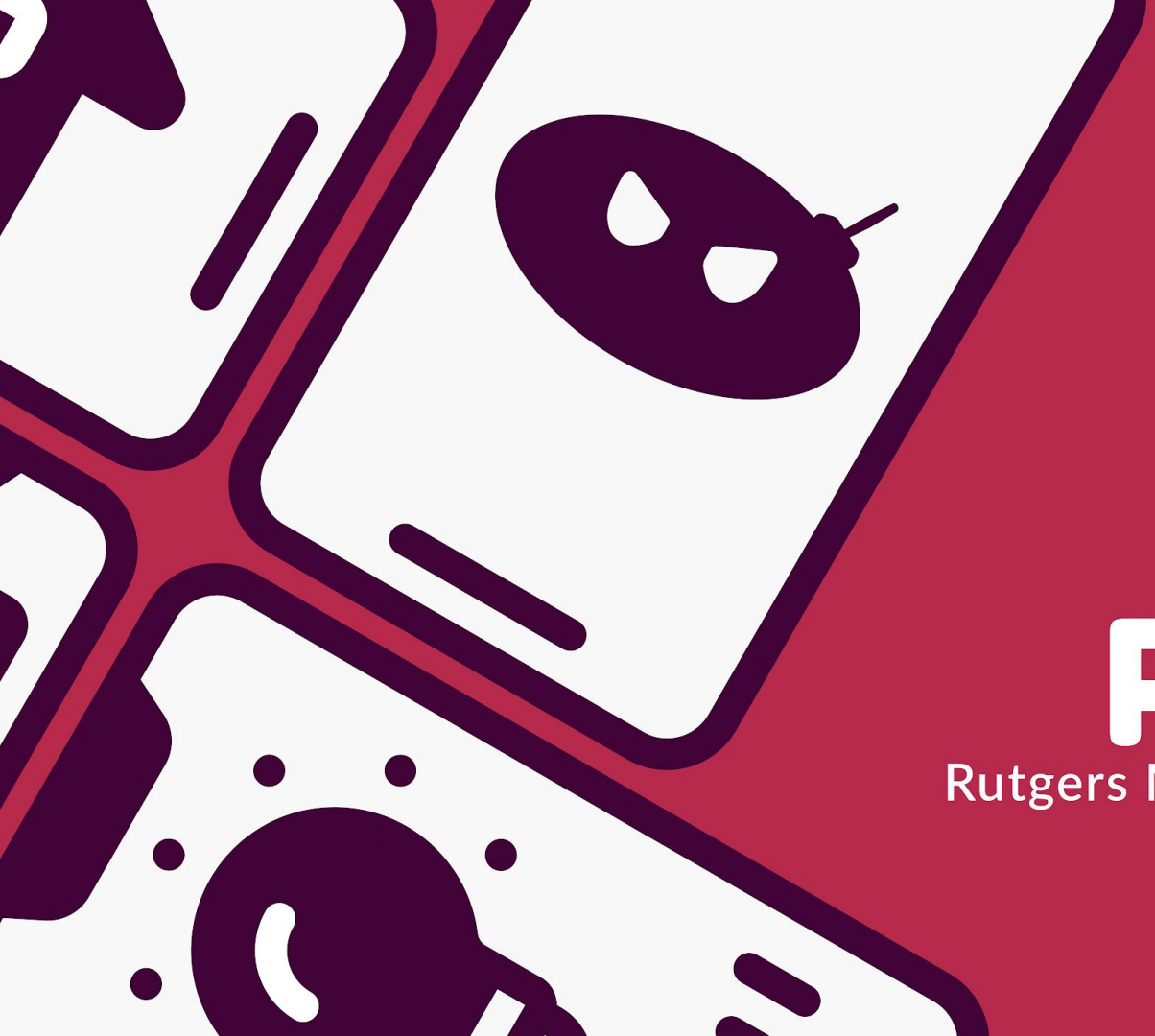


RUMAD

Rutgers Mobile App Development

Overview

For this section, we'll be covering:

- Functions
- Importing Packages

If we have time, I'll briefly go over:

- NPM (Node Package Manager)
- Express


Writing functions

Functions

- Multiple ways to define functions
  - normal way
  - arrow functions
- If you're not sure which one to use, use the "normal" syntax!

```
function func_name(args) {
    return val;
}
```

```
let func_name = ( args ) => {
    return val;
}
```

Functions

You can write anonymous functions!

```
(function(a,b,c) {
    return a * b + c
})()
```

```
((a,b,c) => {
    return a * b + c
})()
```


Let’s do some coding!

Example problems

- FizzBuzz
  - Functions version!
  - Create a function that will take in a number, and execute FizzBuzz!

412. Fizz Buzz

Easy 2.4K 313

Companies

Given an integer `n`, return *a string array* `answer` *(**1-indexed**) where*:

- `answer[i] == "FizzBuzz"` if `i` is divisible by `3` and `5`.
- `answer[i] == "Fizz"` if `i` is divisible by `3`.
- `answer[i] == "Buzz"` if `i` is divisible by `5`.
- `answer[i] == i` (as a string) if none of the above conditions are true.

Example 1:

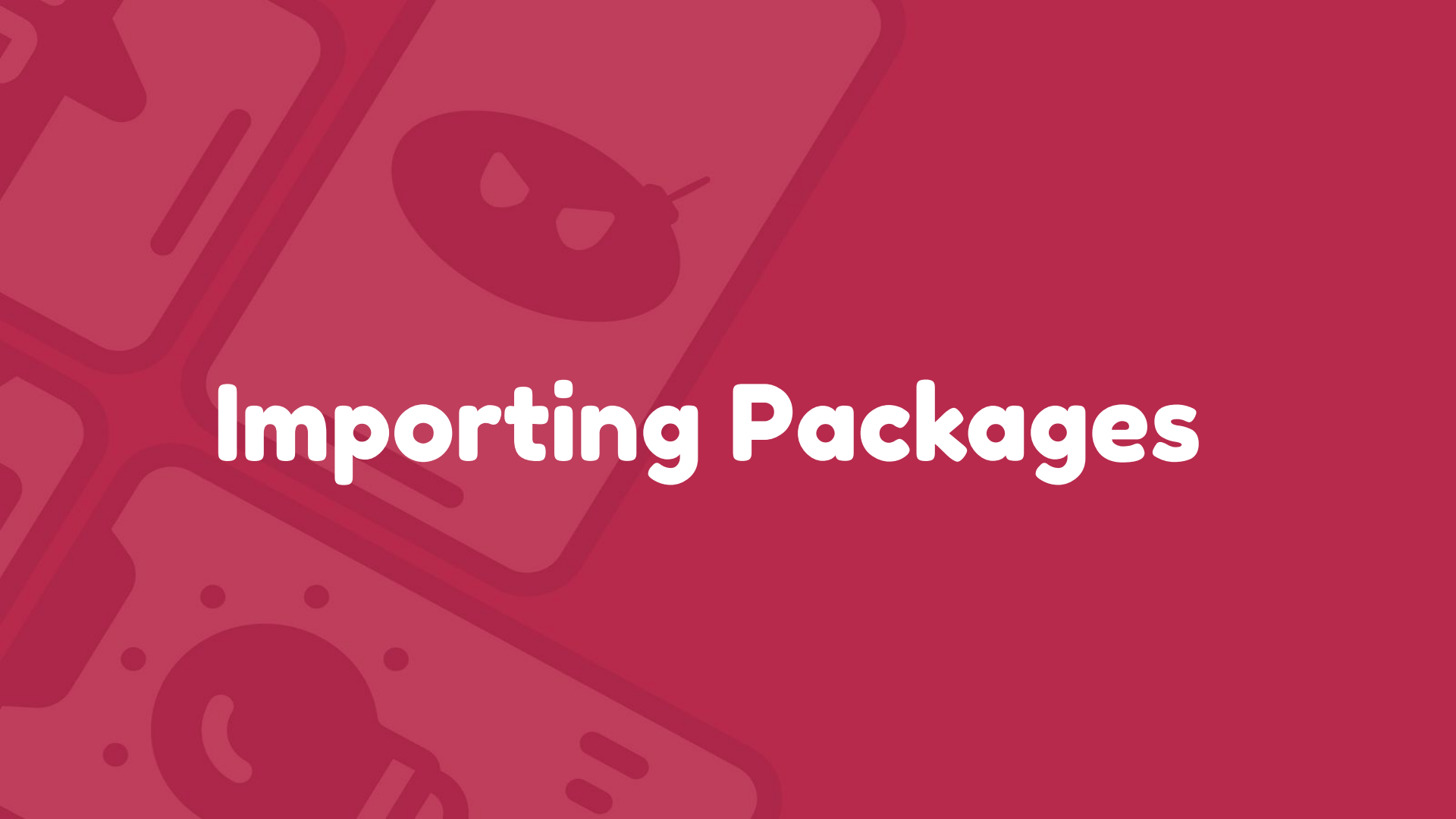
> **Input:** n = 3
> **Output:** ["1","2","Fizz"]

Example 2:

> **Input:** n = 5
> **Output:** ["1","2","Fizz","4","Buzz"]

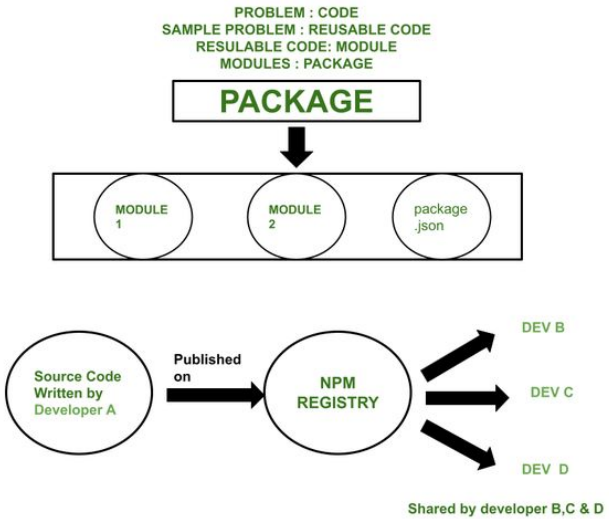
Example 3:

> **Input:** n = 15
> **Output:**
> ["1","2","Fizz","4","Buzz","Fizz","7","8","Fizz","Buzz","11","Fizz","13","14","FizzBuzz"]


Importing Packages

What are Packages?

- One or more “modules” (libraries) grouped together.
- Can be imported and provide functionality for your project.



Default Packages?

- Https, os, fs, path, etc…
- These are built-in to node, without a need to install anything.


Introducing NPM

NPM (Node Package Manager)

- “npm is a package manager for the JavaScript programming language”
- What does that mean?
  - Access and install dependencies from a massive registry
  - Quickly lookup packages

NPM - How do we use it?

- The basic commands are:
  - **npm -v** to confirm the installation
  - **npm init** to initialize a folder as a local package
  - **npm install <package_name>** to install packages
    - npm i <package_name>

Let's test these commands out!

NPM - Configuration file

```
{
  "name": "week2",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

NPM - Configuration file

What else should we know?

- Dependencies
- Using scripts

NPM - Configuration file

What are dependencies?

- “a piece of code—a library, a module, or a package—that a project requires to function correctly”

What are *dev* dependencies?

- Dependencies that are only used in **development** environment
- e.g: *a dependency used for debugging or testing the application*

NPM - Configuration file

What are scripts in NPM and why do we use them?

- Scripts are shortcuts to commands
- Makes workflow quicker
- Run frequently used commands “**npm run <script>**”

First look at Express

What is Express?

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

Express provides us with functionality to:

- Route traffic
- Write and use middleware
- Implement API endpoints

… and has a variety of open-source middleware to help with development!

Express “Hello World”

Let’s create a Hello World application in Node!

To follow along, search **Express “Hello World”**.

Hello world example

Embedded below is essentially the simplest Express app you can create. It is a single file app — ***not*** what you'd get if you use the Express generator, which creates the scaffolding for a full app with numerous JavaScript files, Jade templates, and sub-directories for various purposes.

```
const express = require('express' 4.18.2 )
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}`)
})
```

Save on **RunKit** | Node 10 | help

This app starts a server and listens on port 3000 for connections. The app responds with "Hello World!" for requests to the root URL (/) or ***route***. For every other path, it will respond with a **404 Not Found**.

The example above is actually a working server: Go ahead and click on the URL shown. You'll get a response, with real-time logs on the page, and any changes you make will be reflected in real time. This is powered by RunKit, which provides an interactive JavaScript playground connected to a complete Node environment that runs in your web browser. Below are instructions for running the same app on your local machine.

Questions?

Please fill out the feedback form when you have a chance!

Feedback Form

Next week...

- More on Express
  - Setting up API endpoints
  - Serve index page
  - Create a basic application