Node JS
Concepts & APIs

Agenda

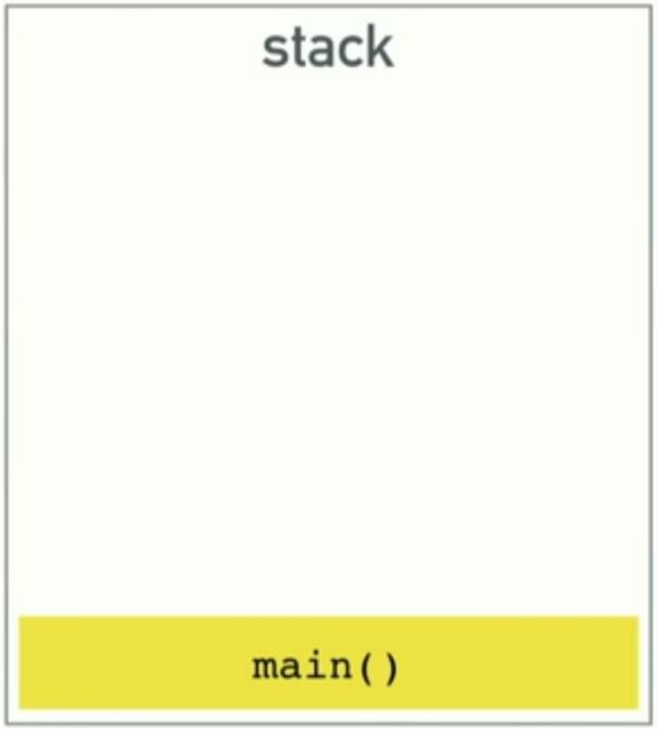
Event Loop
Events
FileSystem
Buffers
Webserver

Event Loop

JS is single-threaded
Uses heap and stack
Callback Queue, WebAPI and Event
Buffers
Webserver

Call Stack

```
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);
```



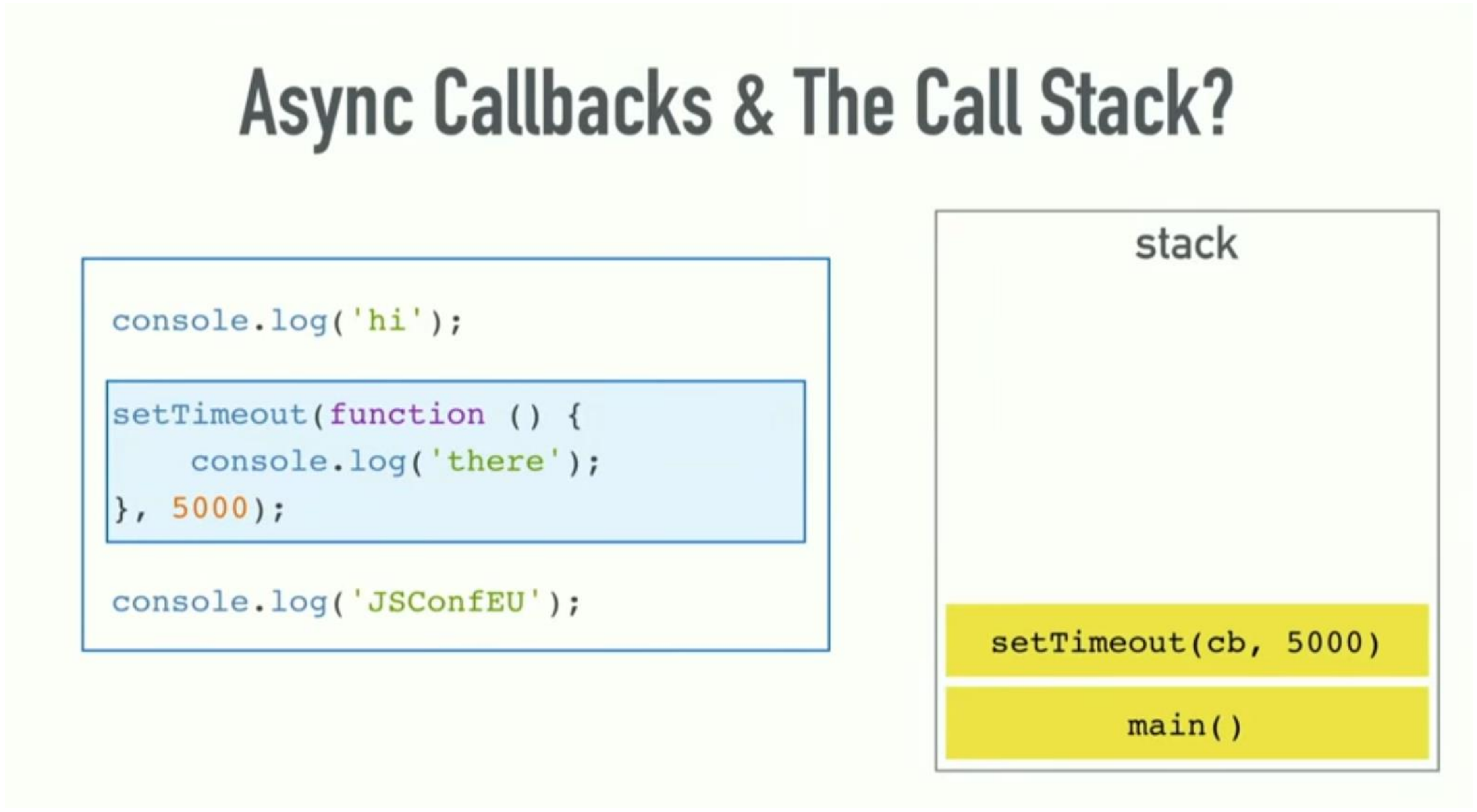
Code Execution

Synchronous execution (Blocking)
Asynchronous execution

Problem on browser – stalling!

Solution!

Asynchronous execution

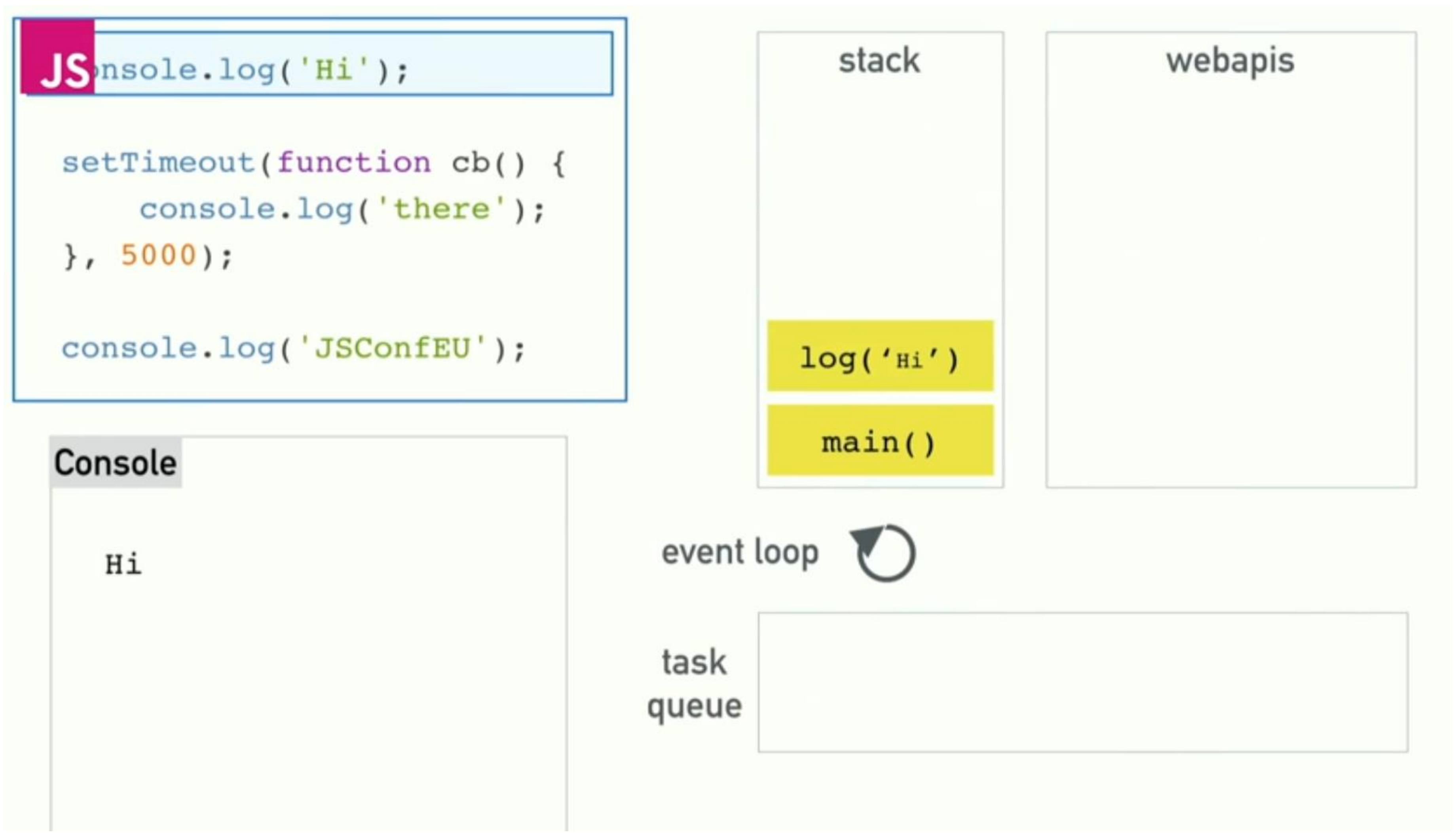


Async Callbacks & The Call Stack?

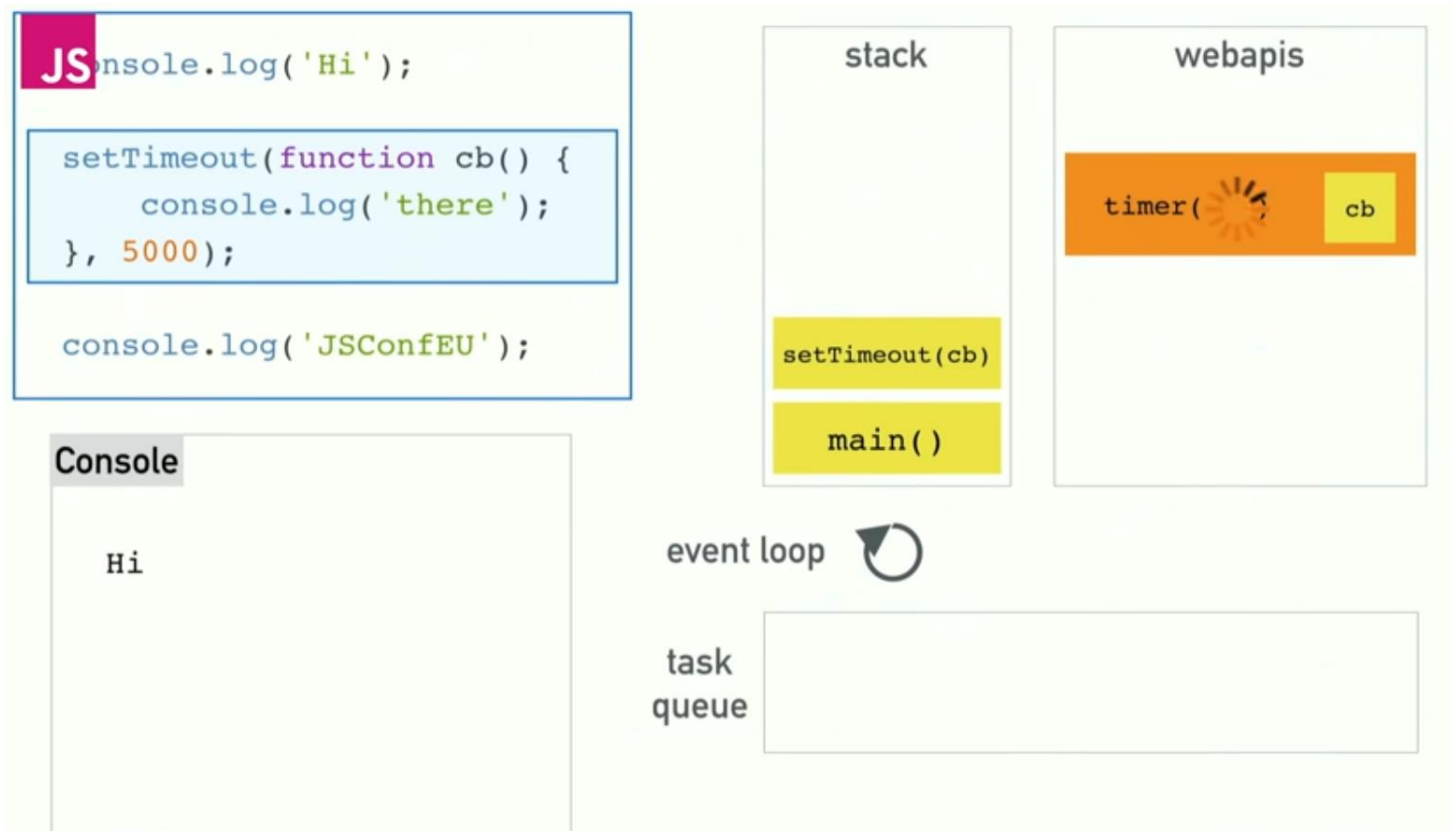
```
console.log('hi');

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

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

stack

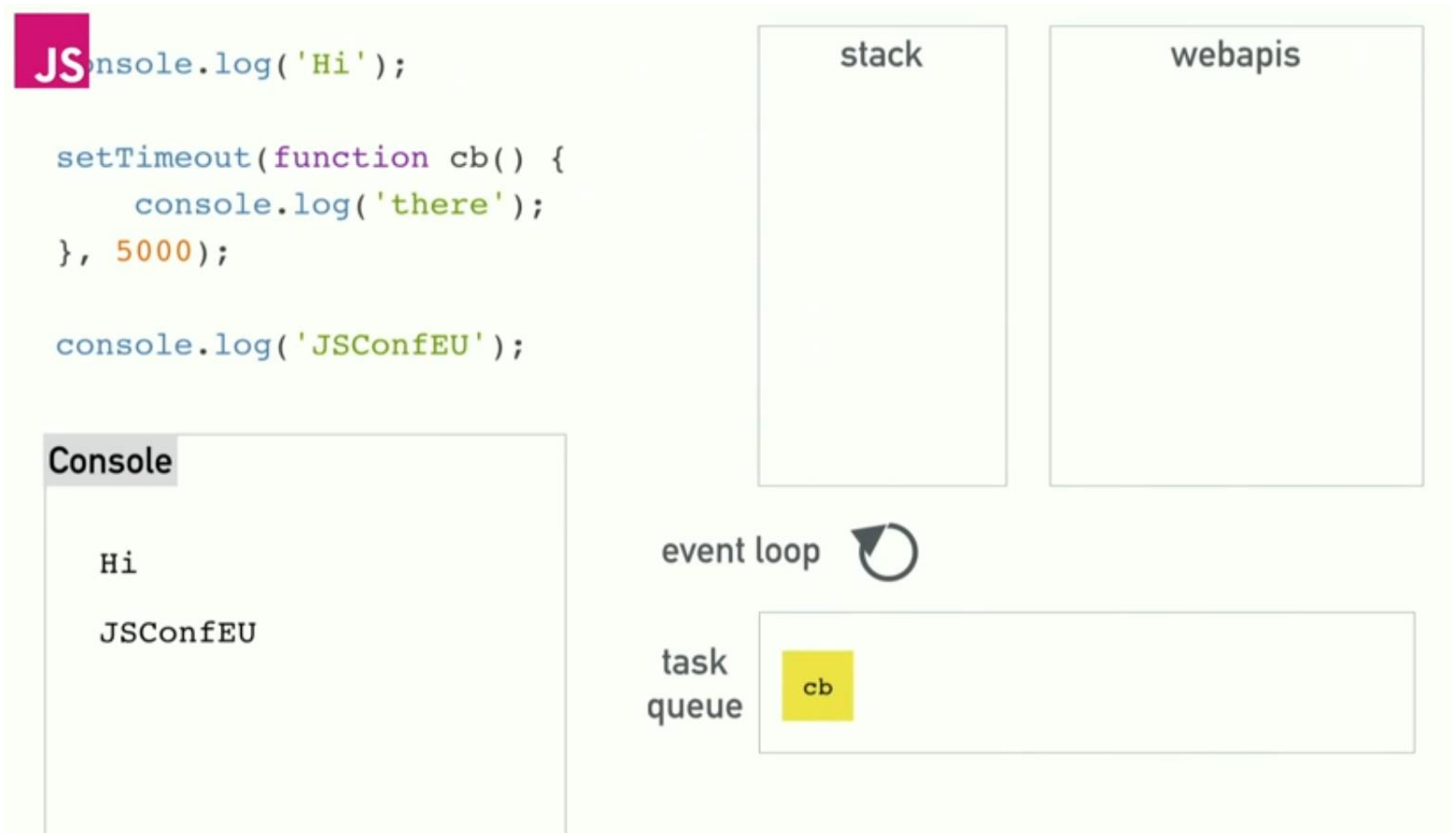
```
setTimeout(cb, 5000)
main()
```



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

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

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



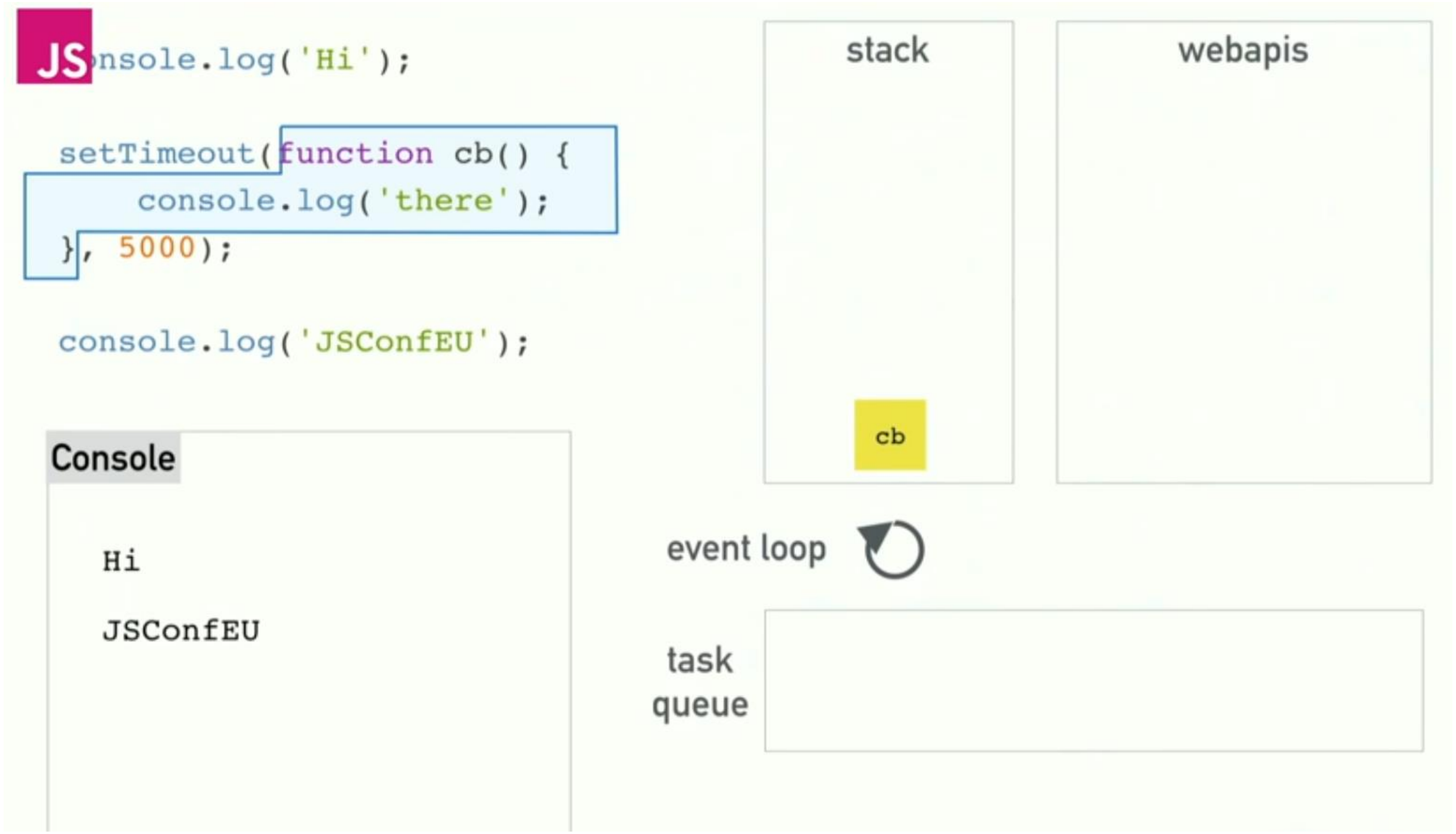
```
console.log('Hi');

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

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



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

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

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

Console

```
Hi

JSConfEU
```



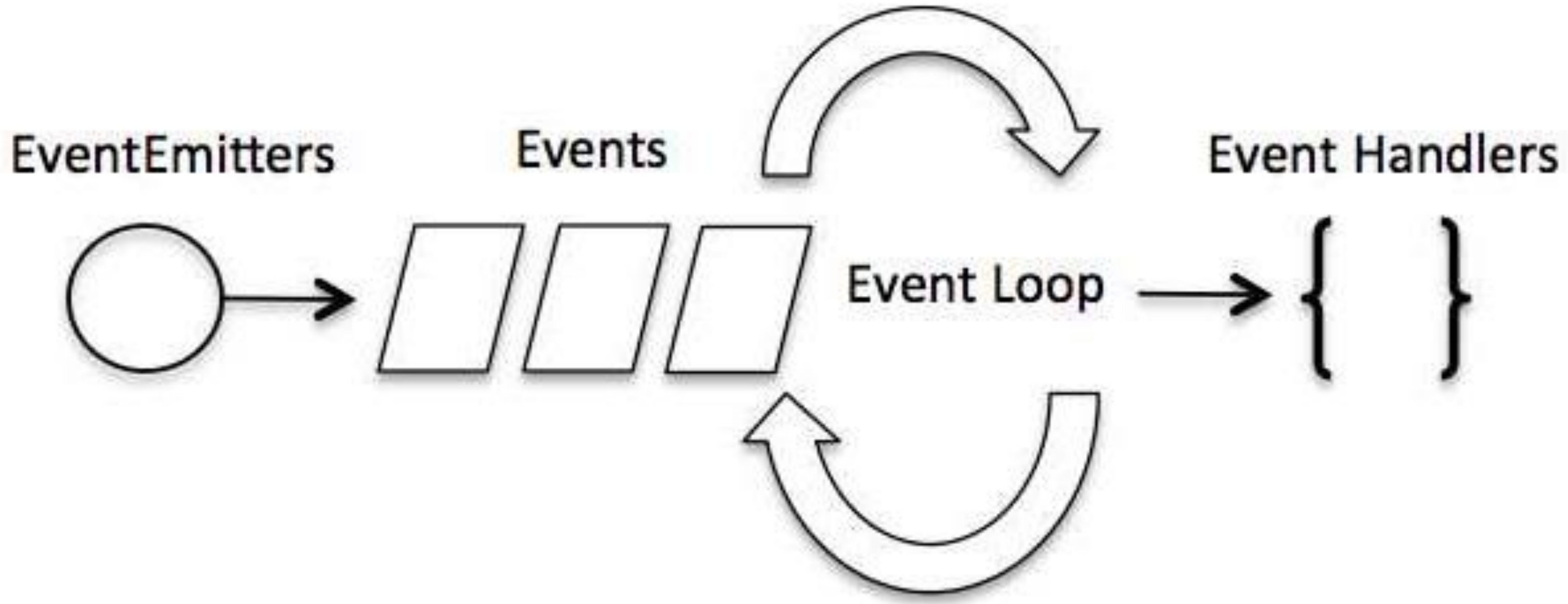
```
nsole.log('Hi');

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

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

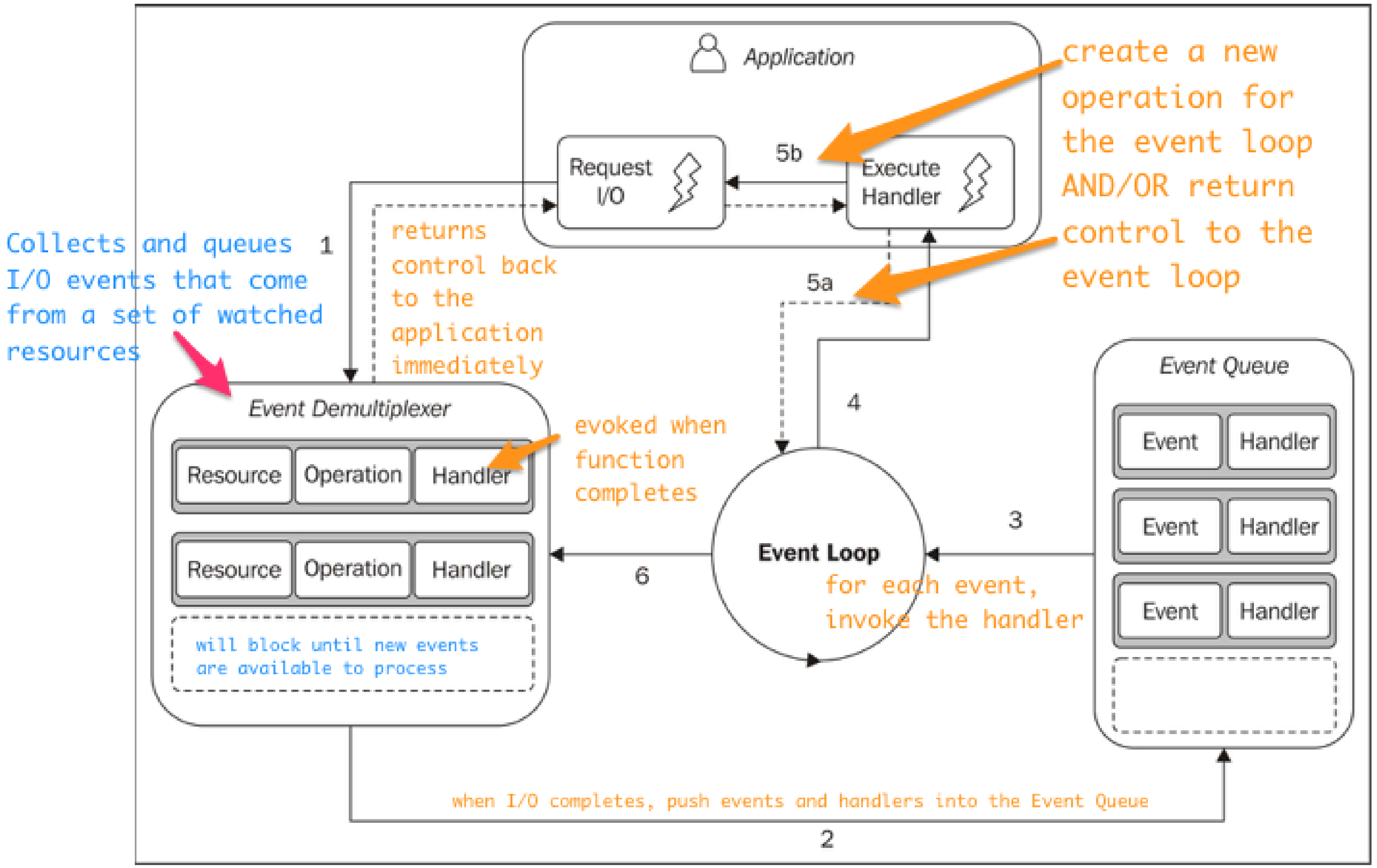
Console

```
Hi

JSConfEU
```

Event-Driven Programming

Using Event-Loop
Observer Pattern
Listeners (Observers)
Events module, EventEmitter
Webserver


EventEmitters
Events
Event Loop
Event Handlers
{ }



Application
Request I/O
5b
Execute Handler
create a new operation for the event loop AND/OR return control to the event loop
Collects and queues I/O events that come from a set of watched resources
1
returns control back to the application immediately
5a
4
Event Queue
Event
Handler
Event
Handler
Event
Handler
Event Demultiplexer
Resource
Operation
Handler
Resource
Operation
Handler
evoked when function completes
will block until new events are available to process
Event Loop
for each event, invoke the handler
6
3
when I/O completes, push events and handlers into the Event Queue
2

```
// Import events module
var events = require('events');

// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();

// Create an event handler as follows
var connectHandler = function connected() {
   console.log('connection succesful.');
   // Fire the data_received event
   eventEmitter.emit('data_received');
}

// Bind the connection event with the handler
eventEmitter.on('connection', connectHandler);
```

```
// Bind the data_received event with the
anonymous function
eventEmitter.on('data_received', function(){
   console.log('data received succesfully.');
});

// Fire the connection event
eventEmitter.emit('connection');
```

Global Objects

Globals
No need to include in modules
Consists of module, functions, strings and object

Global Objects

__filename
__dirname
setTimeout()
Console
Process

Express Framework

Minimal
Flexible web app framework
Middleware for HTTP requests
HTTP Method / URL routing
Dynamic page template rendering

Express Server

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
   res.send('Hello World');
})

var server = app.listen(8081, function () {
   var host = server.address().address
   var port = server.address().port

   console.log("Example app listening at http://%s:%s",
       host, port)
})
```

Thank You!