Assembling a Development Toolset

Paul O’Fallon

@paulofallon

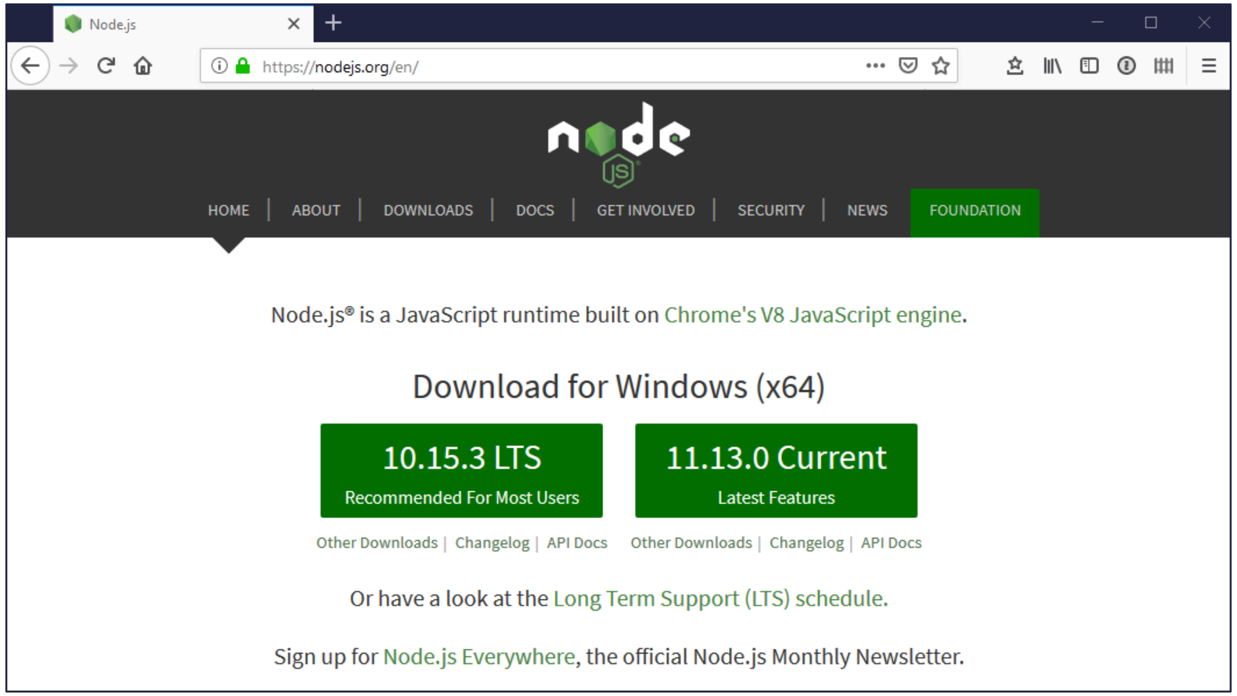
Overview

Installing Node

Building web apps and APIs

Testing and debugging Node apps

Installing Node.js



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for Windows (x64)

10.15.3 LTS
Recommended For Most Users

11.13.0 Current
Latest Features

Other Downloads | Changelog | API Docs   Other Downloads | Changelog | API Docs

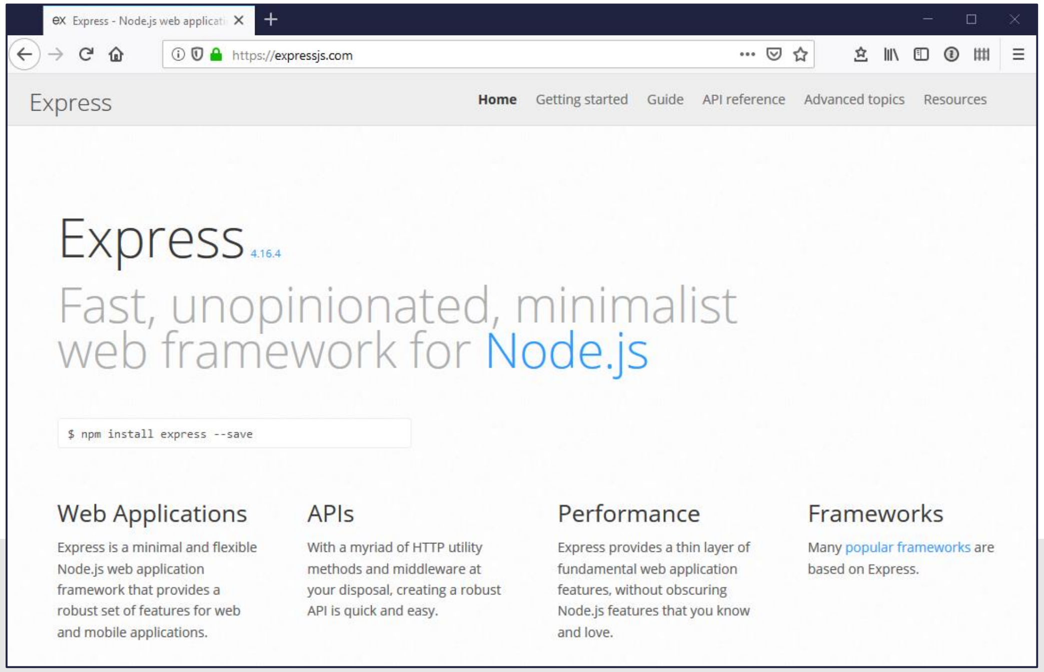
Or have a look at the Long Term Support (LTS) schedule.

Sign up for Node.js Everywhere, the official Node.js Monthly Newsletter.

Mac	**brew install nodejs**
Windows	**choco install nodejs**
Linux	**use nvm (https://github.com/creationix/nvm)**
Docker	**https://hub.docker.com/_/node/**

Building Web Apps and APIs with Express

- **Middleware**
- **Routing**
- **Template Engines**



Express 4.16.4

Fast, unopinionated, minimalist web framework for Node.js

```
$ npm install express --save
```

Web Applications

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

APIs

With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.

Performance

Express provides a thin layer of fundamental web application features, without obscuring Node.js features that you know and love.

Frameworks

Many popular frameworks are based on Express.

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening on port ${port}!`))
```

Testing Node.js Applications

Mocha

Test Framework

mochajs.org

```
describe('a cook', () => {
  it('should prepare food', done => {
    let order = 'pizza’
    cook.prepareFood(order, (error, food) => {
      assert.equal(food, order)
      done()
    })
  })
})
```

Testing Node.js Applications

Mocha

Test Framework

mochajs.org

Chai

Assertion Library

www.chaijs.com

```
describe('a customer', () => {
  it('should place an order', done => {
    let menu = ['pizza', 'wings', 'hotdog’]
    let customer = new Customer()
    customer.placeOrder(menu, (error, order) => {
      expect(order).to.be.oneOf(menu)
      done()
    })
  })
})
```

Testing Node.js Applications

Mocha
Test Framework
mochajs.org

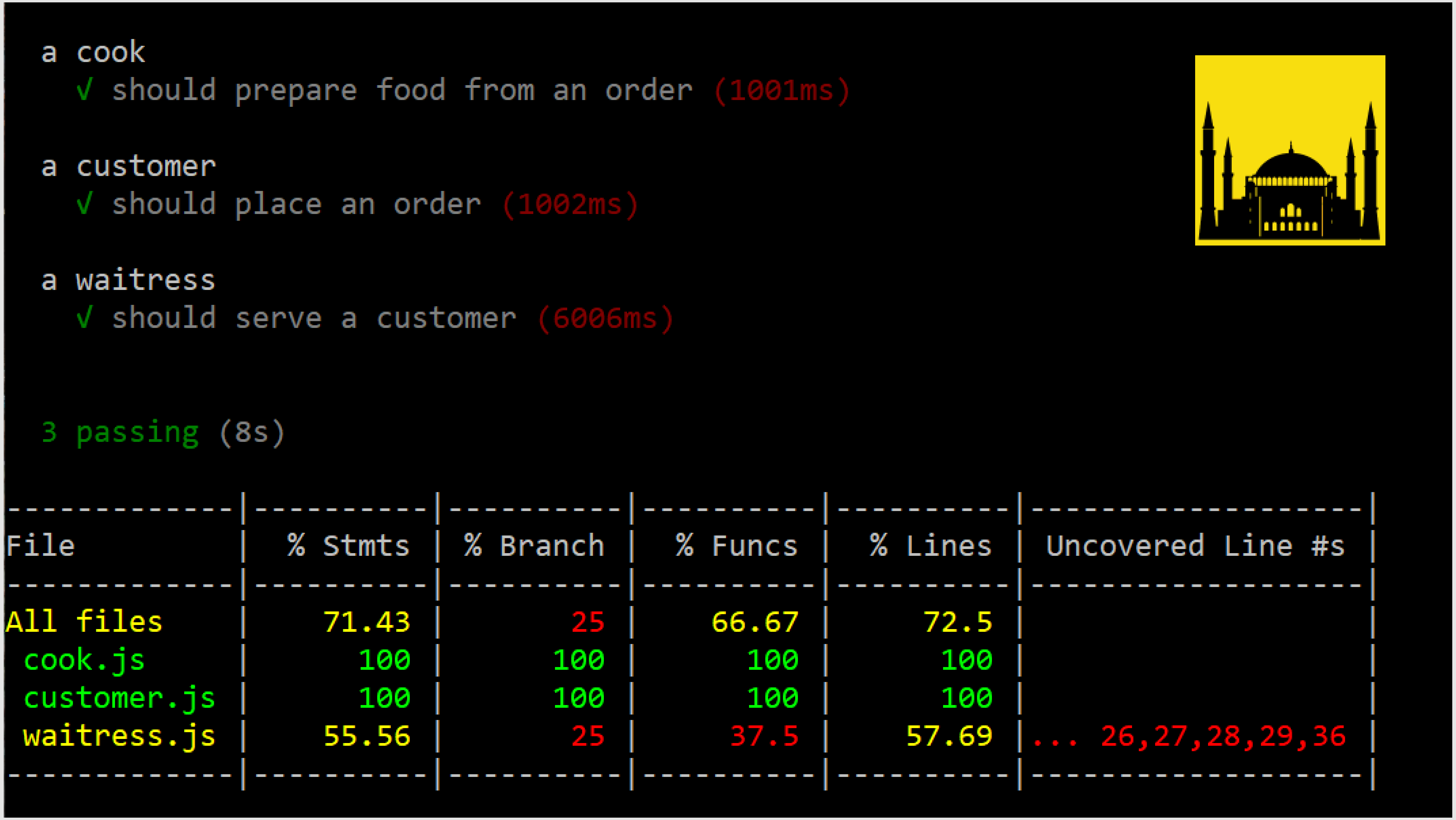
Chai
Assertion Library
www.chaijs.com

Sinon
Spies, Stubs and Mocks
sinonjs.org

```
describe('a waitress', () => {

  it('should serve a customer', done => {

    let stub = sinon.stub(cook, 'prepareFood’)
      .callsFake((error, done) => done('pizza’))

    waitress.serveCustomer(new Customer(), () => {

      expect(stub.calledOnce).to.be.true

      done()

    })

  })

})
```

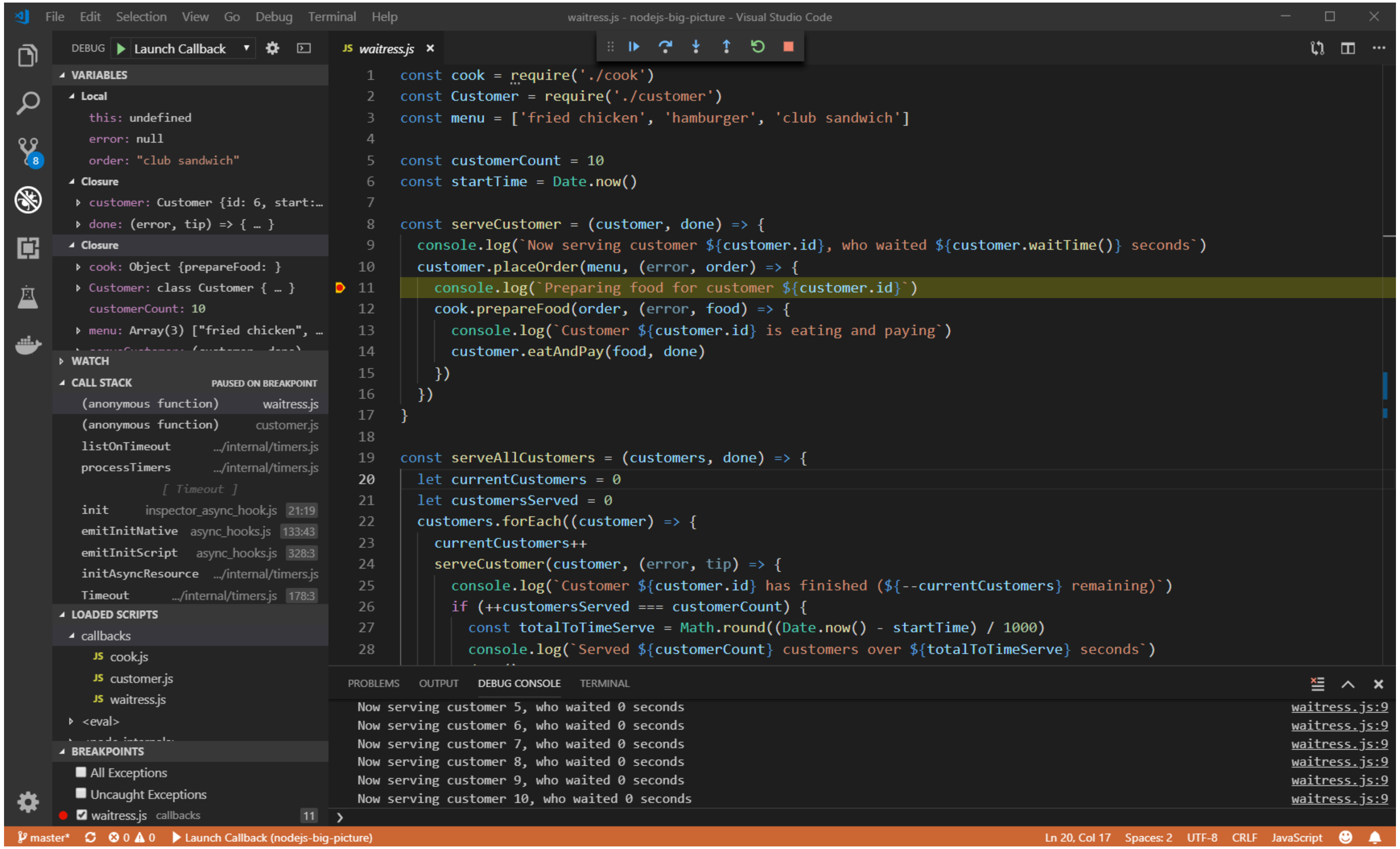
Testing Node.js Applications

Mocha	**Chai**	**Sinon**	**Istanbul**
Test Framework	Assertion Library	Spies, Stubs and Mocks	Code Coverage
mochajs.org	www.chaijs.com	sinonjs.org	istanbul.js.org



```
 a cook
   √ should prepare food from an order (1001ms)

 a customer
   √ should place an order (1002ms)

 a waitress
   √ should serve a customer (6006ms)


 3 passing (8s)

-------------|----------|----------|----------|----------|-------------------|
File         |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s |
-------------|----------|----------|----------|----------|-------------------|
All files    |    71.43 |       25 |    66.67 |     72.5 |                   |
 cook.js     |      100 |      100 |      100 |      100 |                   |
 customer.js |      100 |      100 |      100 |      100 |                   |
 waitress.js |    55.56 |       25 |     37.5 |    57.69 |... 26,27,28,29,36 |
-------------|----------|----------|----------|----------|-------------------|
```

Debugging Node.js Applications



```
VARIABLES
  Local
    this: undefined
    error: null
    order: "club sandwich"
  Closure
    customer: Customer {id: 6, start:…
    done: (error, tip) => { … }
  Closure
    cook: Object {prepareFood: }
    Customer: class Customer { … }
    customerCount: 10
    menu: Array(3) ["fried chicken", …
WATCH
CALL STACK                                 PAUSED ON BREAKPOINT
  (anonymous function)       waitress.js
  (anonymous function)       customer.js
  listOnTimeout              …/internal/timers.js
  processTimers              …/internal/timers.js
          [ Timeout ]
  init                 inspector_async_hook.js  21:19
  emitInitNative       async_hooks.js  133:43
  emitInitScript       async_hooks.js  328:3
  initAsyncResource    …/internal/timers.js
  Timeout              …/internal/timers.js  178:3
LOADED SCRIPTS
  callbacks
    cook.js
    customer.js
    waitress.js
  <eval>
BREAKPOINTS
  All Exceptions
  Uncaught Exceptions
  waitress.js  callbacks  11
```

```javascript
const cook = require('./cook')
const Customer = require('./customer')
const menu = ['fried chicken', 'hamburger', 'club sandwich']

const customerCount = 10
const startTime = Date.now()

const serveCustomer = (customer, done) => {
  console.log(`Now serving customer ${customer.id}, who waited ${customer.waitTime()} seconds`)
  customer.placeOrder(menu, (error, order) => {
    console.log(`Preparing food for customer ${customer.id}`)
    cook.prepareFood(order, (error, food) => {
      console.log(`Customer ${customer.id} is eating and paying`)
      customer.eatAndPay(food, done)
    })
  })
}

const serveAllCustomers = (customers, done) => {
  let currentCustomers = 0
  let customersServed = 0
  customers.forEach((customer) => {
    currentCustomers++
    serveCustomer(customer, (error, tip) => {
      console.log(`Customer ${customer.id} has finished (${--currentCustomers} remaining)`)
      if (++customersServed === customerCount) {
        const totalToTimeServe = Math.round((Date.now() - startTime) / 1000)
        console.log(`Served ${customerCount} customers over ${totalToTimeServe} seconds`)
```

```
Now serving customer 5, who waited 0 seconds                 waitress.js:9
Now serving customer 6, who waited 0 seconds                 waitress.js:9
Now serving customer 7, who waited 0 seconds                 waitress.js:9
Now serving customer 8, who waited 0 seconds                 waitress.js:9
Now serving customer 9, who waited 0 seconds                 waitress.js:9
Now serving customer 10, who waited 0 seconds                waitress.js:9
>
```

Summary

Common Node.js use cases

A brief history

Asynchronous development

Modules, dependencies and npm

Testing and debugging Node applications