INTRODUCTION TO NODE + EXPRESS

Web Programming

TODAY'S TOPICS

- What is Node.js
- The Node REPL + process
- NPM + package.json
- Intro to Express + installation
- Hello world app
- Express basics
  - The Request and Response objects
  - Basic routing
  - Path parameters
  - Query string
  - Serving static files
- Templating basics

WHAT IS NODE.JS

- Node.js is an asynchronous event-driven JavaScript runtime.
- What Can Node.js Do?
  - Can generate dynamic page content
  - Can create, open, read, write, delete, and close files on the server
  - Can collect form data
  - Can add, delete, modify data in your database

WHY NODE.JS

Node.js uses asynchronous programming! -> JavaScript

A common task for a web server can be to open a file on the server and return the content to the client.

HOW PHP OR ASP HANDLES A FILE REQUEST:

1. Sends the task to the computer's file system.
2. Waits while the file system opens and reads the file.
3. Returns the content to the client.
4. Ready to handle the next request.

HOW NODE.JS HANDLES A FILE REQUEST:

1. Sends the task to the computer's file system.
2. Ready to handle the next request.
3. When the file system has opened and read the file, the server returns the content to the client.

LET'S TRY NODE.JS

The node REPL = Chrome's console

WRITING YOUR FIRST NODE.JS FILE

- Create a .js file and write some JavaScript code.
- In Terminal/Powershell use this command
  - > node filename.js

NODE.JS MODULES

- What is a Module in Node.js?
  - Consider modules to be the same as JavaScript libraries.
  - A set of functions you want to include in your application.

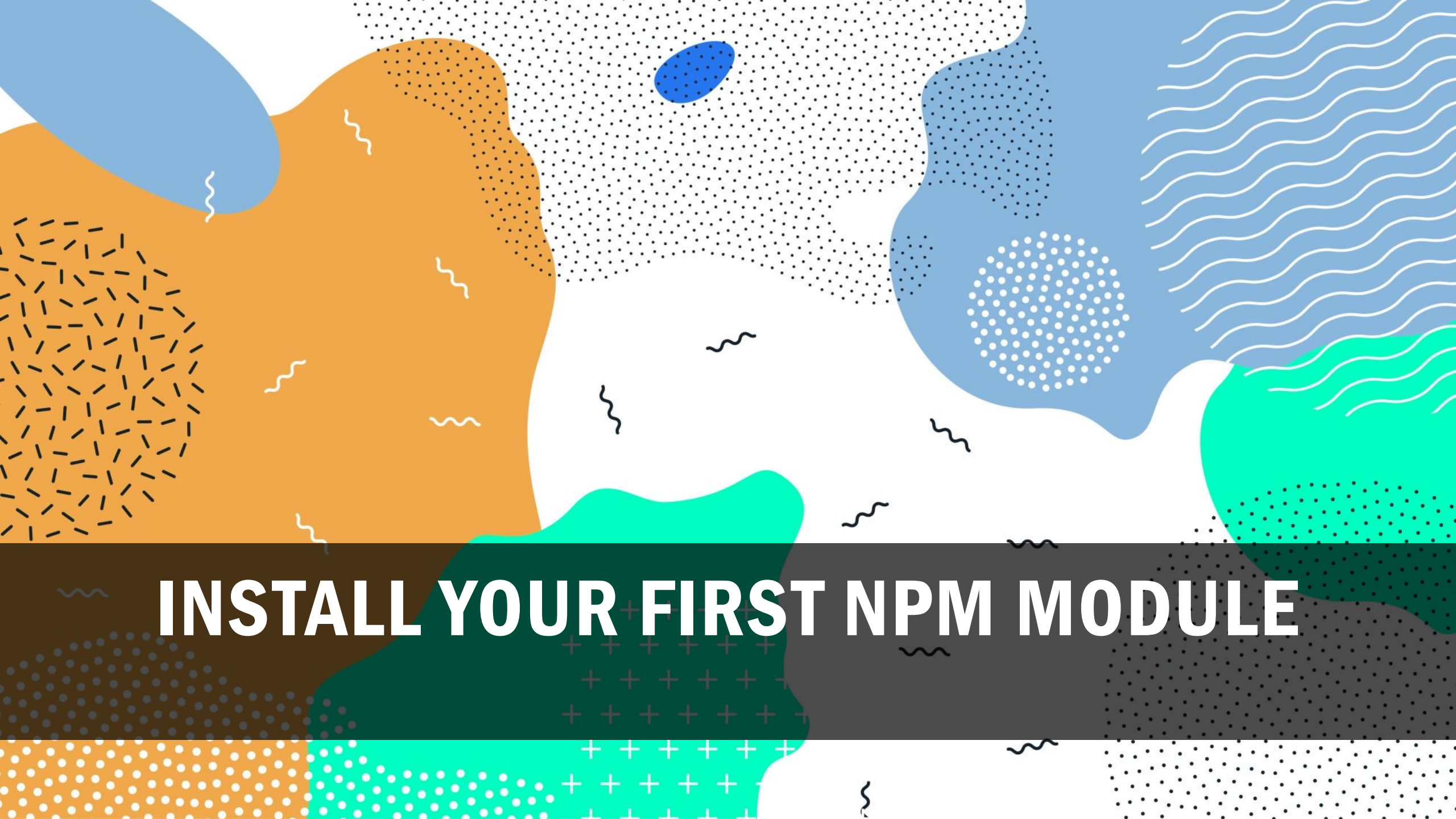
- Built-in Modules
  - Node.js has a set of built-in modules which you can use without any further installation. (ref)
- Custom Modules
  - You can also create your own modules or use others' modules

LET'S CREATE A NODE.JS MODULE

- require
- module.exports

WHAT IS NPM?

- NPM is a package manager for Node.js packages, or modules if you like.
- www.npmjs.com hosts thousands of free packages to download and use.
- The NPM program is installed on your computer when you install Node.js
- What is a Package?
  - A package in Node.js contains all the files you need for a module.
  - Modules are JavaScript libraries you can include in your project.


INSTALL YOUR FIRST NPM MODULE

UPPER-CASE

- In Terminal/Powershell, go to your project directory
- Type this command:

> npm install upper-case

- NPM creates a folder named "node_modules", where the package will be placed.
- All packages you install in the future will be placed in this folder.

upper-case TS

2.0.2 • Public • Published 3 months ago

Readme | Explore BETA | 1 Dependency

Upper Case

npm v2.0.2 | downloads 29M/month | minzipped size 371 B

> Transforms the string to upper case.

Installation

```
npm install upper-case --save
```

Usage

```
import { upperCase, localeUpperCase } from "upper-case";

upperCase("string"); //=> "STRING"

localeUpperCase("string", "tr"); //=> "STRİNG"
```

PACKAGE.JSON

- The package.json file is kind of a manifest for your project.
- It's a central repository of configuration for tools, for example. It's also where npm stores the names and versions for all the installed packages.
- Let’s try:
  - Create a folder
  - Type command: npm init


express

INTRODUCTION

Express (https://expressjs.com/) is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

What is a framework?

INSTALLATION

Node.js must be installed first!

1. Open the Terminal/Powershell
2. Create your project folder: mkdir myproject
3. Go to the folder: cd myproject
4. Create package.json file: npm init
5. Install Express: npm install express


HELLO WORLD APP
Let’s create your first Express application

THE REQUEST AND RESPONSE OBJECTS

- The req object represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, and so on. (https://expressjs.com/en/4x/api.html#req)

- The res object represents the HTTP response that an Express app sends when it gets an HTTP request. (https://expressjs.com/en/4x/api.html#res)

EXPRESS ROUTING BASICS

Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).

- Each route can have one or more handler functions, which are executed when the route is matched.

```
app.METHOD(PATH, HANDLER)
```

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

PATH PARAMETERS

Documentation:
https://expressjs.com/en/guide/routing.html

Sometimes we want to pass parameters with the URL.

- Route parameters are named URL segments that are used to capture the values specified at their position in the URL.
- The captured values are populated in the req.params object.

```
app.get('/users/:userId/books/:bookId', function (req, res) {
  res.send(req.params)
})
```

```
Route path: /users/:userId/books/:bookId
Request URL: http://localhost:3000/users/34/books/8989
req.params: { "userId": "34", "bookId": "8989" }
```

QUERY STRING

```
// GET /search?q=tobi+ferret
console.dir(req.query.q)
// => 'tobi ferret'

// GET /shoes?order=desc&shoe[color]=blue&shoe[type]=converse
console.dir(req.query.order)
// => 'desc'

console.dir(req.query.shoe.color)
// => 'blue'

console.dir(req.query.shoe.type)
// => 'converse'

// GET /shoes?color[]=blue&color[]=black&color[]=red
console.dir(req.query.color)
// => ['blue', 'black', 'red']
```

SERVING STATIC FILES

To serve static files such as images, CSS files, and JavaScript files, use the express.static built-in middleware function in Express.

```
express.static(root, [options])
```

- The root argument specifies the root directory from which to serve static assets.
- For more information on the options argument, see express.static.

TEMPLATING BASICS

Templating allows us to define a preset pattern for a webpage, that we can dynamically modify.

CREATE YOUR FIRST TEMPLATE

Steps:

1. Call app.set() in your index.js file
2. Create “views” folder in your project directory
3. Create views/home.ejs and write some HTML code
4. Call res.render(‘home.ejs’) in the controller

```
app.set('view engine', 'ejs');
```

```
app.get("/", (req, res) => {
  res.render('home.ejs')
});
```

EJS SYNTAX

Interpolation

- <% 'Scriptlet' tag, for control-flow, no output
- <%= Outputs the value into the template (HTML escaped)
- <%- Outputs the unescaped value into the template
- <%# Comment tag, no execution, no output
- %> Ending tag

PASSING DATA TO TEMPLATE

```
app.get("/", (req, res) => {
  res.render('home.ejs',
    {
      title: "This is my home",
      message: "Welcome to Web Programming!"
    }
  )
});
```

LET'S BEUTIFY OUR HOME PAGE WITH BULMA