Event-driven and asynchronous programming with JavaScript and Node.js

Trayan Iliev

IPT – Intellectual Products & Technologies
e-mail: tiliev@iproduct.org
web: http://www.iproduct.org

Agenda

1. Non-blocking IO. Event loop. Using callbacks and promises.
2. Node.js and npm – installation, packages, command line arguments. Running scripts.
3. Using Node.js shell (REPL)
4. Using Visual Studio Code.
5. Modules and module usage patterns, require, core modules.
6. Global objects in Node.js.
7. Developing hello-world web server using HTTP module.
8. Routing requests. HTTP methods.
9. Developing HTTP clients using http.get() and http.request().
10. Creating custom modules.

Where is The Code?

Full-stack Development with Node.js and React.js code is available @GitHub:

https://github.com/iproduct/course-node-express-react

Brief History of Node.js

- Node.js is created by Ryan Dahl in 2009
- Initially only for Linux – now everywhere!
- Using Google's V8 JavaScript engine

Source: Wikipedia (https://en.wikipedia.org/wiki/Node.js), License: CC-SA

Node.js Main Features

- Highly efficient server-side platform based on **Google V8 JS engine**, compiles JS to executable code Just In Time (JIT) during execution (used both - at the client & server)
- Combines V8 with **non-blocking event loop** +low-level **I/O API**
- **npm package manager** – introduced in 2011 – easier publishing and sharing source code of Node.js libraries and is designed to simplify (un)installation, updating of libraries
- **Node.js** – single thread, non-blocking I/O calls, thousands of concurrent connections, without cost of thread context switching
- Sharing a single thread between all the requests using **observer pattern**, any function performing I/O uses **callback**

Node.js Platform & Language Support

- Node.js applications can run on **Linux, Mac OS X, Microsoft Windows, NonStop**, and **Unix** servers
- Apps written in **JavaScript (ES5, ES6, ES7), CoffeeScript, Dart** or **TypeScript** (strongly typed forms of JavaScript), or any other language that can compile to JavaScript
- **Node.js Foundation (2015)** – both **Node.js** and **io.js** communities voted to work under the Node.js Foundation, facilitated by Linux Foundation's Collaborative Projects program
- Many clones: **EventMachine** for Ruby, **libevent** for C, Perl **Object Environment** - Perl, **Twisted** for Python, **Vert.x** - Java, JavaScript, Groovy, Ruby, Python, Scala, Clojure and Ceylon

Node.js Use

- Node.js corporate users include: GoDaddy, Groupon, IBM, LinkedIn, Microsoft, Netflix, PayPal, Rakuten, SAP, Voxer, Walmart, Yahoo!, and Cisco Systems.
- Node.js has an **event-driven architecture capable of asynchronous I/O** - optimizes throughput and scalability in Web applications with many input/output operations, as well as for real-time Web applications (e.g., **real-time communication programs and browser games**).
- Node.js allows the creation of **Web servers and networking tools** using JavaScript and a collection of **"modules"** that handle various core functionality

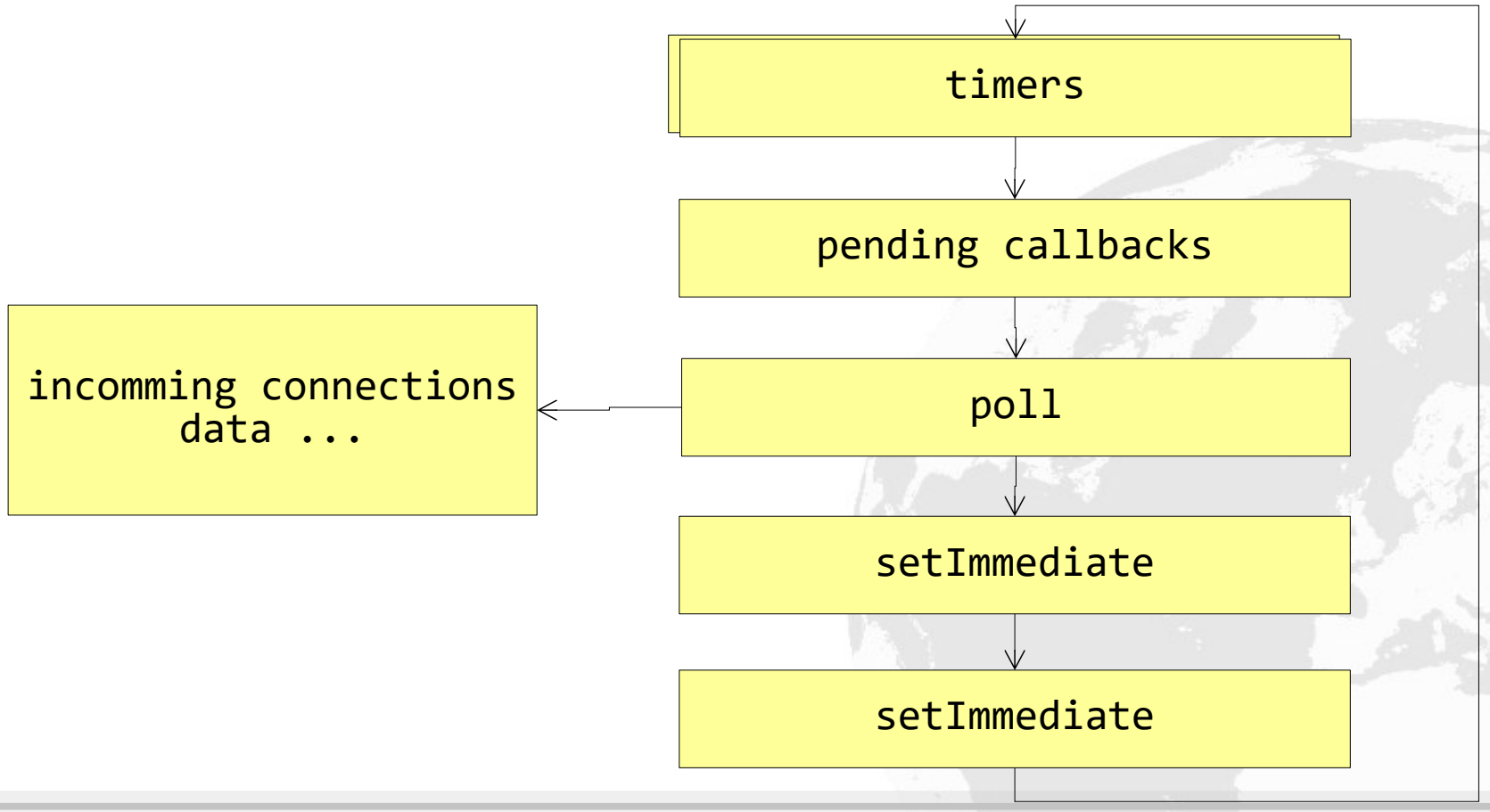
Source: Wikipedia (https://en.wikipedia.org/wiki/Node.js),

Node.js Event Loop

- Node.js registers itself with the operating system so that it is notified when a connection is made, and the **operating system will issue a callback**.
- Within the Node.js runtime, **each connection is a small heap allocation** - traditionally, relatively heavyweight OS processes or threads handled each connection. Node.js uses an event loop for scalability, instead of processes or threads.
- Node.js's event loop does not need to be called explicitly. Instead callbacks are defined, and the **server automatically enters the event loop** at the end of the callback definition.
- Node.js exits the event loop when there are no further callbacks to be performed.

Source: Wikipedia (https://en.wikipedia.org/wiki/Node.js),

Event loop. Non-blocking IO. Callbacks



Event Loop Phases

- Timers – this phase executes callbacks scheduled by setTimeout() and setInterval()
- Pending callbacks – executes I/O callbacks deferred to the next loop iteration
- Poll – retrieve new I/O events; execute I/O related callbacks (almost all with the exception of close callbacks, the ones scheduled by timers, and setImmediate()); node will block here when appropriate
- Check – setImmediate() callbacks are invoked here
- Close callbacks – some close callbacks, e.g. socket.on('close',)

Node.js Main Modules

- **HTTP** – web server and clients, routing, HTTP methods support
- **File System** – asynchronous and synchronous file and directory operations, getting read/write streams, statistics, watching file-system for changes
- **Buffer** - reading or manipulating streams of binary data, ES 6 Uint8Array TypedArray instances, size of the Buffer is established when it is created and cannot be resized
- **Events** - asynchronous event-driven architecture in which certain kinds of objects (instances of the EventEmitter class) periodically emit named events that cause Function objects ("listeners") to be called.

Source: Wikipedia (https://en.wikipedia.org/wiki/Node.js),

Node.js Main Modules

- **Streams** - Readable, Writable, Duplex and Transform Streams and piping them in streaming pipelines
- **Process** – creating and managing processes, statistics, spawn(), exec(), fork(), event-driven inter-process communication
- And more ...

Node.js by Example

Required Module
(explicit dependency management)

Callback Function
Continuation- Passing Style (CPS)

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

dns.resolve4('iproduct.org', function (err, addresses) {
    if (err) throw err;
    console.log('addresses: ' + JSON.stringify(addresses));
});
```

Node.js by Example – Simple HTTP Server

```
const http = require('http');
const hostname = '127.0.0.1';
const port = 3000;
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:$
{port}/`);
});
```

Lets do Some Practice ...

Node.js API:

https://nodejs.org/dist/latest-v11.x/docs/api/

Node.js interactive tutorials:

https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

https://www.airpair.com/javascript/node-js-tutorial

http://www.tutorialspoint.com/nodejs/index.htm

https://github.com/substack/stream-handbook

https://nodesource.com/blog/understanding-the-nodejs-event-loop/

https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/

https://nodejs.org/en/docs/guides/anatomy-of-an-http-transaction/

Thanks for Your Attention!

Questions?