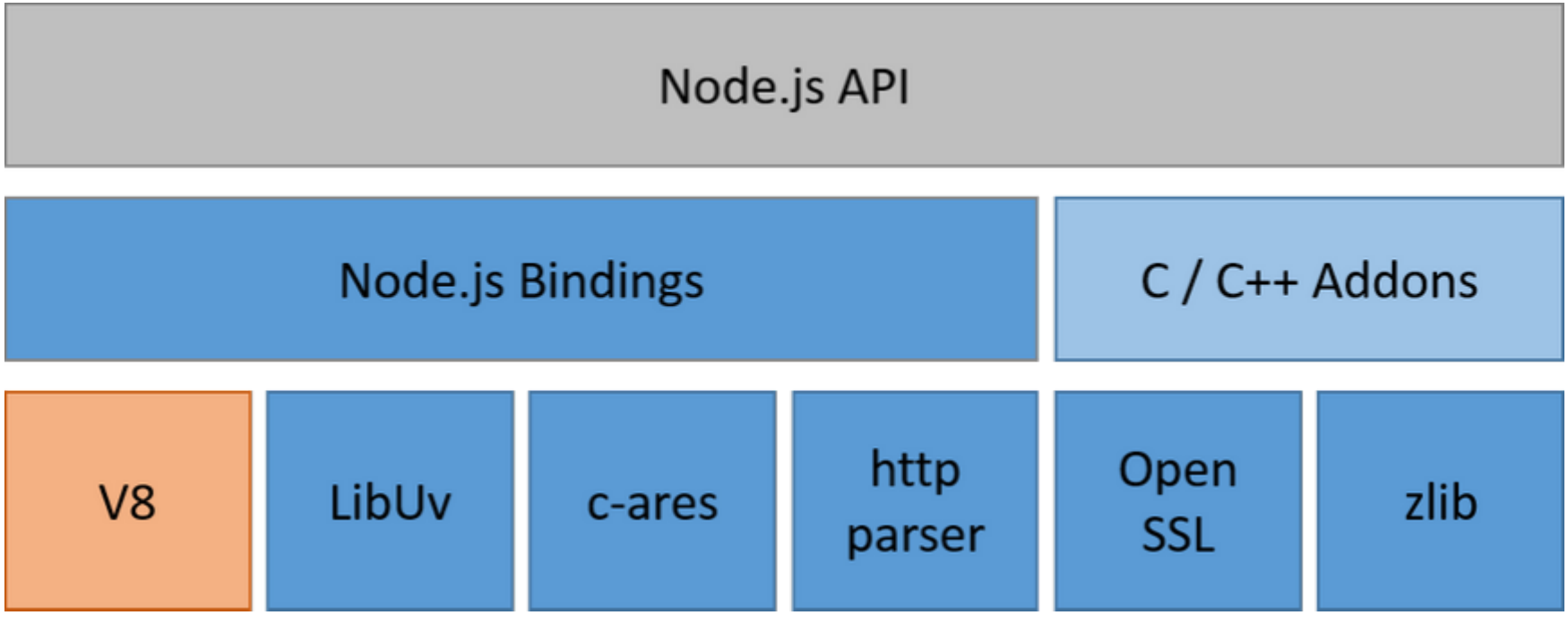
17 Oct 2019 - Hafiz Joundy Syafie - Devtalks

NODE.JS

What is Node.JS ?

Node.js is an open-source server side runtime environment built on Chrome's **V8 JavaScript engine**. It provides an event driven, non-blocking (asynchronous) I/O and **cross-platform runtime** environment for **building highly scalable server-side application** using JavaScript.

Node.JS architecture



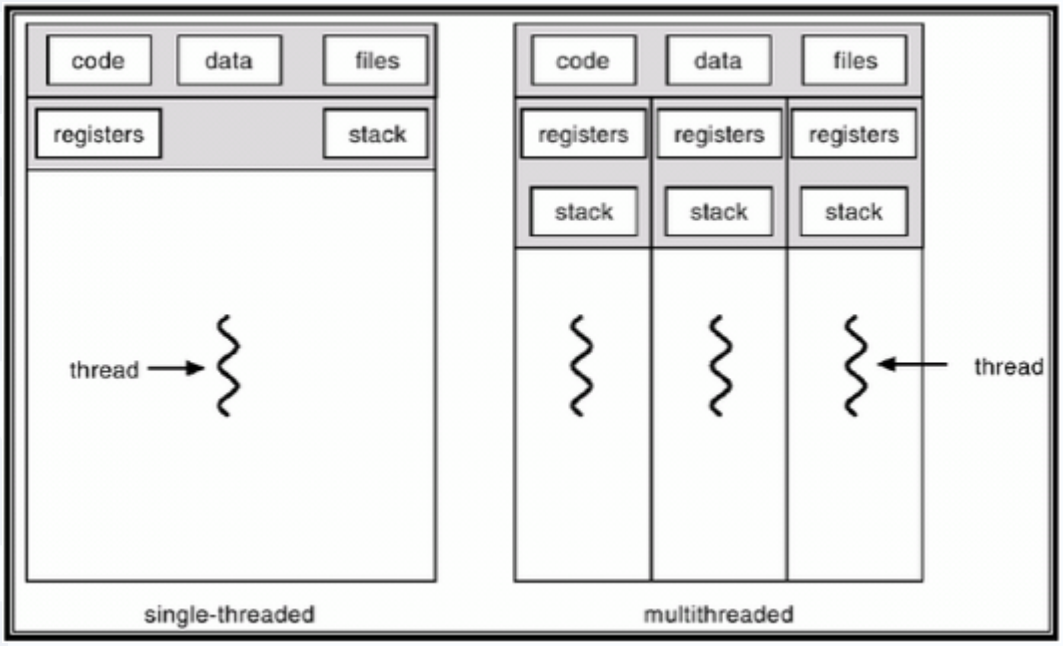
WHY NODEJS?

- Good for beginner developers, JavaScript is simple to learn, rich framework (example: **Express**, Koa, Hapi, etc).
- It is fast, due to Google innovative technologies and **the event loop.**
- Multiple modules (**NPM**, Grunt, etc.) and supportive community.
- Wide range of hosting options.
- JS is the longest running language, 95% of developers know some of it.

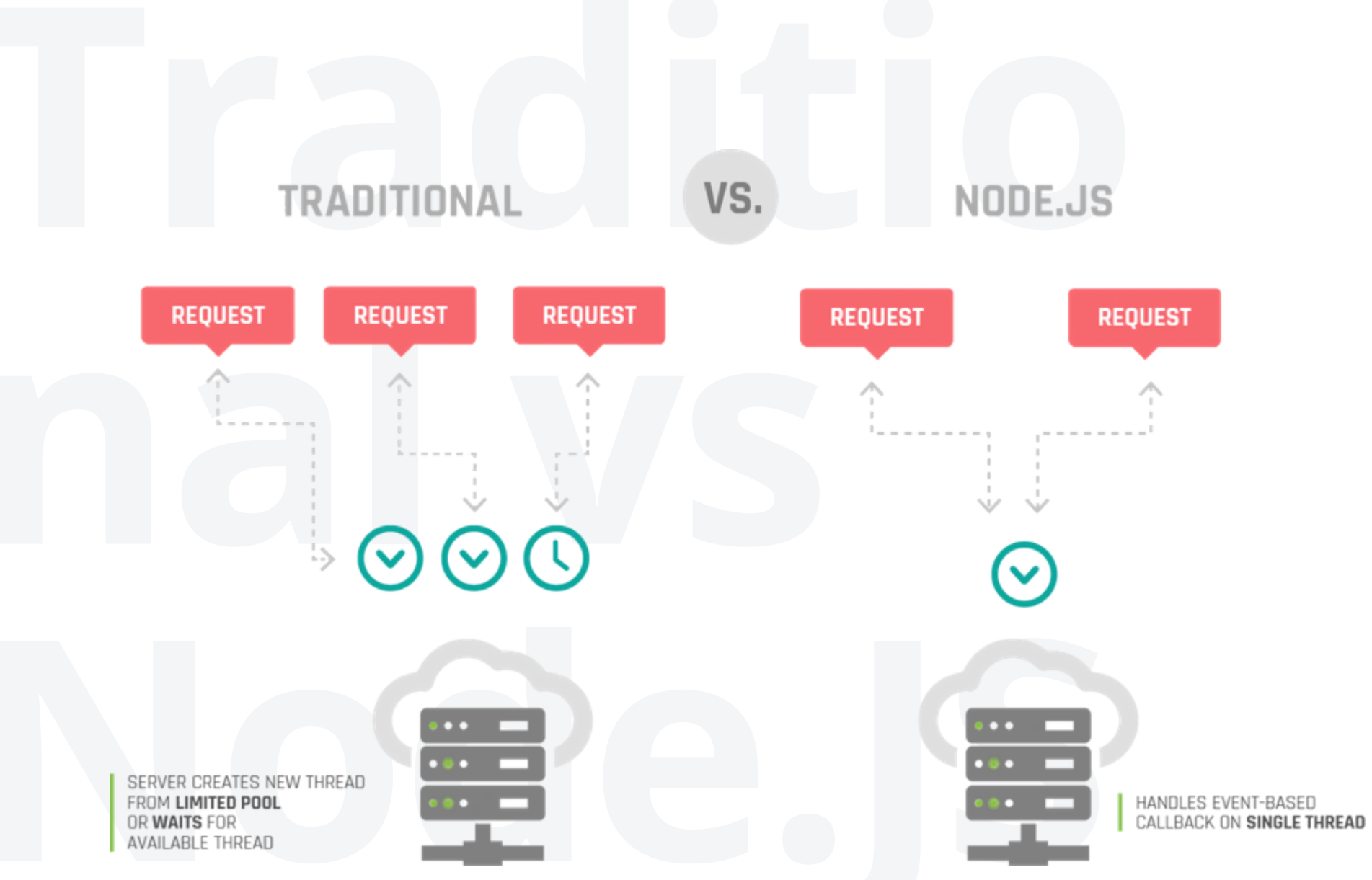
Javascript is a **single threaded** application

Thread

WHAT IS THREAD?



A THREAD IS A SMALL SEQUENCE OF PROGRAMMED INSTRUCTIONS. THREADS REFER TO THE **HIGHEST LEVEL OF CODE** YOUR PROCESSOR CAN EXECUTE.



Traditional vs Node.JS

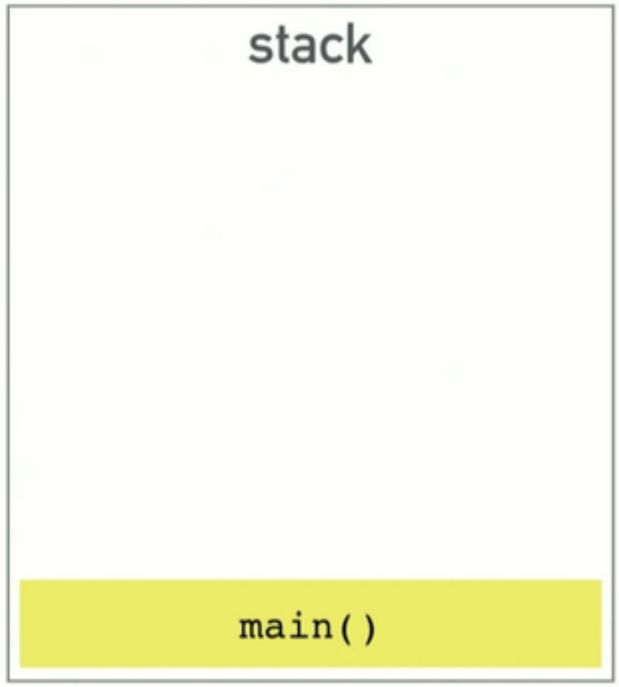
The Call Stack

because javascript run in a single thread, so javascript **only execute 1 tasks in one time.**

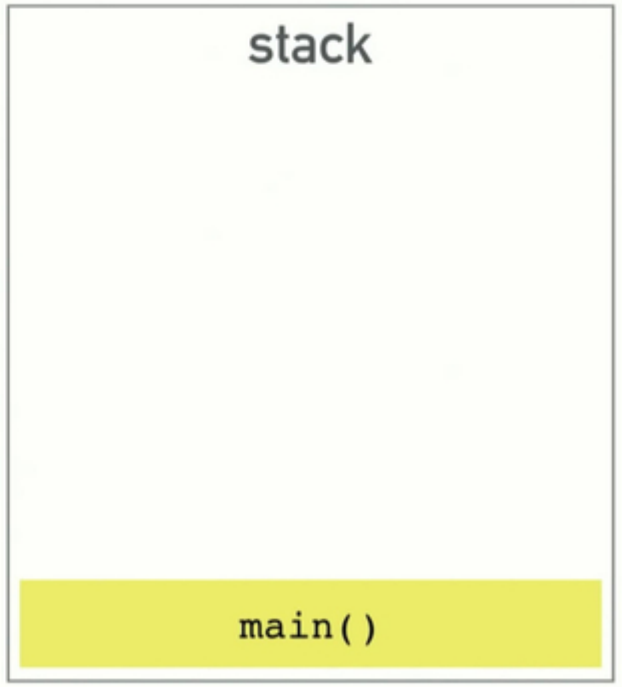
```
function multiply(a, b) {
    return a * b;
}

function square(n) {
    return multiply(n, n);
}

function printSquare(n) {
    var squared = square(n);
    console.log(squared);
}

printSquare(4);
```

```
function multiply(a, b) {
    return a * b;
}

function square(n) {
    return multiply(n, n);
}

function printSquare(n) {
    var squared = square(n);
    console.log(squared);
}

printSquare(4);
```

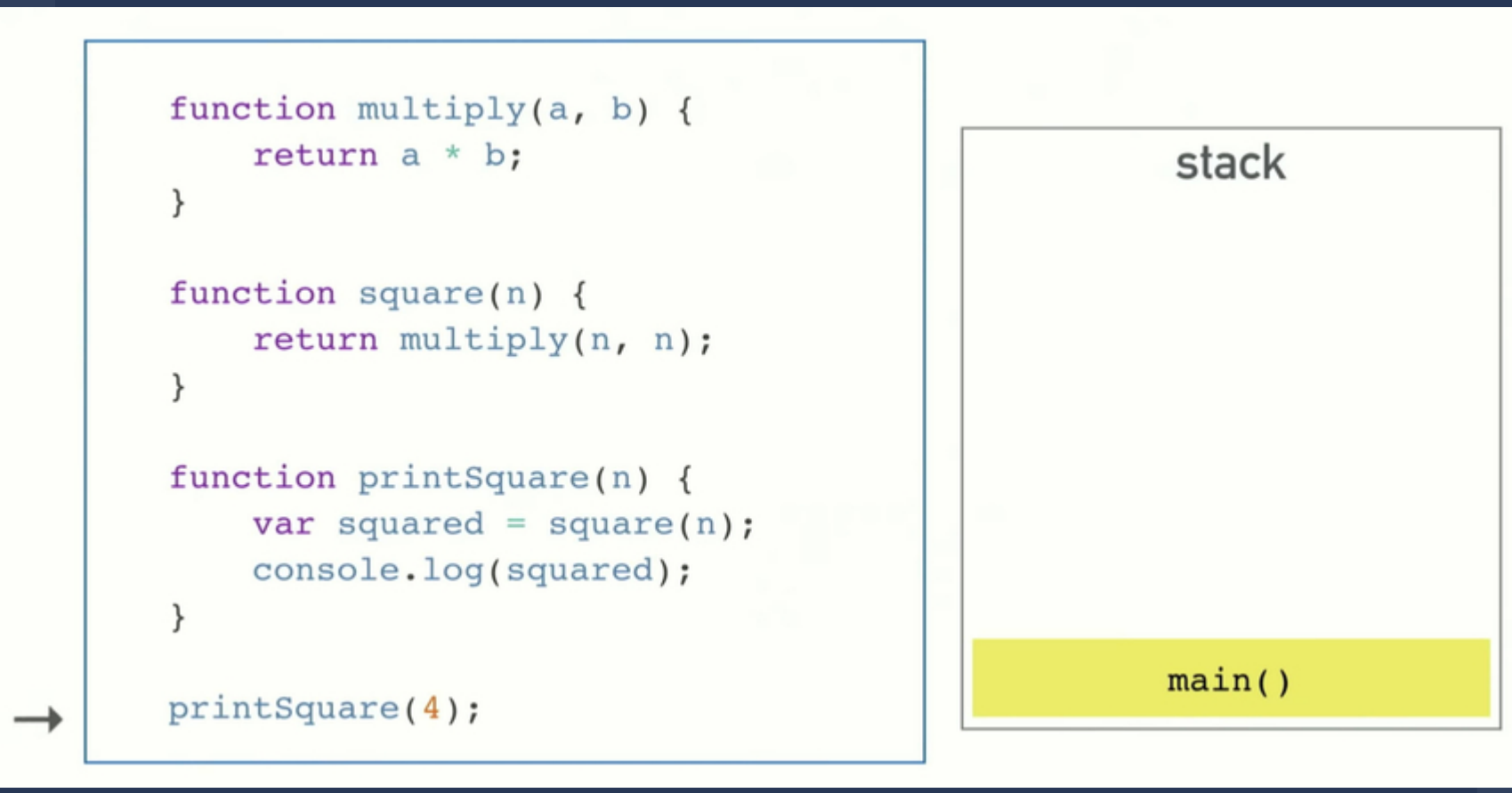
```
function multiply(a, b) {
    return a * b;
}

function square(n) {
    return multiply(n, n);
}

function printSquare(n) {
    var squared = square(n);
    console.log(squared);
}

printSquare(4);
```

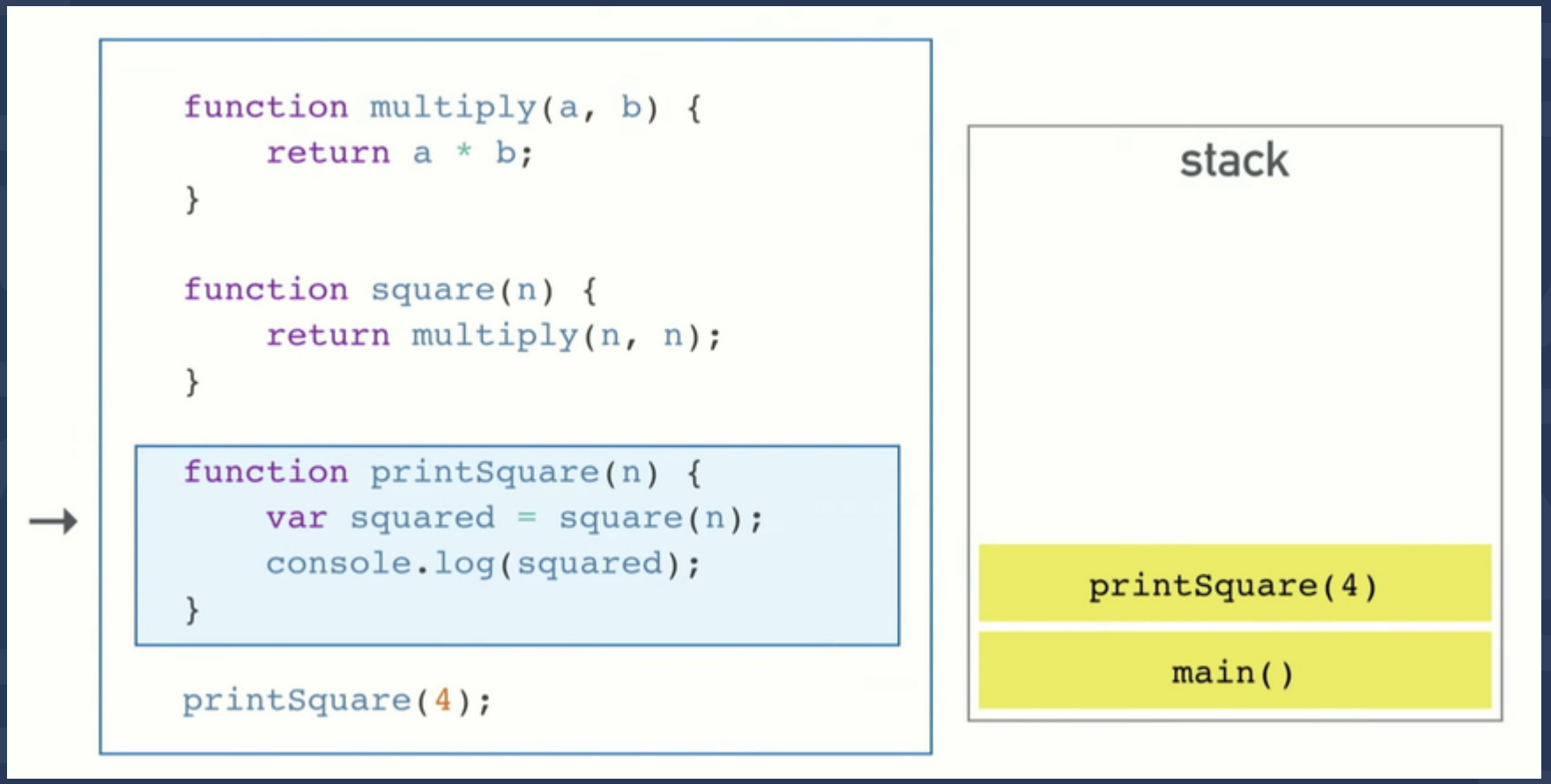


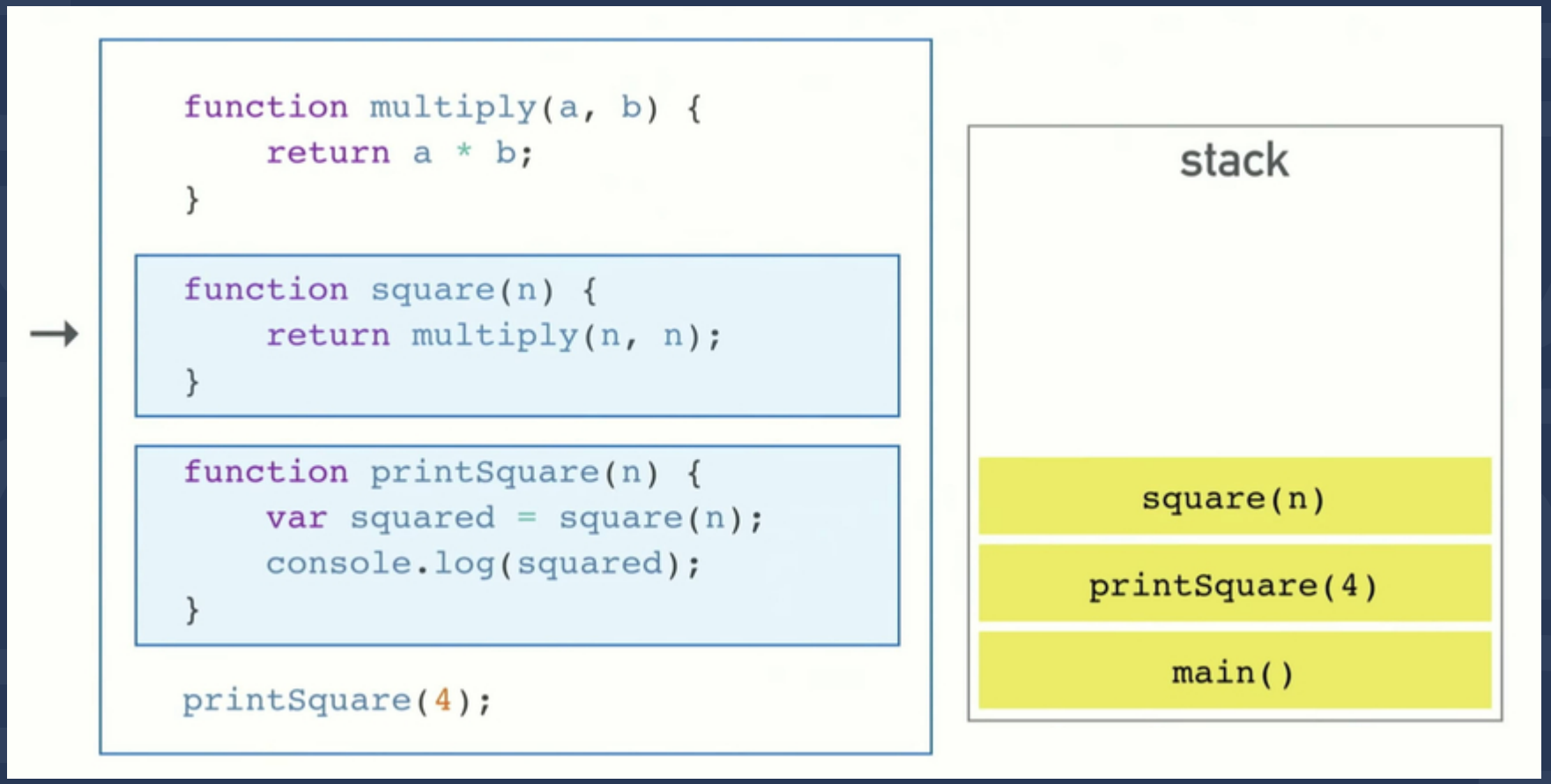
```
function multiply(a, b) {
    return a * b;
}

function square(n) {
    return multiply(n, n);
}

function printSquare(n) {
    var squared = square(n);
    console.log(squared);
}

printSquare(4);
```

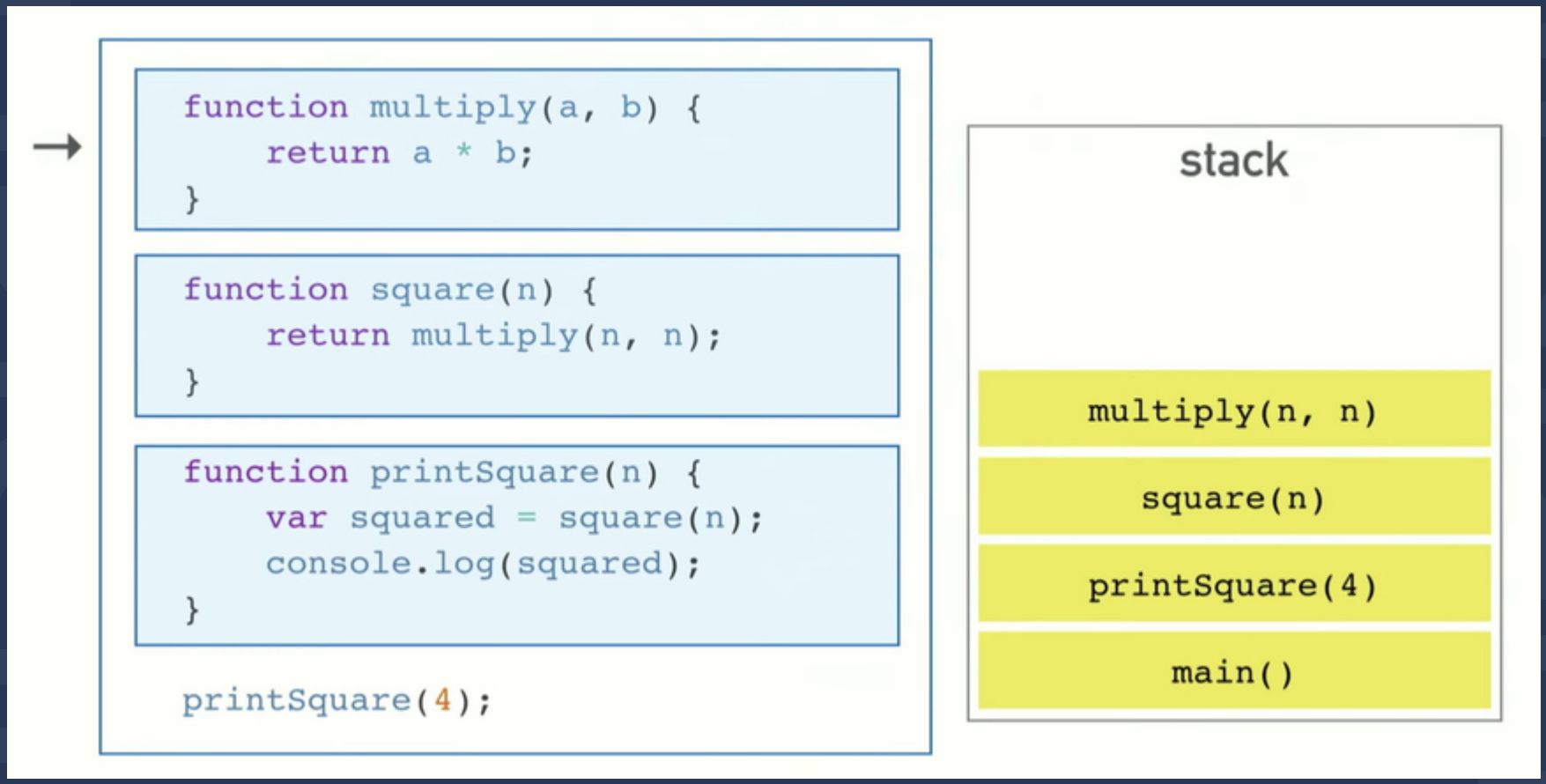


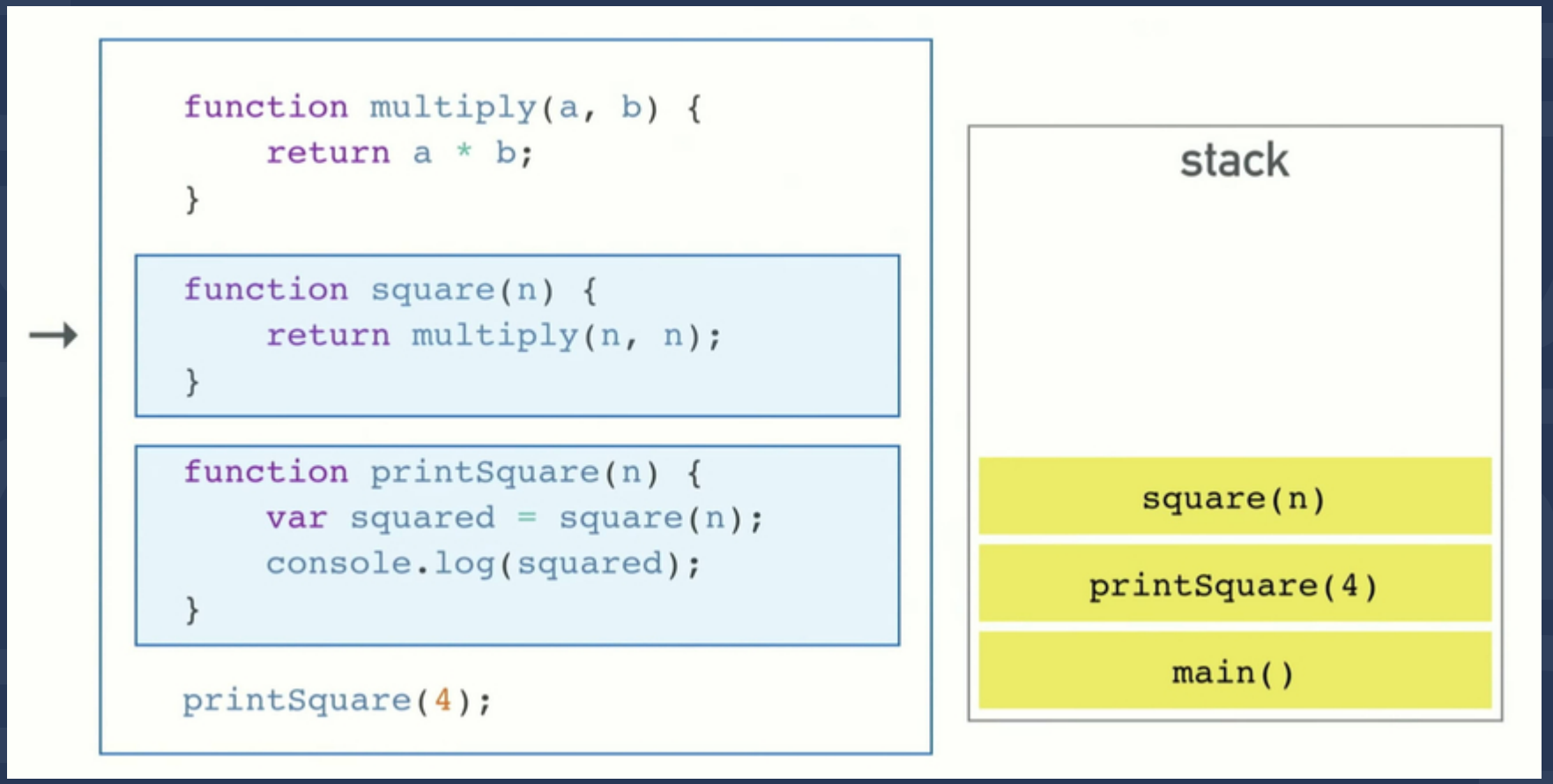
```
function multiply(a, b) {
    return a * b;
}

function square(n) {
    return multiply(n, n);
}

function printSquare(n) {
    var squared = square(n);
    console.log(squared);
}

printSquare(4);
```

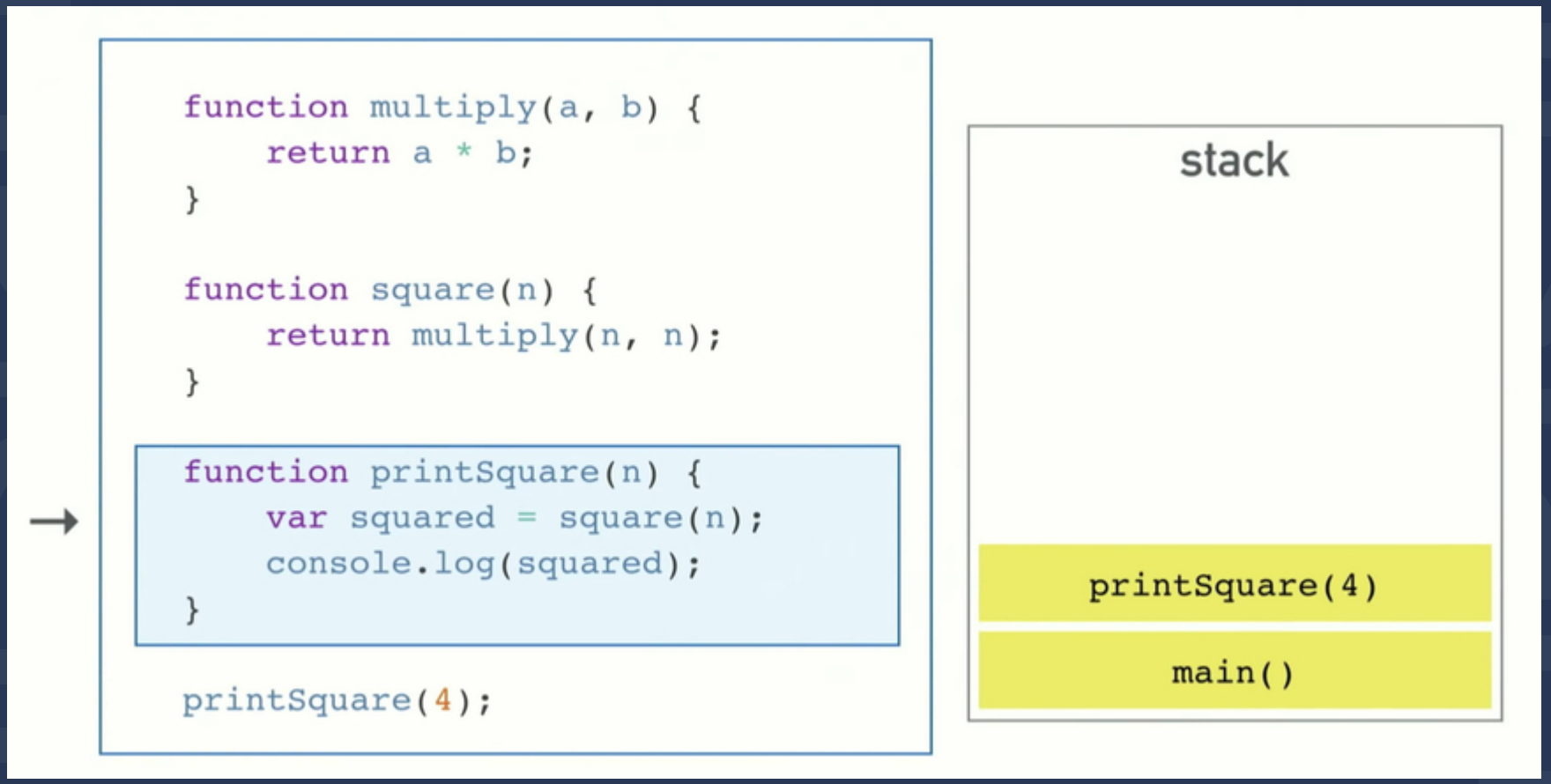


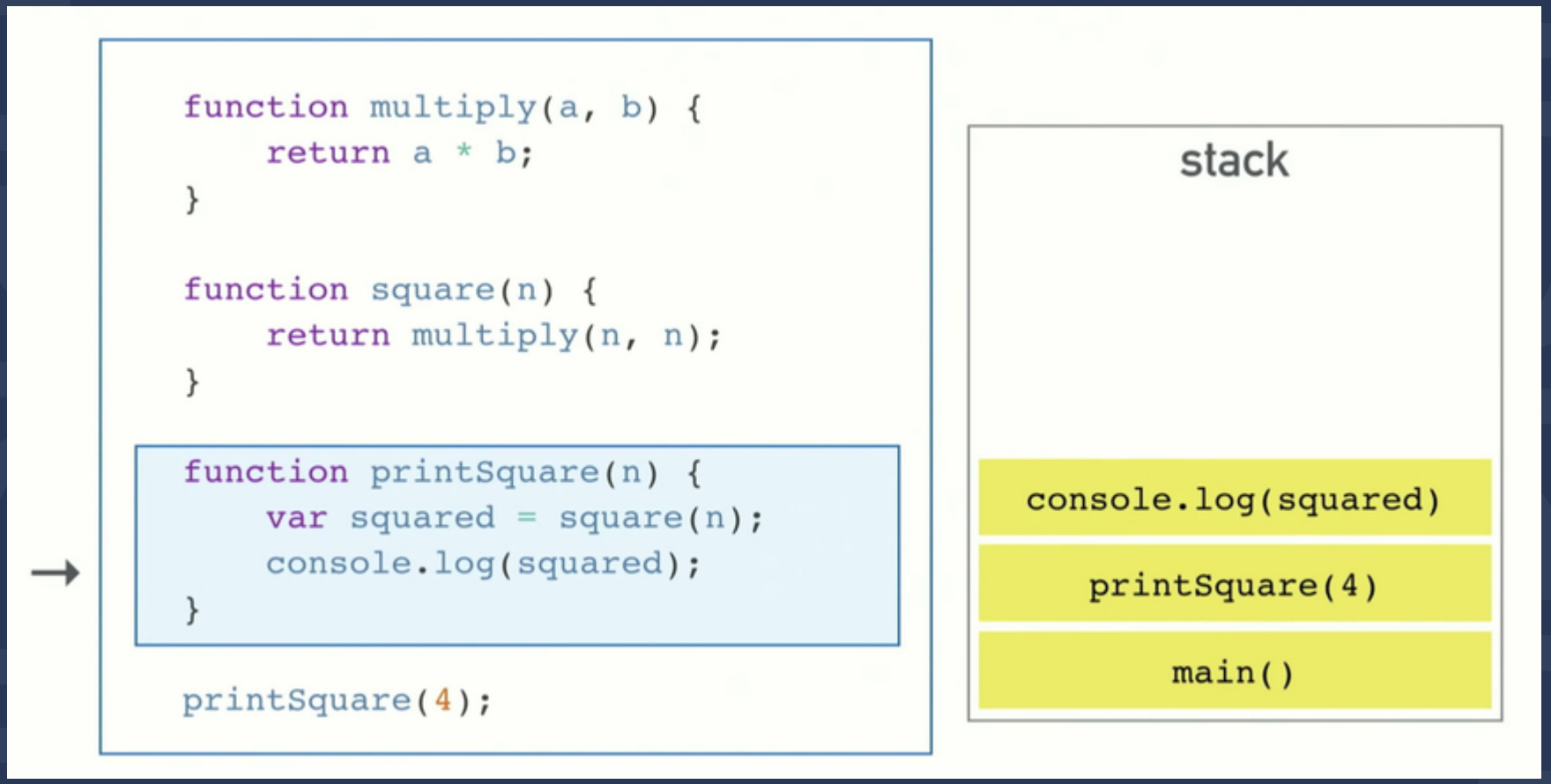

function multiply(a, b) {
return a * b;
}
function square(n) {
return multiply(n, n);
}
function printSquare(n) {
var squared = square(n);
console.log(squared);
}
printSquare(4);
stack
printSquare(4)
main()


function multiply(a, b) {
return a * b;
}
function square(n) {
return multiply(n, n);
}
function printSquare(n) {
var squared = square(n);
console.log(squared);
}
printSquare(4);
stack
square(n)
printSquare(4)
main()

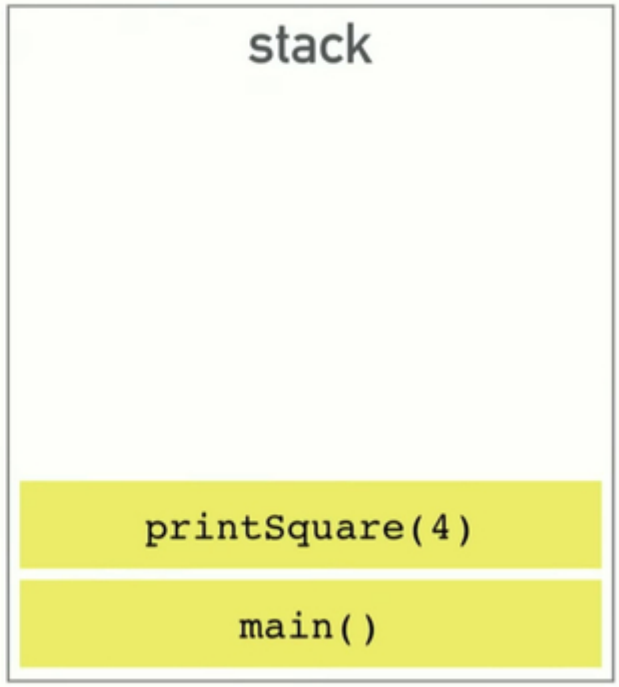

function multiply(a, b) {
return a * b;
}
function square(n) {
return multiply(n, n);
}
function printSquare(n) {
var squared = square(n);
console.log(squared);
}
printSquare(4);
stack
multiply(n, n)
square(n)
printSquare(4)
main()


function multiply(a, b) {
return a * b;
}
function square(n) {
return multiply(n, n);
}
function printSquare(n) {
var squared = square(n);
console.log(squared);
}
printSquare(4);
stack
square(n)
printSquare(4)
main()

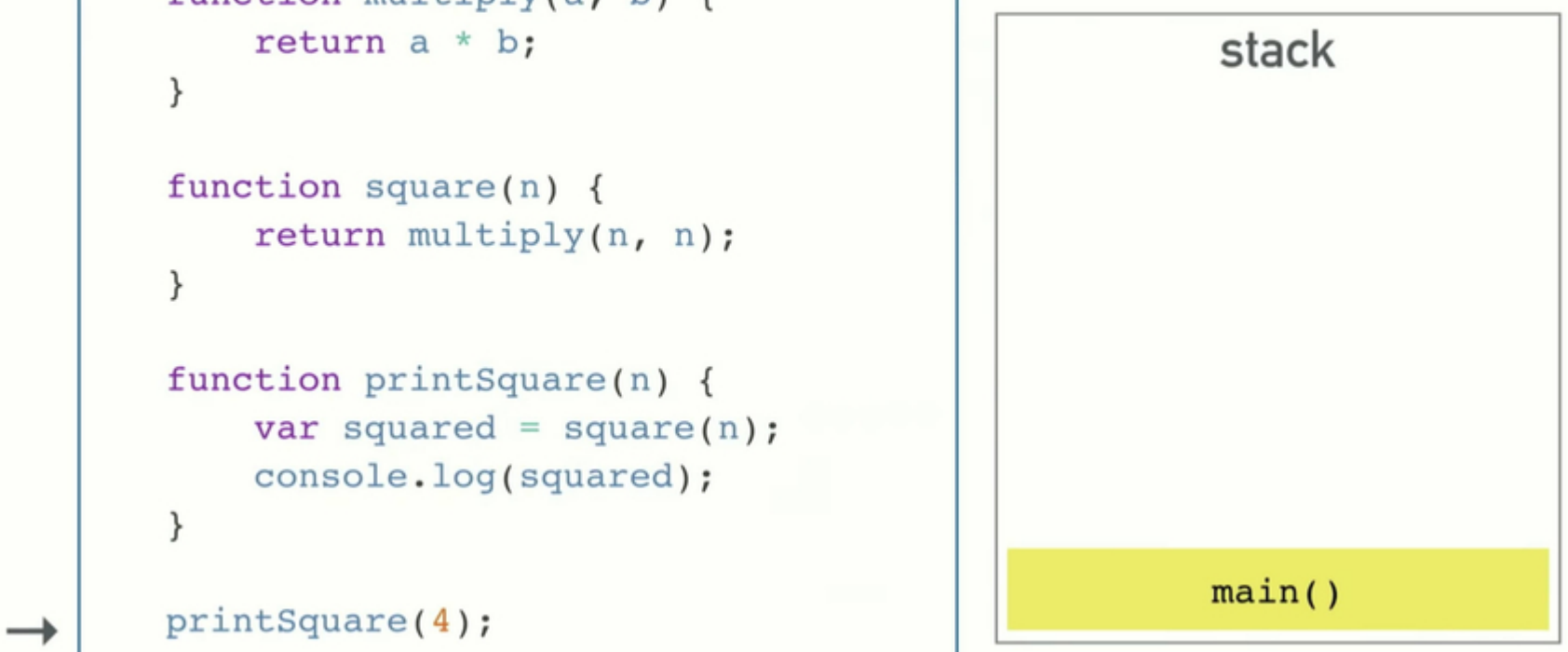

function multiply(a, b) {
return a * b;
}
function square(n) {
return multiply(n, n);
}
function printSquare(n) {
var squared = square(n);
console.log(squared);
}
printSquare(4);
stack
printSquare(4)
main()


function multiply(a, b) {
return a * b;
}
function square(n) {
return multiply(n, n);
}
function printSquare(n) {
var squared = square(n);
console.log(squared);
}
printSquare(4);
stack
console.log(squared)
printSquare(4)
main()

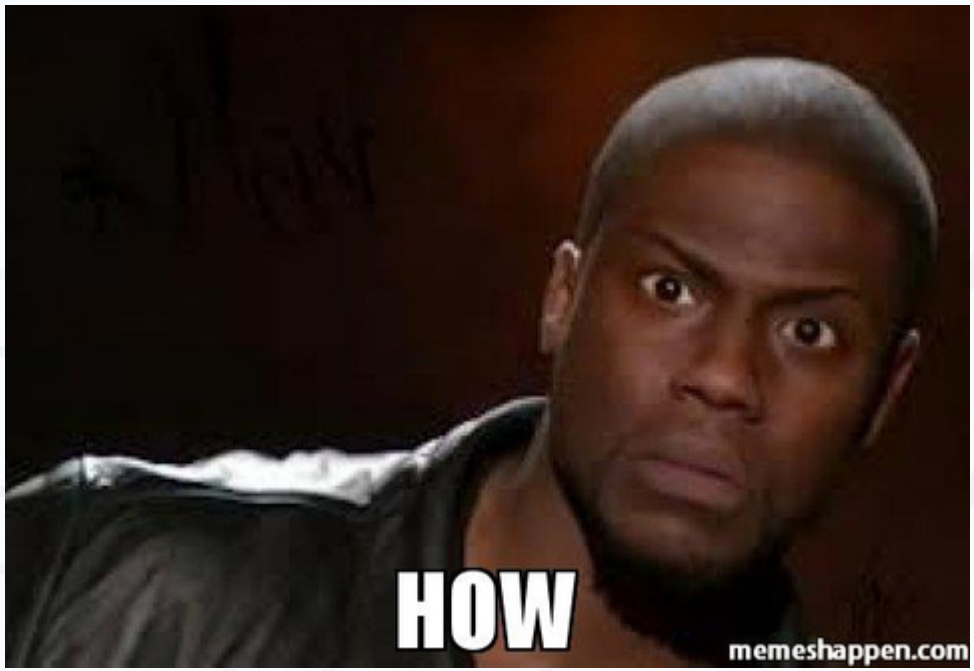
```
function multiply(a, b) {
    return a * b;
}

function square(n) {
    return multiply(n, n);
}

function printSquare(n) {
    var squared = square(n);
    console.log(squared);
}

printSquare(4);
```



```
function multiply(a, b) {
    return a * b;
}

function square(n) {
    return multiply(n, n);
}

function printSquare(n) {
    var squared = square(n);
    console.log(squared);
}

printSquare(4);
```

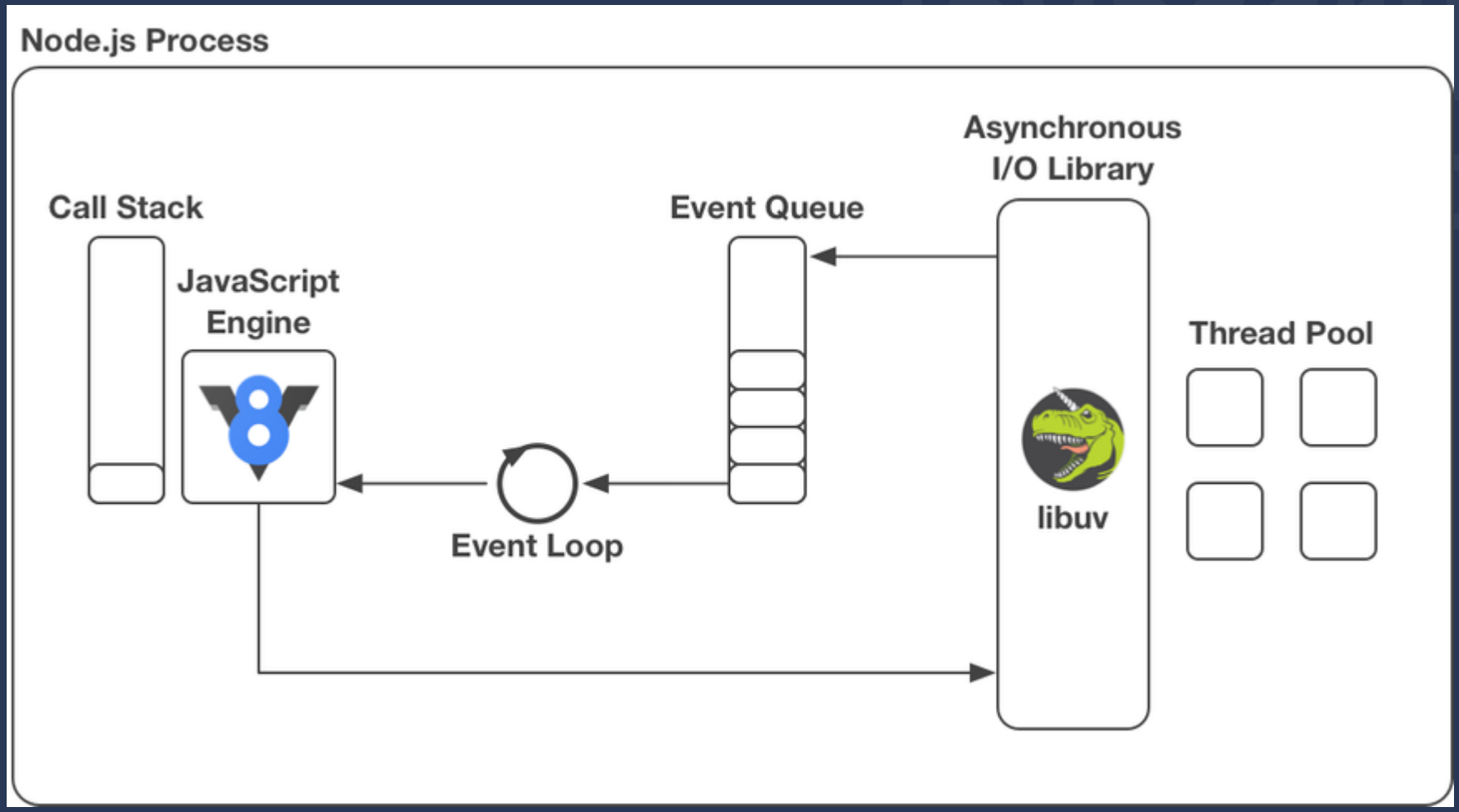


```
function multiply(a, b) {
    return a * b;
}

function square(n) {
    return multiply(n, n);
}

function printSquare(n) {
    var squared = square(n);
    console.log(squared);
}

printSquare(4);
```

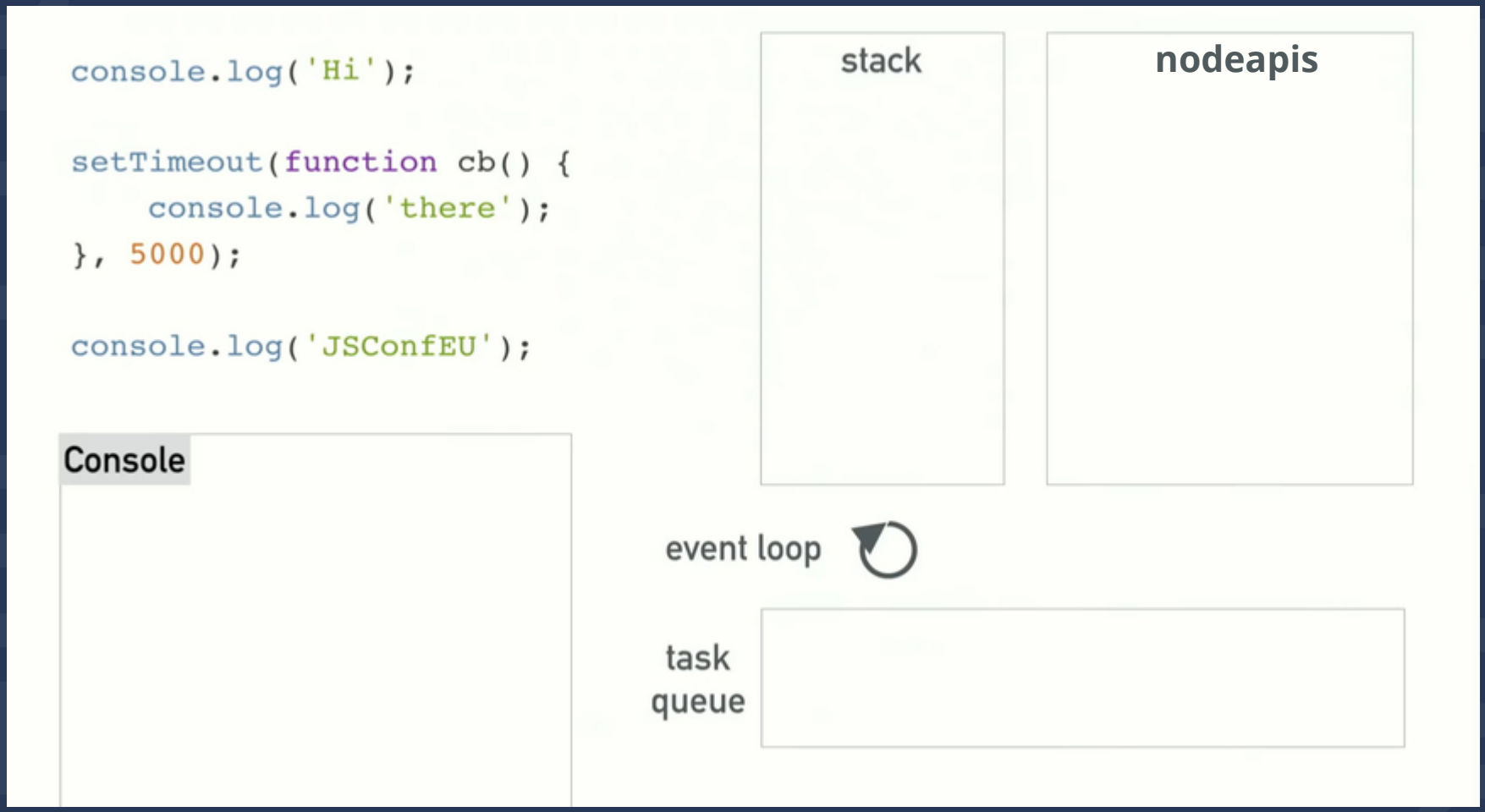


Non-Blocking I/O in Javascript ?

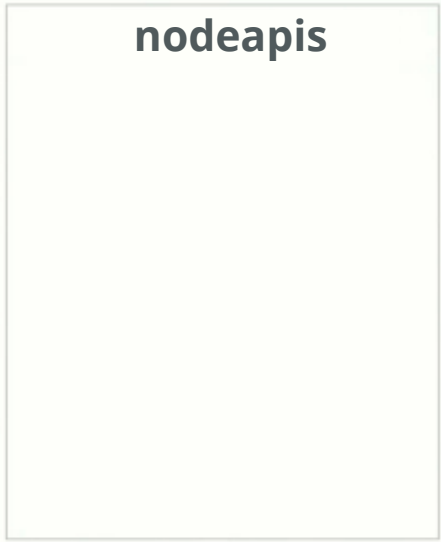
The Event Loop

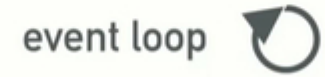


```
console.log('Hi');

setTimeout(function cb() {
    console.log('there');
}, 5000);

console.log('JSConfEU');
```

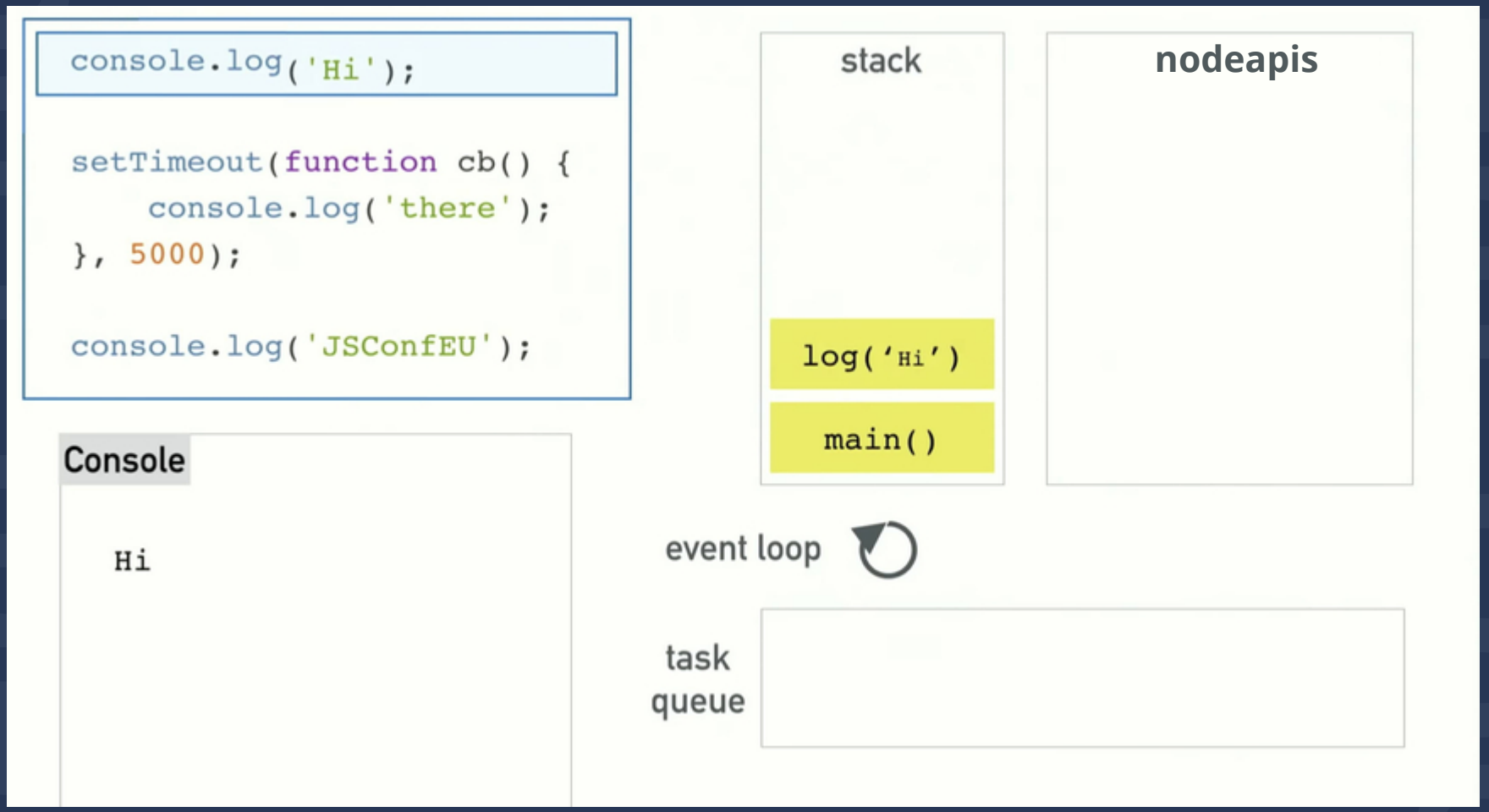


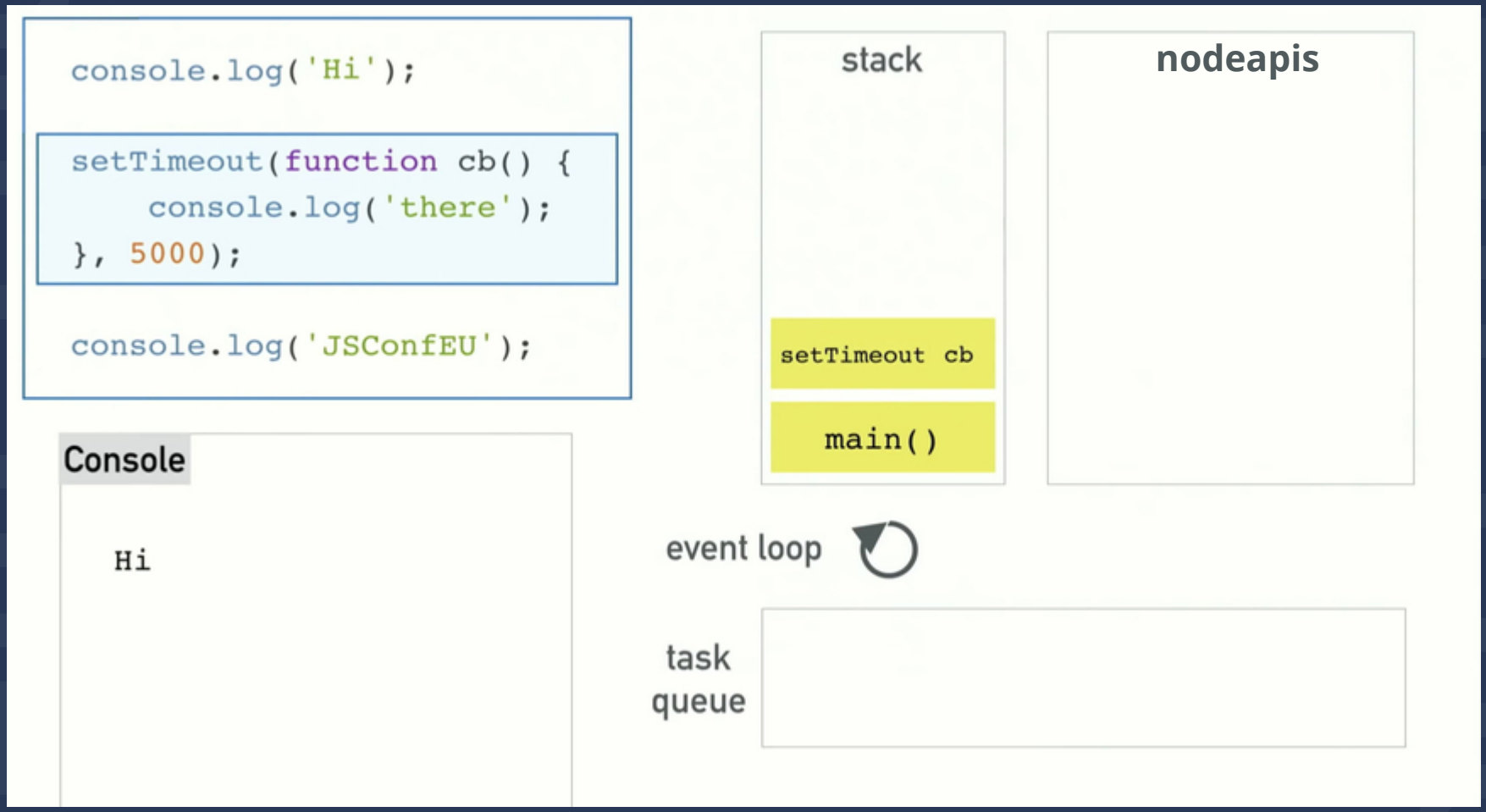
```
console.log('Hi');

setTimeout(function cb() {
    console.log('there');
}, 5000);

console.log('JSConfEU');
```



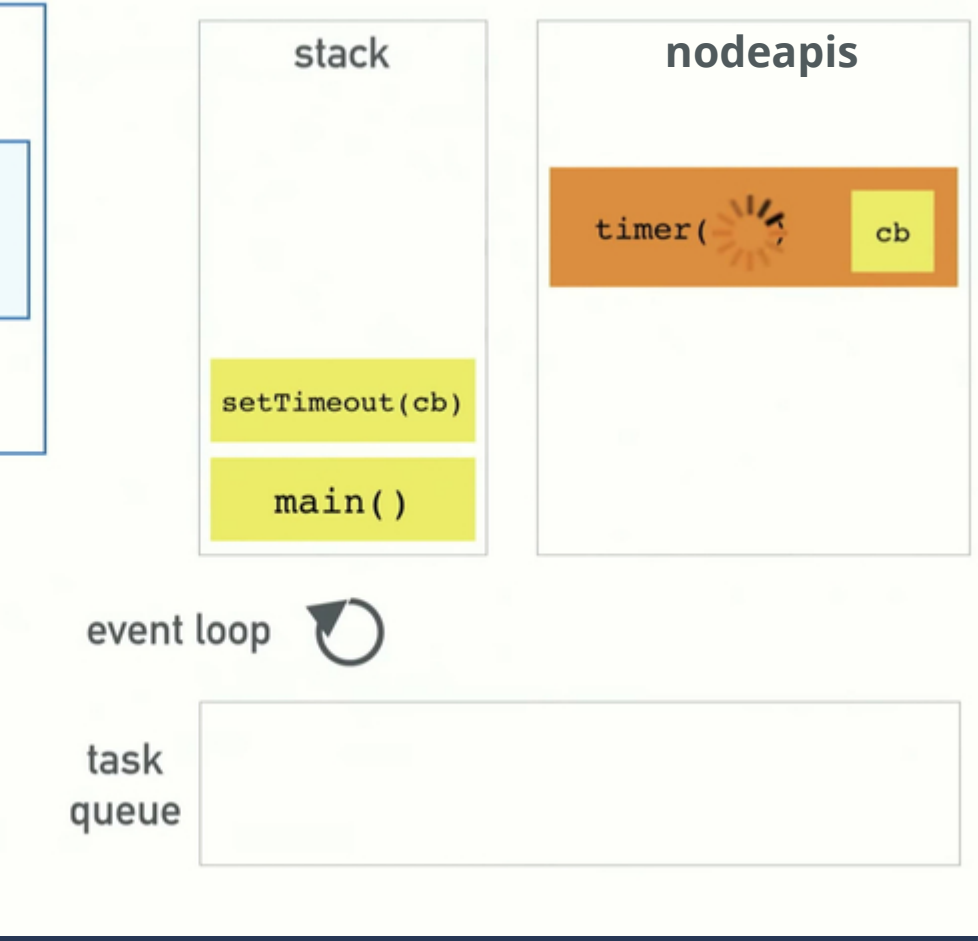


```
console.log('Hi');

setTimeout(function cb() {
    console.log('there');
}, 5000);

console.log('JSConfEU');
```




console.log('Hi');
setTimeout(function cb() {
console.log('there');
}, 5000);
console.log('JSConfEU');
Console
Hi
stack
setTimeout cb
main()
nodeapis
event loop
task queue

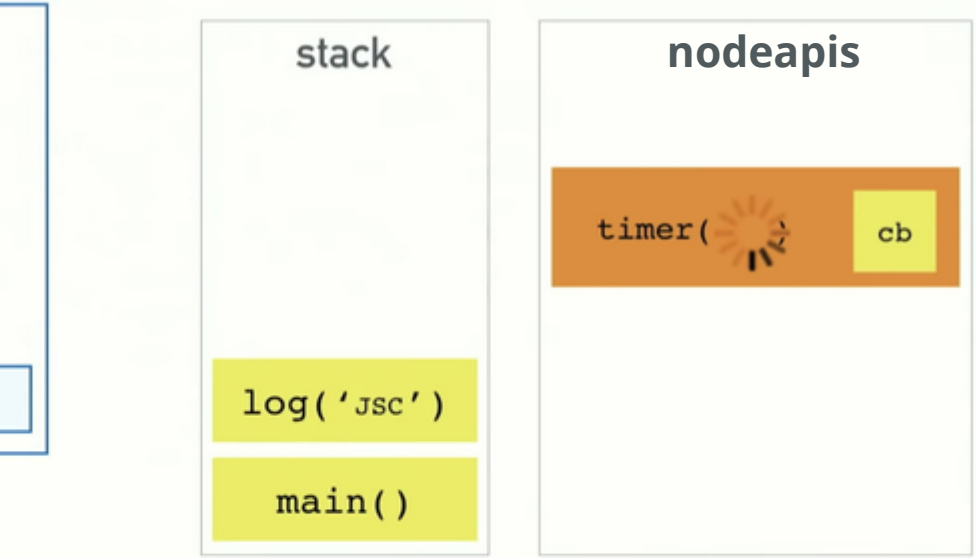
```
console.log('Hi');

setTimeout(function cb() {
    console.log('there');
}, 5000);

console.log('JSConfEU');
```



Console

```
Hi
```

```
console.log('Hi');

setTimeout(function cb() {
    console.log('there');
}, 5000);

console.log('JSConfEU');
```

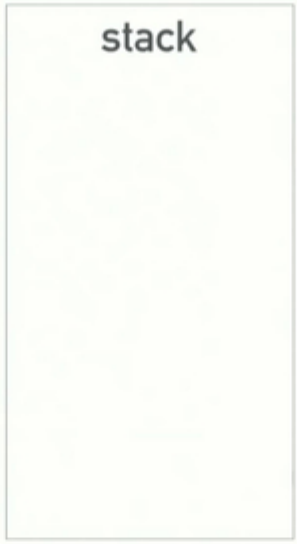


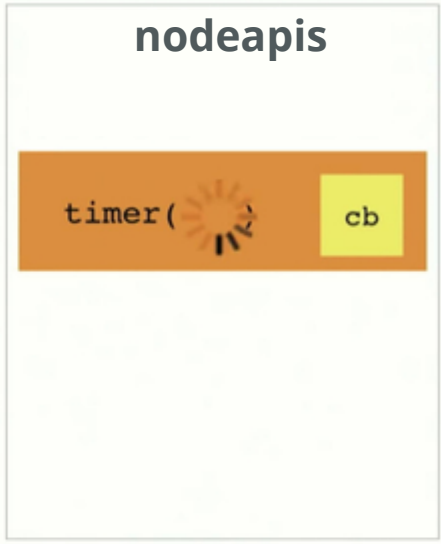
Console

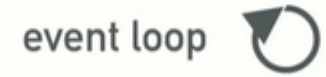
```
Hi

JSConfEU
```

Event Loop

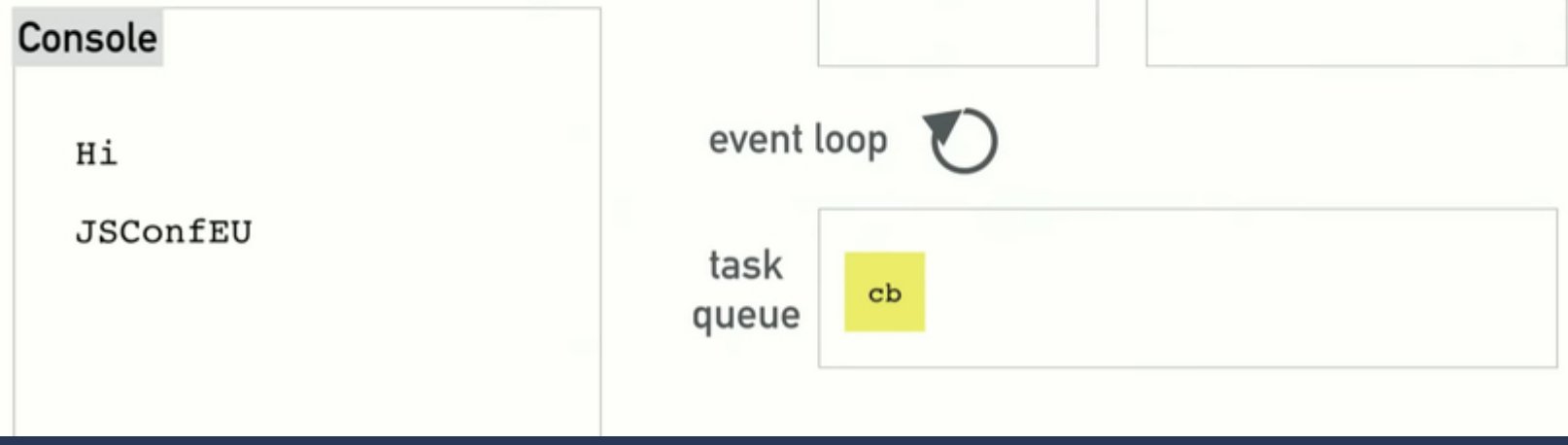
```
console.log('Hi');

setTimeout(function cb() {
    console.log('there');
}, 5000);

console.log('JSConfEU');
```



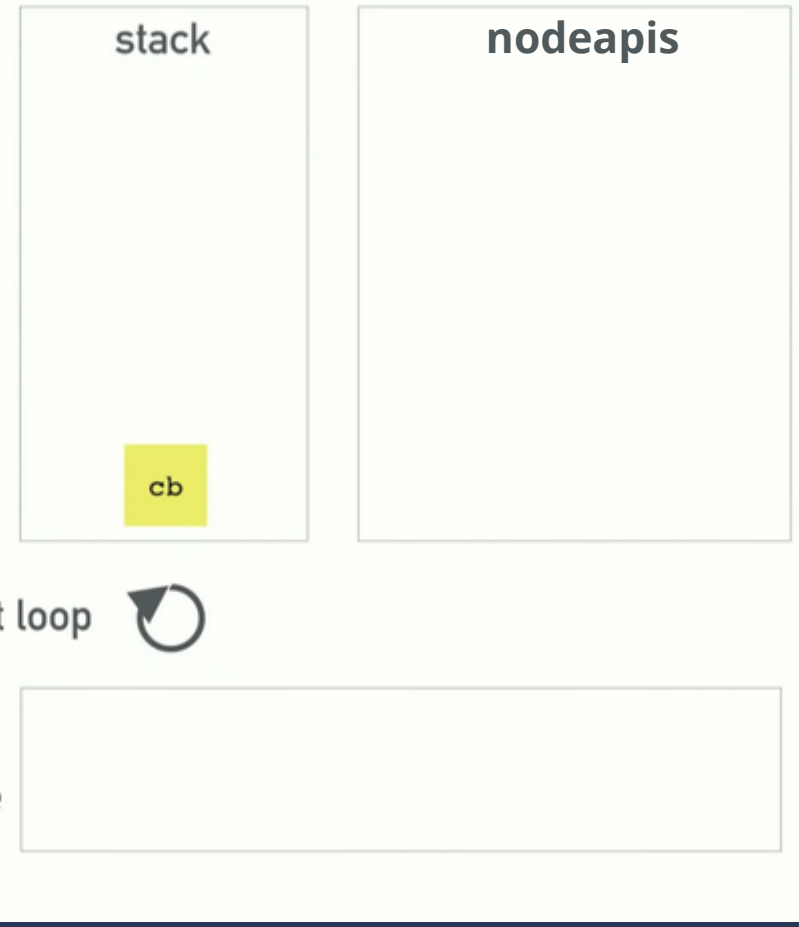




```
console.log('Hi');

setTimeout(function cb() {
    console.log('there');
}, 5000);

console.log('JSConfEU');
```



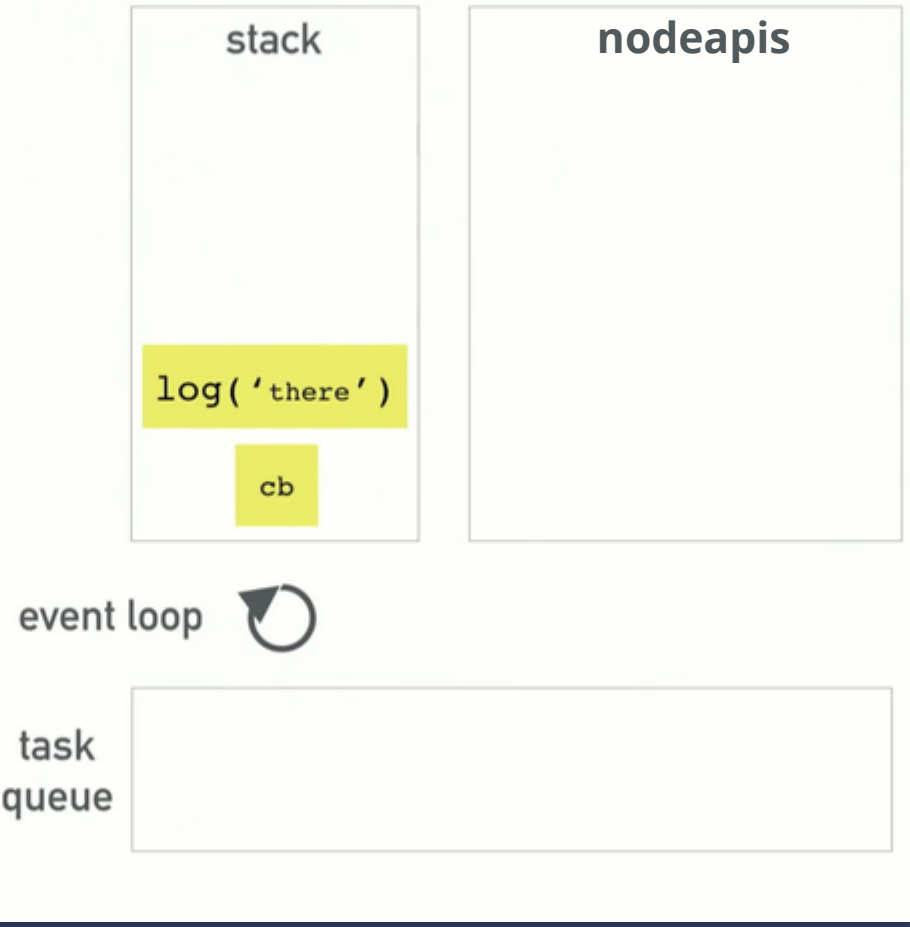
```
console.log('Hi');

setTimeout(function cb() {
    console.log('there');
}, 5000);

console.log('JSConfEU');
```



Console

```
Hi

JSConfEU
```

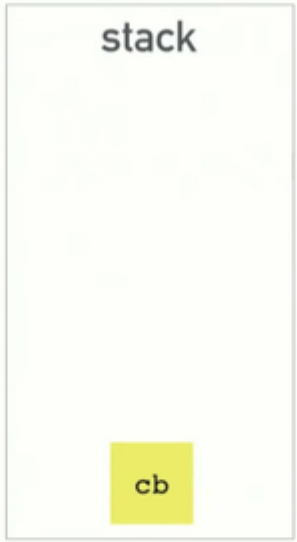
```
console.log('Hi');

setTimeout(function cb() {
    console.log('there');
}, 5000);

console.log('JSConfEU');
```

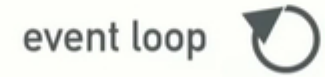


Console

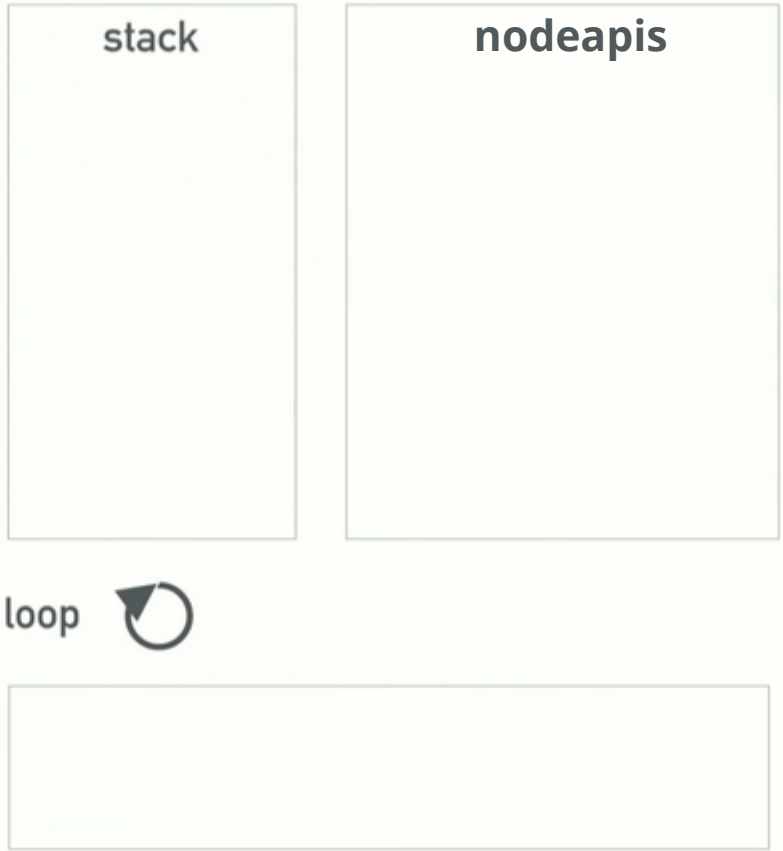
```
Hi
JSConfEU
there
```

```
console.log('Hi');

setTimeout(function cb() {
    console.log('there');
}, 5000);

console.log('JSConfEU');
```





```
console.log('Hi');

setTimeout(function cb() {
    console.log('there');
}, 5000);

console.log('JSConfEU');
```

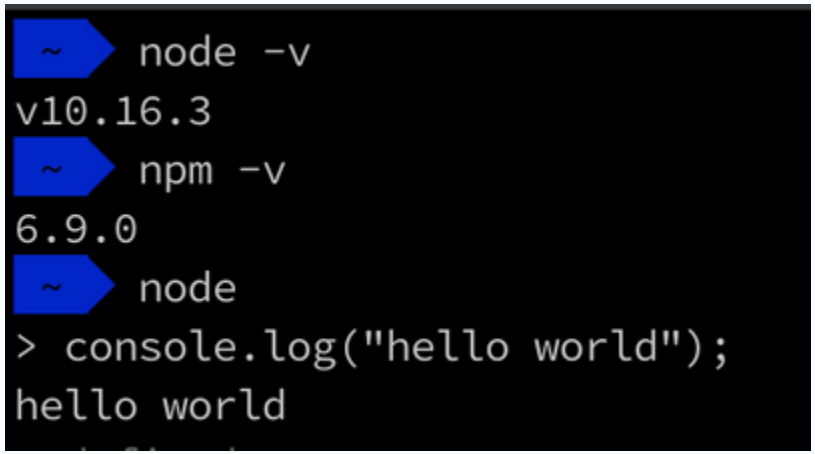


How to Install Node.JS

Download : https://nodejs.org/

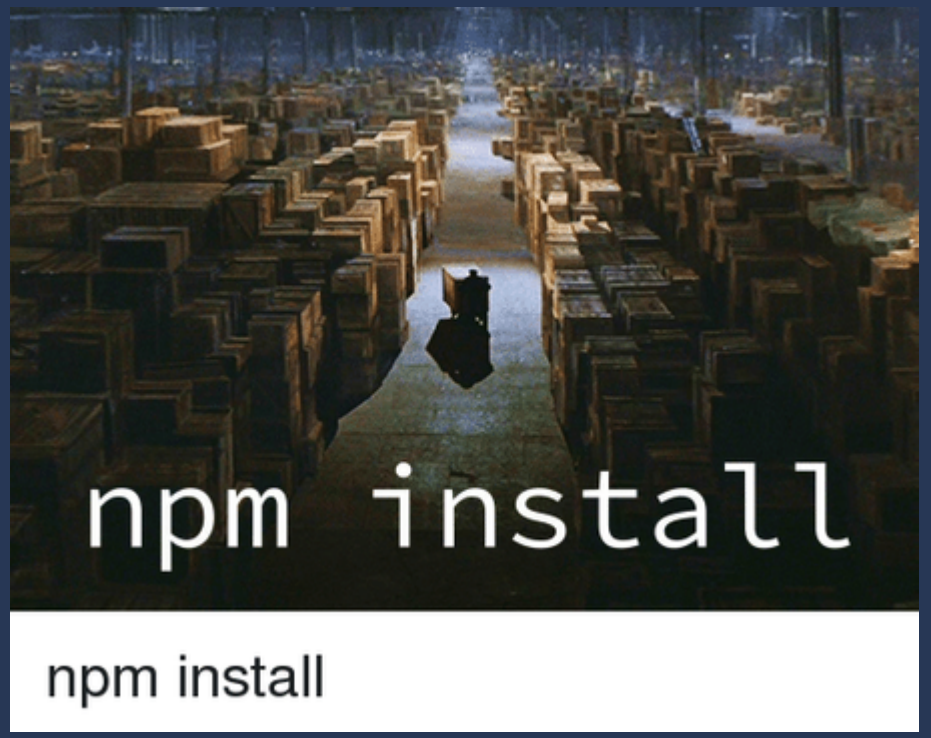
Test It!

- Test **node**, type **node -v**
- Test **NPM**, type **npm -v**



NPM

- **NPM** is a package manager for Node.js packages, or modules if you like.
- **npmjs.com** hosts thousands of free packages to download and use.
- The **NPM** program is installed on your computer when you install Node.js



Basic NPM Command

npm init	: Creating package.json
npm install	: Installing all packages in package.json
npm install <package>	: Installing package localy
npm install <package> --save-dev	: Installing pakage localy development
npm uninstall <package>	: Uninstall package

Package.json

Package.json holds various metadata relevant to the project. This file is used to give information to npm that allows it to identify the project as well as handle the project's dependencies.

Live Coding

Thank You :)

Reference

- https://www.youtube.com/watch?v=8aGhZQkoFbQ
- https://stackoverflow.com/questions/36766696/which-is-correct-node-js-architecture
- http://thewebstop.blogspot.com/2017/09/why-nodejs-is-asynchronous.html
- https://medium.com/beginners-guide-to-mobile-web-development/introduction-to-npm-and-basic-npm-commands-18aa16f69f6b
- https://thinkmobiles.com/blog/why-use-nodejs/
- https://nodejs.org/en/knowledge/getting-started/npm/what-is-the-file-package-json/
- https://nodejs.org/en/
- https://www.npmjs.com/