# Node.js

Day 1

---

Alexandre Perrin

July 10, 2017

nomades.ch

- alex@kaworu.ch
- Open Source enthusiast
- Working daily with Node.js since 2014

## Today

Node.js

    The event loop

    CLI and debugging

    HTTP server

    release schedule
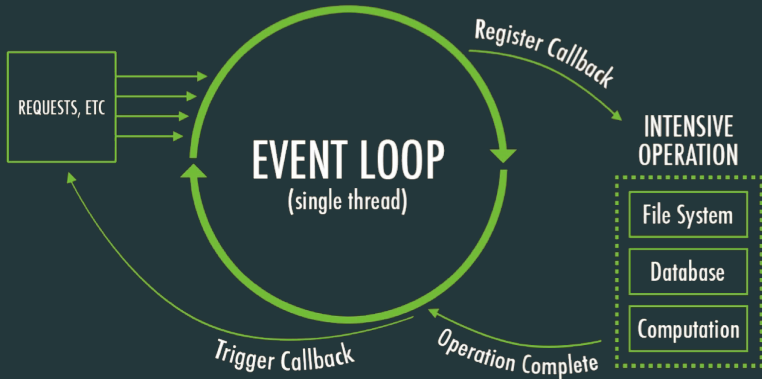
npm

    package.json

Express.js

nodemon

Let's code!

# Node.js

A server-side Javascript run-time environment.

- is based on Google's V8 Javascript Engine
- has an event-driven architecture
- handle I/O asynchronously

You already know how the event loop work from the browser:

```
1   // executed immediately by the main thread.
2   console.log("zero seconds");
3
4   // register a callback that will be executed once the operation is completed.
5   // In this case the "operation" is waiting two seconds.
6   setTimeout(() => console.log("two seconds."), 2000);
7
8   // the same as previously, although this operation will be completed faster.
9   // Thus, this callback will be executed before the previous one by the main
10  // thread.
11  setTimeout(() => console.log("one second."),  1000);
12
13  // As the first console.log(), this will be executed immediately by the main
14  // thread.
15  console.log("some seconds");
```

## Node.js script example

```
1   // always start with this for clarity and speed.
2   "use strict";
3
4   // this is how modules are "imported"
5   const fs = require("fs");
6
7   // console.log() works in Node.js too
8   console.log("sync ready");
9   // NOTE: readFileSync() I/O is executed in the main thread,
10  // could throw an Error.
11  let buf = fs.readFileSync("/etc/hosts");
12  console.log("done reading.");
13  console.log(buf.toString());
14
15  console.log("async ready");
16  // NOTE: readFile() I/O is executed in a worker thread,
17  // could callback an Error.
18  fs.readFile("/etc/hosts", function callback(err, buf) {
19      console.log(buf.toString());
20  });
21  console.log("done reading?");
```

## REPL

```
% node
> console.log("hello form the REPL :)")
hello form the REPL :)
undefined
> x = 1.9
1.9
> x
1.9
> _
1.9
> x === 1.8 + 0.1
false
```

### debugging

```
% node debug ./event-loop.js
< Debugger listening on [::]:5858
connecting to 127.0.0.1:5858 ... ok
break in event-loop.js:2
  1 // always start with this for clarity and speed.
> 2 "use strict";
  3
  4 // this is how modules are "imported"
debug> help
Commands: run (r), cont (c), next (n), step (s), out (o), backtrace (bt),
      setBreakpoint (sb), clearBreakpoint (cb),
watch, unwatch, watchers, repl, exec, restart, kill, list, scripts, breakOnException,
      breakpoints, version
```

### The classic Hello World

```
 1  "use strict"; // always
 2
 3  // "http" and "util", like "fs", are standard Node.js module.
 4  // see https://nodejs.org/dist/latest-v6.x/docs/api/
 5  const http = require("http");
 6  const util = require("util");
 7
 8  // the port on which our server will listen on.
 9  const port = 8080;
10
11  // create a http server to handle requests.
12  const server = http.createServer((req, res) => {
13      // util.inspect() is helpful for debugging. See also console.dir().
14      console.log(util.inspect(req.socket.address(), {colors:true}));
15      res.end(`Welcome to ${req.url}\n`); // ES6 Template literals
16  });
17
18  // listen() will register into the event loop.
19  server.listen(port, () => console.log("listening on port " + port));
20
21  // just like "done reading?" in our previous example, this will be printed
22  // *before* the server is listening.
23  console.log("EOF");
```
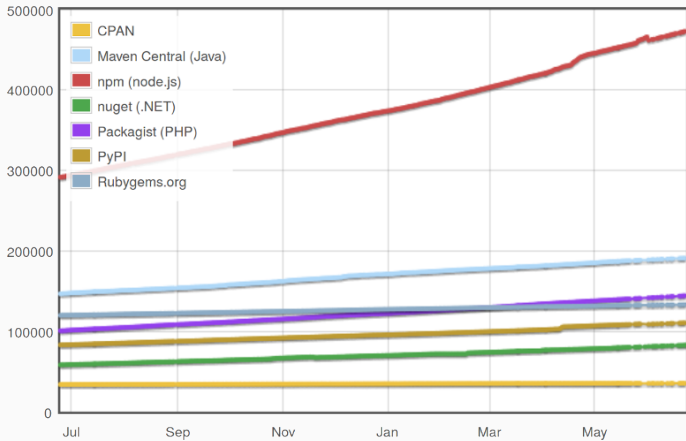
# LTS release schedule

11
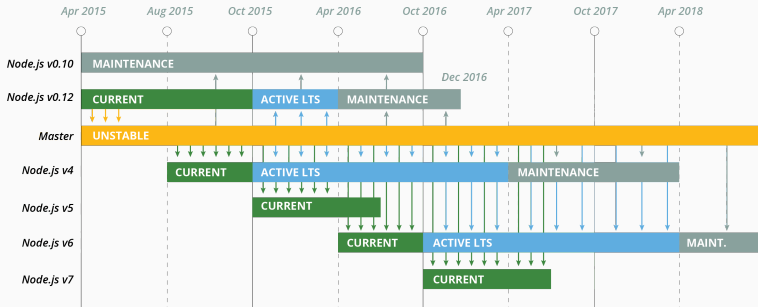
**npm**

## What is npm?

npm is the Node.js package manager. It is a command line tool that is primarily used to interact with the *npm registry* (for example to install modules).

# the npm registry



modulecounts.com

13

```
% npm install express
```

will install the "express" module from the registry into the node_modules/ directory. Once installed, you may *require* it:

```
const express = require("express");
```

## package.json

package.json contains the dependencies of your project (and some other metadata). The usual way to create it is by calling:

```
% npm init
```

Consider setting `private` to avoid accidently publishing your module to the public registry.

Once your project has a package.json, you can tell npm to install a module and save it as a dependency:

```
% npm install --save express
```

More later and at https://docs.npmjs.com

# Express.js

# Express.js

Express.js is a very popular minimalist web server framework for Node.js inspired by Sinatra.

# Express.js

## Hello Express:

```
1   /* see http://expressjs.com/en/starter/hello-world.html */
2   const express = require('express')
3   const app = express()
4
5   app.get('/', function (req, res) {
6     res.send('Hello World!')
7   })
8
9   app.listen(3000, function () {
10    console.log('Example app listening on port 3000!')
11  })
```

## HTTP methods examples:

```
app.get("/login",       (req, res) => ...); // HTTP GET
app.post("/login",      (req, res) => ...); // HTTP POST
app.put("/users/:id",   (req, res) => ...); // HTTP PUT
app.patch("/users/:id", (req, res) => ...); // HTTP PATCH
app.delete("/users/:id", (req, res) => ...); // HTTP DELETE
```

Full documentation and examples at expressjs.com

# nodemon

## nodemon

nodemon is a tool to automatically restart your application when it crash or you make changes.

Local installation:

```
% npm install --save-dev nodemon
```

Usage:

```
% ./node_modules/.bin/nodemon --verbose
```

More about nodemon at https://nodemon.io

# Let's code!

Write a simple blogging server with express. For now we'll use a static hardcoded array of posts items:

```
 1  app.locals.posts = [
 2      {_id: 1, title: "Bacon Avocado Salad", body: "Place bacon in a large..."},
 3      {_id: 2, title: "Crispy Orange Beef", body: "Lay beef strips out in..."},
 4      {_id: 3, title: "Simple BBQ Ribs", body: "Place ribs in a large..."},
 5  ];
 6  // create
 7  app.post("/api/posts", (req, res) => ...);
 8  // read
 9  app.get("/api/posts", (req, res) => ...);
10  app.get("/api/posts/:id", (req, res) => ...);
11  // update
12  app.put("/api/posts/:id", (req, res) => ...);
13  // destroy
14  app.delete("/api/posts/:id", (req, res) => ...);
```

Questions?

Farewell Node.js by TJ Holowaychuk.