BCDV 1022

Node Fundamentals I

2023 Fall

week 02 - class 02

Mike Denton

Topics

- **Node Fundamentals I**
  - Modules, Export, Require
  - Events and Event Emitter

Node Fundamentals I

Modules, Export, Require

Video - Modules & Require

What is a module?

- A module encapsulates related code into a single unit of code.
- A module is a reusable piece of JavaScript which exports specific objects, making them available for other modules to require in their programs
- Each module in Node.js has its own context, so it cannot interfere with other modules or pollute global scope.

Requiring Modules

- In order to use Node.js core or NPM modules, you first need to import it using require()
- The require() function will return an object, function, property or any other JavaScript type, depending on what the specified module returns.

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const mongoose=require('mongoose');
const movieRouter = require('./routes/movies');
const adminRouter = require('./routes/admin');
const Movie = require('./models/movie.ts');
```

Three sources of Node modules

1. **Build-in modules**

- Come pre-packaged with Node
- Are required with a simple string identifier
  - var f = required('foo');
- A sample of built-in modules include:
  - fs
  - http
  - crypto
  - os

Three sources of Node modules

#2: Your Projects' files

- Each .js file is its own module
- A way to modularize your application's code
- Each file is required with file system-like semantics

```
var data = require('./data');          // data.js in the same directory
var foo = require('./other/foo');      // foo.js in the other subdirectory
var bar = require('../lib/bar');       // bar.js in 'lib' directory
```

Three sources of Node modules

#2 Your Project's files

Variables are marked for export via 'module.exports'

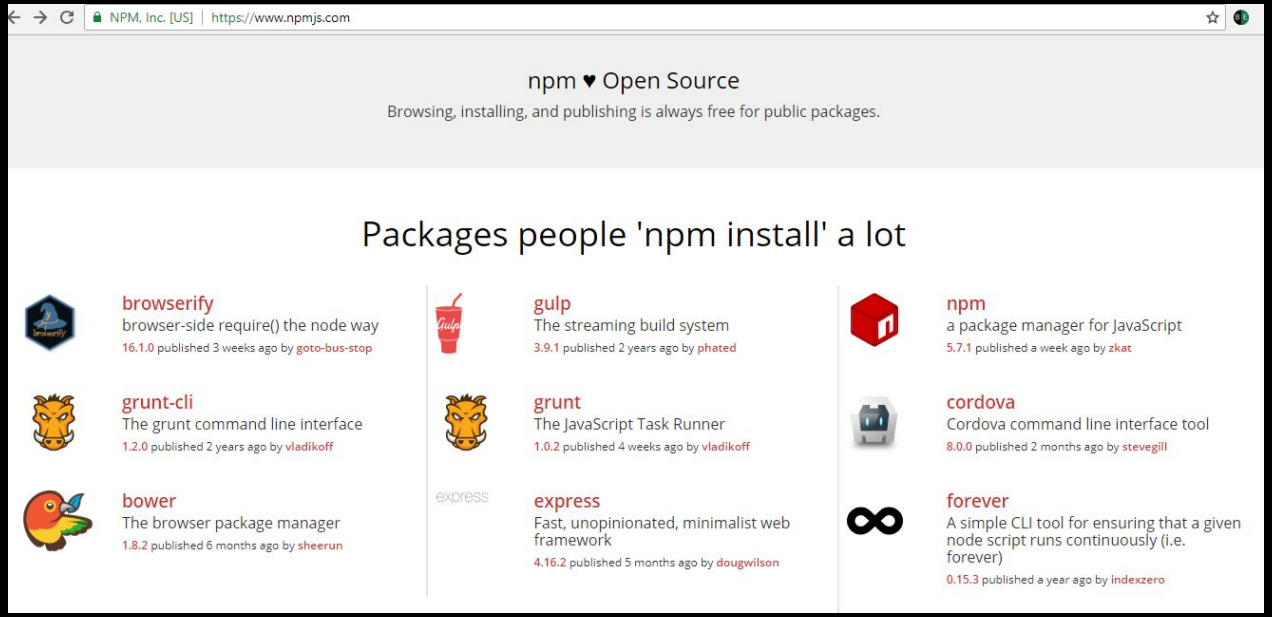
- Q: Is answers variable available to caller?

```
var answers = 0;

var greeter = function () {
    console.log("Hello? Anyone there?");
}

module.exports.greeter = greeter;
```

- There are a few different module export patterns.
  https://darrenderidder.github.io/talks/ModulePatterns

Three sources of Node modules

3 Third Party Modules via Node Package Manager (NPM) registry



Three sources of Node modules

3 Third Party Modules via Node Package Manager (NPM) registry

- Installed via "npm install module_name" *into* "node_modules" *folder*

- Are required()'d via simple string identifiers, similar to built-ins
  - var http = require('http');

- Some modules provide command line utilities as well
- Install these modules with "npm install -g module_name"
  - *Examples include: express, mocha*

    ** Note: difference between install -g and install --save*

How do do modules really work?

- **Require** is a function, that you pass a path too
- **Module.exports** is what the require function returns
- *This works because **your code is actually wrapped in a function** that is given these things as function parameters*

Video - Module Patterns

Events and Event Emitter

Event Emitters vs Events in NodeJs

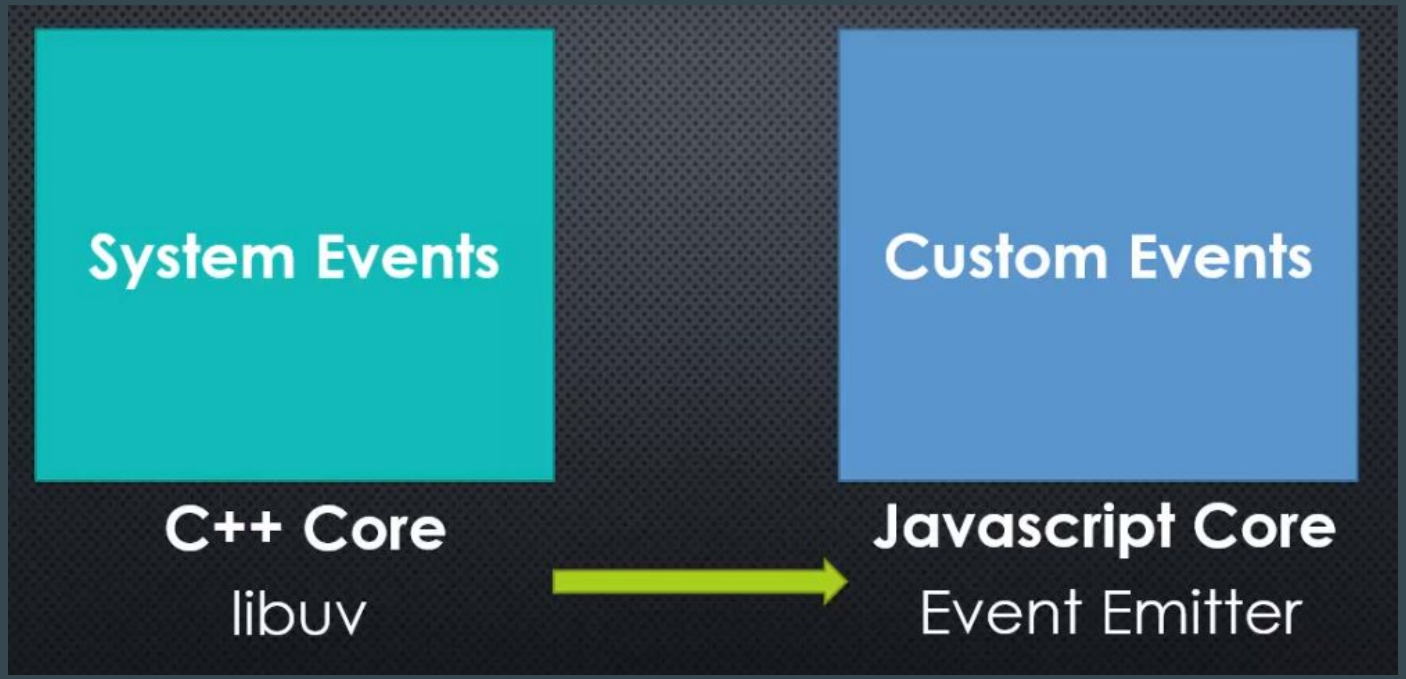
Event Emitters

- Node core is based on asynchronous event-driven architecture.
- Emitter objects periodically emit events that cause listener objects to be called.
- When the EventEmitter object emits an event, all of the functions attached to that specific event are called synchronously.

Events

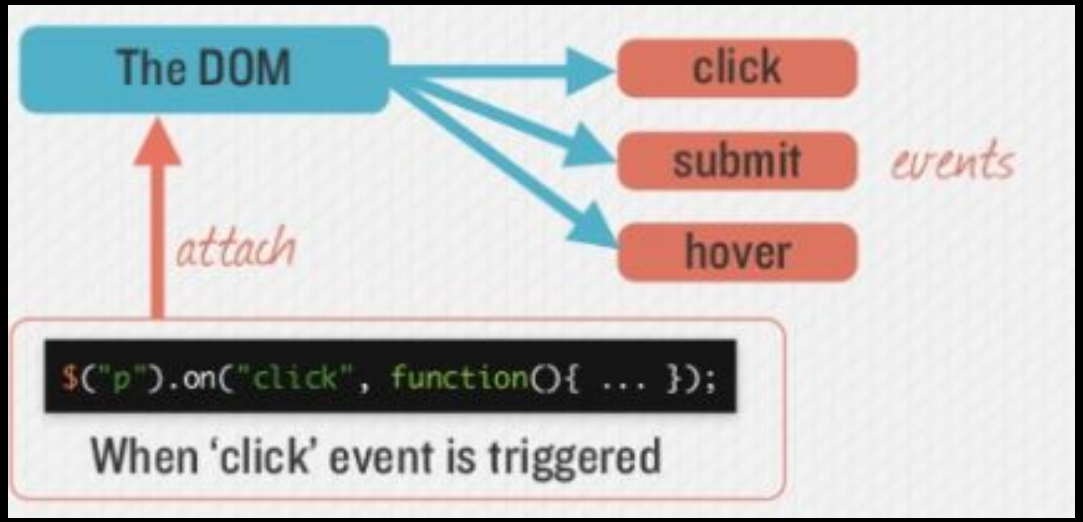
- An Event is something that has happened in our app that we can respond to.

Two different types of events in Node

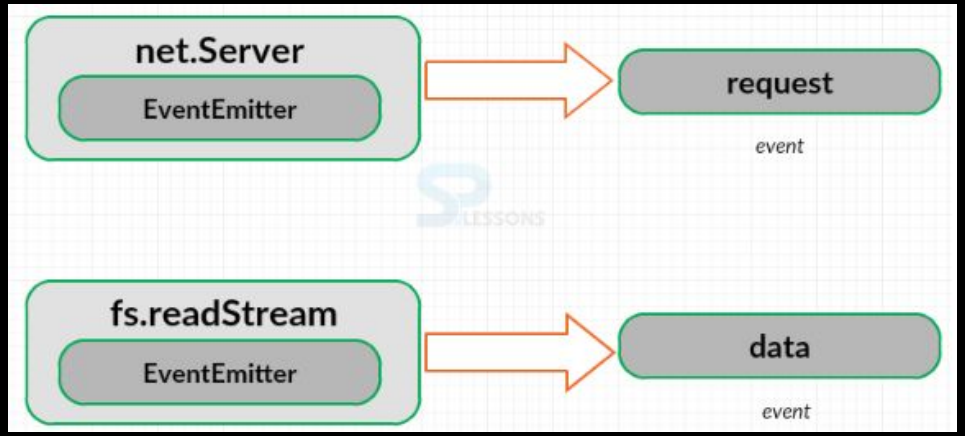


Events in the DOM

- The DOM triggers Events you can listen for those events



Node Emits Events



- The net.Server class inherits from EventEmitter, and it emits the request event.
- If user reading a file and call fs.readStream then it returns a stream which inherits from EventEmitter and will emit the data event as user reading the data out of the file.

Event Emitter

Create a new Event Emitter

```
var events = require('events');

var eventEmitter = new events.EventEmitter();

eventEmitter.on('data_received', function() {
    console.log('data received succesfully.');
});

eventEmitter.emit('data_received');
```

1. Use the events module
2. Create a new event emitter
3. Create a callback function for the event
4. Trigger the event.

Event Emitter Example

```
// get the reference of EventEmitter class of events module
var events = require('events');

//create an object of EventEmitter class by using above reference
var em = new events.EventEmitter();

//Subscribe for FirstEvent
em.on('FirstEvent', function (data) {
    console.log('First subscriber: ' + data);
});

// Raising FirstEvent
em.emit('FirstEvent', 'This is my first Node.js event emitter example.');
```

You can also use addListener() methods to subscribe for an event as shown below.

```
var emitter = require('events').EventEmitter;

var em = new emitter();

//Subscribe FirstEvent
em.addListener('FirstEvent', function (data) {
    console.log('First subscriber: ' + data);
});

//Subscribe SecondEvent
em.on('SecondEvent', function (data) {
    console.log('First subscriber: ' + data);
});

// Raising FirstEvent
em.emit('FirstEvent', 'This is my first Node.js event emitter example.');

// Raising SecondEvent
em.emit('SecondEvent', 'This is my second Node.js event emitter example.');
```

Video - Event Emitter

Custom Event Emitter

Custom Event Emitters

- One can create their own Custom EventEmitter using the EventEmitter constructor

```
var EventEmitter = require('events').EventEmitter;
```

```
var logger = new EventEmitter();
```

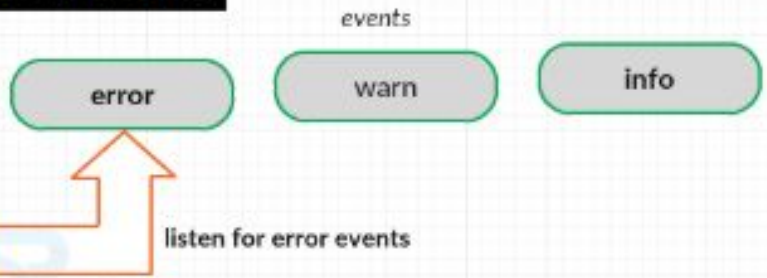
```
logger.on('error', function(message){
    console.log('ERR: ' + message);
});
```

```
logger.emit('error', 'Egg Cracked');
```

--> ERR : Egg Cracked

```
logger.emit('error', 'Spilled Milk');
```

--> ERR : Spilled Milk



Common Patterns for EventEmitters

There are two common patterns that can be used to raise and bind an event using EventEmitter class in Node.js.

1. Return EventEmitter from a function
2. Extend the EventEmitter class

Return Custom Event Emitter

```
const emitter = require('events').EventEmitter;

const MyEmitter = () => {
    var e = new emitter();
    return e;
}

const myEmitter = MyEmitter();

myEmitter.on('event', () => {
    console.log('an event occurred!');
  });

  myEmitter.emit('event');
```

Extend Custom Event Emitter

```
const EventEmitter = require('events');

class MyEmitter extends EventEmitter {}

const myEmitter = new MyEmitter();

myEmitter.on('event', () => {
  console.log('an event occurred!');
});

myEmitter.emit('event');
```