

BUNDLERS

Lesson 8



Lesson Plan

1

Automatization - What is it

2

Bundlers - Types and Features

3

Node.js -> npm -> package.json

4

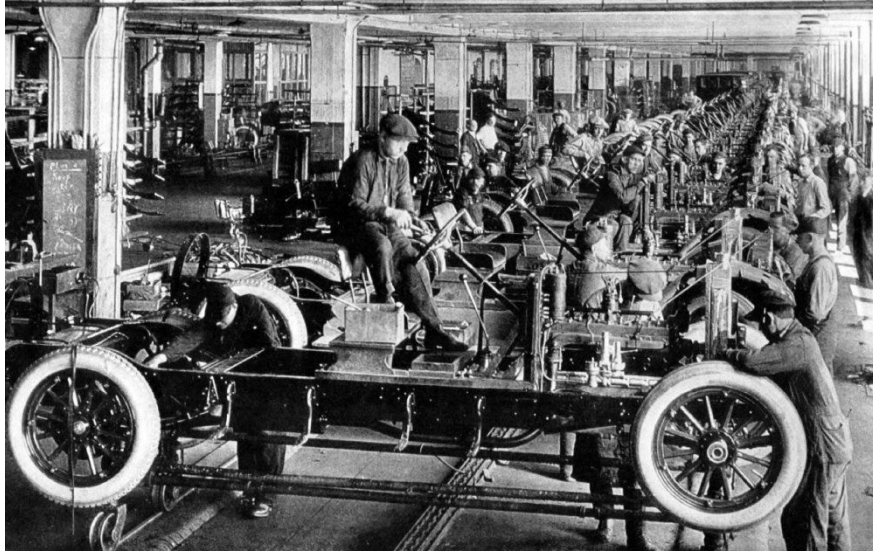
Gulp

Automatization

is the process of transferring routine development tasks that can be done manually

- ✓ It **saves the developer's time**
- ✓ helps to **find and fix errors**
- ✓ generates code that **saves the browser's resources** and **speeds up website loading**.

BEFORE



AFTER





◆ A file packager 📦

◆ A tool that collects files into one, optimizes them, and speeds up the website. 🚀

💡 Many brilliant things are born from laziness. Build systems make developers' lives easier! 😎

Types of bundlers

- ◆ **Webpack** – The most popular and powerful, but complex
- ◆ **Parcel** – Simple, but slow
- ◆ **Vite** – Fast and convenient



We will use Gulp



webpack



VITE



PARCEL





What does a bundler do?

 **Combines files**

File transformation:

 **Removes duplication**


 **Minifies code**

 **Supports SCSS, PostCSS, images**

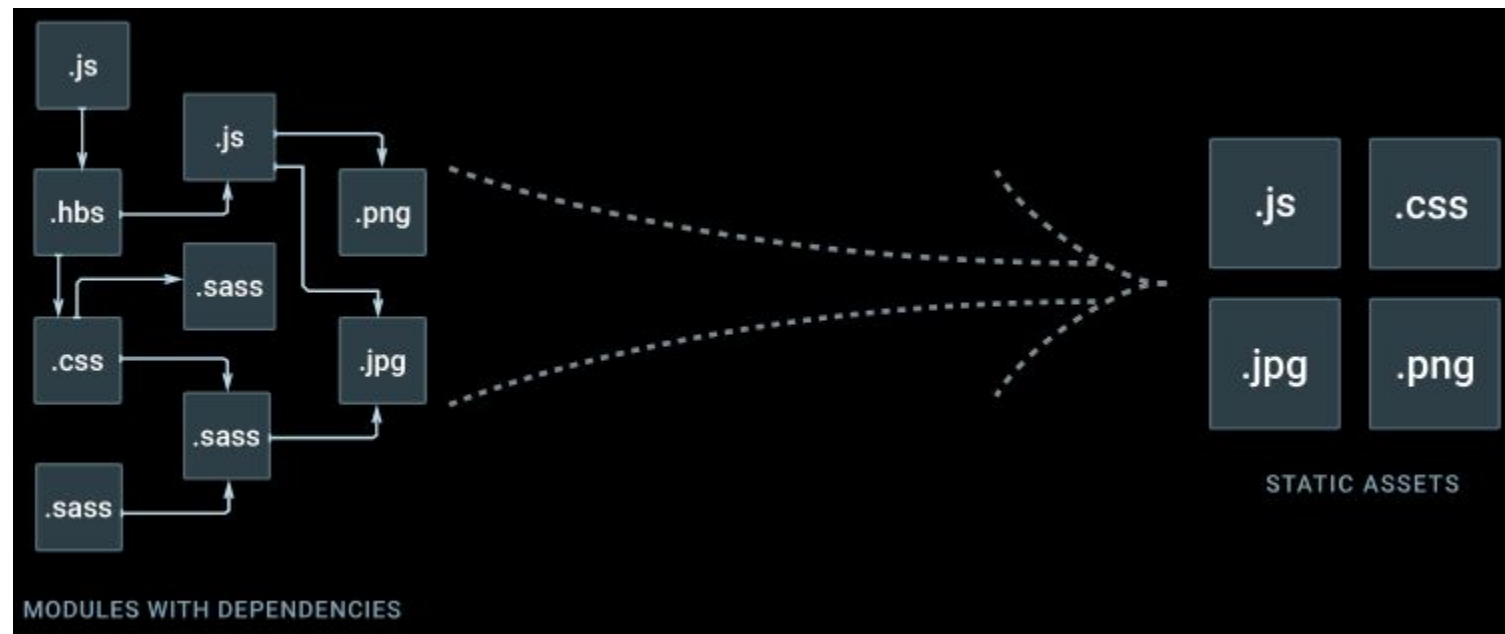
Without a bundler:

 **Many separate CSS and JS files**

 **Code duplication makes maintenance difficult**

 **Slow loading (no minification)**

 **No convenient tools (SCSS, PostCSS, images)**



1 Development Server & Live Reload

 **Runs a local development server with auto-reloading:**

- `browser-sync` → **Serves files and refreshes browser automatically** when changes occur.

 **Why?**

- ✓ **Instant feedback** while coding (no need to manually refresh).
- ✓ **Allows multiple devices to preview changes live.**

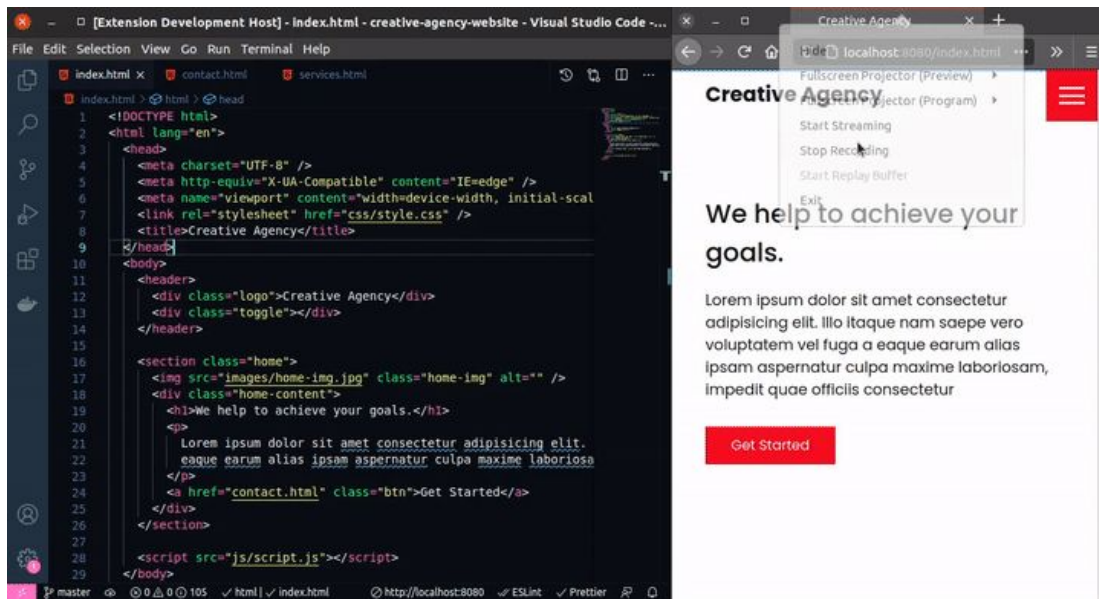
Live Server

Automatic Page Refresh

Edit →

Save →

Browser Updates!



2 CSS Processing & Linting

Tools for working with SCSS & PostCSS:

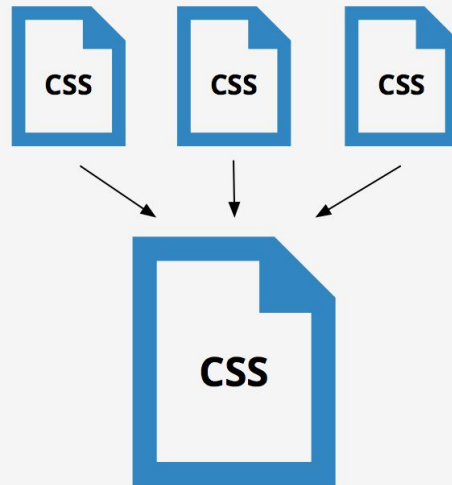
- `sass` → Compiles SCSS to CSS
- `postcss` → Processes CSS with plugins
- `autoprefixer` → Adds vendor prefixes automatically
- `csso` → Minifies CSS for better performance
- `sourcemaps` → Generates **source maps** for debugging

Why?

- ✓ **SCSS support** with clean syntax and mixins
- ✓ **Autoprefixer ensures cross-browser compatibility**
- ✓ **Minified CSS loads faster** for better performance
- ✓ **Source maps make debugging easier** in DevTools

Work with styles files

✓ **Combines files**



```
.entry-content p {  
  font-size: 14px !important;  
}  
  
.entry-content ul li {  
  font-size: 14px !important;  
}  
  
.product_item p a {  
  color: #000;  
  padding: 10px 0px 0px 0;  
  margin-bottom: 5px;  
}
```

Minify



```
.entry-content p,.entry-content ul li  
{font-size:14px!important}.product_item  
p a{color:#000;padding:10px 0 0;margin-  
bottom:5px;border-bottom:none}
```

✓ **Minifies code**

Source map

Settings

Preferences

Preferences

Sources

Workspace

Experiments

Blackboxing

Devices

Throttling

Locations

Shortcuts

Show whitespace characters: None

☒ Display variable values inline while debugging

☒ Enable CSS source maps

☒ Allow scrolling past end of file

Default indentation: 4 spaces

```
▼<div class="width-wrapper">
  ▶<div class="title-header pens-title-
header">...</div>
  ▼<div id="picks-grid" class="pen-grid
view-grid">
    ▼<div class="single-pen
      " data-slug-hash="aygmoG">
        ▶<div class="iframe-wrap loaded">
...</div>
        ▶<div class="meta">...</div> == $0
      </div>
    ▶<div class="single-pen
      " data-slug-hash="bryQGJ">...</div>
```

```
}
.meta {
  background: ▶ ■ #32333b;
  position: relative;
  font-size: 0.85em;
  white-space: nowrap;
  padding: ▶ 1rem 1rem 0.5rem;
  z-index: 3;
}
* {
  margin: ▶ 0;
  padding: ▶ 0;
  webkit-box-sizing: border-box;
  box-sizing: border-box;
}
```

page.css:528

global.css:3

IMPORTANT

For style file use only this path with folder and file name:

```
<link rel="stylesheet" href="/css/style.css">
```

3 Image Optimization & Responsive Images

📌 Processes images for performance optimization:

- `sharpResponsive` → Creates responsive images using Sharp.

💡 Why?

- ✓ Converts images to **different sizes for responsive design**.
- ✓ Optimizes file size without losing quality.
- ✓ Generates modern formats like **WebP and AVIF**.

📌 Combines multiple **SVG** icons into a single optimized **sprite file**:

- `gulp-svg-sprite` → Creates **SVG sprites** to reduce HTTP requests.

💡 We will use it later

Before

Quality: great

78

PageSpeed Score

Original image

JPG

1.2 MB



Optimize

After

Quality: great

100

PageSpeed Score

Optimized image

WebP

0.3 MB

IMPORTANT

For images use only path with this folder name:

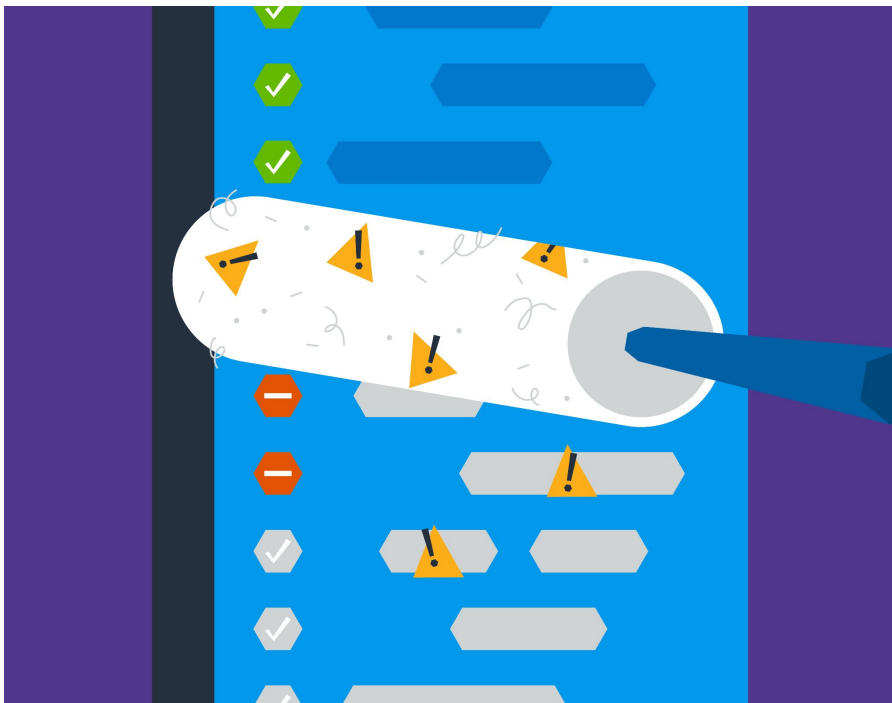
```

```

4 CSS Linting

 **Clean code = Easier to maintain**

- 1 **.editorconfig** – Unified indentation and spaces
- 2 **stylelint** – Ensures consistent CSS coding style
- 3 **prettier** – automatically formats code to follow a consistent style



```
npm run lint
> tutorialcode@1.0.0 lint
> eslint .

/Users/andrescourt/projects/alcbssystem/tutorialcode/dist/helpers/sumTwoNumbers.js
1:1  error  Strings must use singlequote      quotes
1:12  error  Extra semicolon                  semi
2:32  error  Strings must use singlequote      quotes
2:62  error  Extra semicolon                  semi
3:23  error  Missing space before function parentheses  space-before-function-paren
4:1  error  Expected indentation of 2 spaces but found 4  indent
4:17  error  Extra semicolon                  semi
6:32  error  Extra semicolon                  semi

/Users/andrescourt/projects/alcbssystem/tutorialcode/dist/index.js
1:1  error  Strings must use singlequote      quotes
1:13  error  Extra semicolon                  semi
2:1  warning  Unexpected var, use let or const instead  no-var
3:1  error  Expected indentation of 2 spaces but found 4  indent
3:46  error  Unnecessarily quoted property 'default' found  quote-props
3:46  error  Strings must use singlequote      quotes
3:62  error  Extra semicolon                  semi
4:2  error  Extra semicolon                  semi
5:32  error  Strings must use singlequote      quotes
5:62  error  Extra semicolon                  semi
6:7  error  Identifier 'express_1' is not in camel case  camelcase
6:43  error  Strings must use singlequote      quotes
6:54  error  Extra semicolon                  semi
7:17  error  Identifier 'express_1' is not in camel case  camelcase
7:27  error  Extra semicolon                  semi
9:1  error  Expected indentation of 2 spaces but found 4  indent
9:21  error  Strings must use singlequote      quotes
9:36  error  Extra semicolon                  semi
10:3  error  Extra semicolon                  semi
12:1  error  Expected indentation of 2 spaces but found 4  indent
12:55  error  Extra semicolon                  semi
13:3  error  Extra semicolon                  semi

/Users/andrescourt/projects/alcbssystem/tutorialcode/jest.config.js
4:26  error  Unexpected trailing comma      comma-dangle
5:2  error  Extra semicolon                  semi
5:3  error  Newline required at end of file but not found  eol-last

/Users/andrescourt/projects/alcbssystem/tutorialcode/src/helpers/sumTwoNumbers.ts
1:38  error  Missing space before function parentheses  @typescript-eslint/space-before-function-paren
2:1  error  Expected indentation of 2 spaces but found 4  @typescript-eslint/indent
2:23  error  Extra semicolon                  @typescript-eslint/semi
3:2  error  Newline required at end of file but not found  eol-last

/Users/andrescourt/projects/alcbssystem/tutorialcode/src/index.ts
1:1  error  Imports "Express", "Request" and "Response" are only used as types  @typescript-eslint/consistent-type-imports
6:1  error  Expected indentation of 2 spaces but found 4  @typescript-eslint/indent
6:21  error  Strings must use singlequote                  @typescript-eslint/quotes
10:1  error  Expected indentation of 2 spaces but found 4  @typescript-eslint/indent
```

.editorconfig

 **A configuration file that defines basic code formatting rules**

Ensures that all developers on a project use the same indentation, encoding, and line endings.

 **Defines coding style rules for text editors and IDEs**

 **Focuses on core formatting settings:**

- Spaces/tabs
- Indentation
- Encoding
- Line ending style

 **Does not check syntax or code structure, only formatting!** 

```
root = true
```

```
[*]
```

```
indent_style = space
```

```
indent_size = 2
```

```
end_of_line = lf
```

```
insert_final_newline = false
```

```
trim_trailing_whitespace = true
```

```
charset = utf-8
```

Ensures that all files in the project **use the same type of line breaks**, regardless of the developer's OS.

If different developers use different operating systems, Git might **show unnecessary changes** due to line-ending differences.

Prevents adding a **blank line at the end of files**.

Some editors **automatically add an empty newline at the end of a file**; this setting disables that behavior.

Automatically removes spaces or tabs at the end of each line when saving the file.

Trailing whitespace is unnecessary and can cause **Git diffs** to show unwanted changes.

stylelint

 **Ensures consistent CSS coding style** 

✅ **Checks CSS/SCSS** code for errors and enforces coding style rules.

✅ **Prevents common mistakes** (e.g., unknown properties, missing quotes).

✅ **Ensures the use of BEM** naming convention for class names.

◆ **Keeps your styles clean and maintainable!** 

Example `stylelint.config.js`:

```
module.exports = {
  extends: "stylelint-config-standard",
  rules: {
    "indentation": 2,
    "color-hex-case": "lower",
    "string-quotes": "double",
    "no-empty-source": true
  }
};
```

prettier

What does it do?

- Automatically **formats code** to follow a consistent style.
- Works with **CSS, SCSS, JavaScript, JSON, and more.**
- **Fixes indentation, line breaks, spacing, and quotes.**

Your code

body {
 margin: 0; padding: 0;
background: #fafafa;
}

h1 {
 color: #145cf2 ; font-weight : 300;
font-family: Roboto, sans-serif;
}

Beautify

Preview

body {
 margin: 0;
 padding: 0;
 background: #fafafa;
}

h1 {
 color: #145cf2;
 font-weight: 300;
 font-family: Roboto, sans-serif;
}

Copy

💡 Why use **prettier**?

- ✓ **Formats code automatically** so you don't have to.
- ✓ **Works with multiple languages** (JavaScript, CSS, JSON, etc.).
- ✓ **Prevents unnecessary code diffs** in Git (e.g., inconsistent spacing).


```
{  
  "tabWidth": 2,  
  "useTabs": false,  
  "endOfLine": "lf"  
}
```

Uses **2 spaces per indent**
(same as `.editorconfig`).

Always uses **spaces instead of tabs**.

Uses **LF (`\n`) line endings**
(like `.editorconfig`).

The Key Differences Between These Tools

Feature	<code>.editorconfig</code>	<code>stylelint</code>	<code>prettier</code>
Controls formatting in the editor	✓ Yes	✗ No	✗ No
Linting & error checking	✗ No	✓ Yes	✗ No
Automatic code formatting	✗ No	✗ No	✓ Yes
Works with	All file types	CSS & SCSS	JS, CSS, JSON, etc.
Example rules	Indentation, line endings	Class naming, color rules	Line breaks, spacing

Break time




Node.js

Node.js



– JavaScript Runtime
Outside the Browser

 **Why do we need it?**

-  **Allows running tools**, such as bundlers
-  **Manages dependencies** in a project
-  **Works with packages**

NPM

Node Package Manager is a package manager for JavaScript.

📌 **Main functions of npm:**

- ✅ **Installs and manages libraries and tools**
- ✅ **Works through the command line (terminal)**
- ✅ **Uses `package.json` to store project**

dependencies

💡 **npm comes bundled with Node.js** – if you've installed Node.js, you already have npm!



package.json

 The **package.json** file stores project commands and dependencies.

✓ Example of **package.json**:

```
{
  "name": "binabox-build-gulp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",

  "scripts": {
    "start": "NODE_ENV=production
gulp build",
    "dev": "gulp serve --lint"
  },
}
```

Let's do it

**create new
branch**

How to Install Node.js?

- 1 **Simple method:** Download from nodejs.org
- 2 **Best method:** Use NVM (*Node Version Manager*)

```
nvm install --lts
```

✓ **Check the installed version:**

```
node -v
```

"Node.js is installed. Now we can set up a bundler!" 🚀

Open your project and start

npm ci – install dependencies



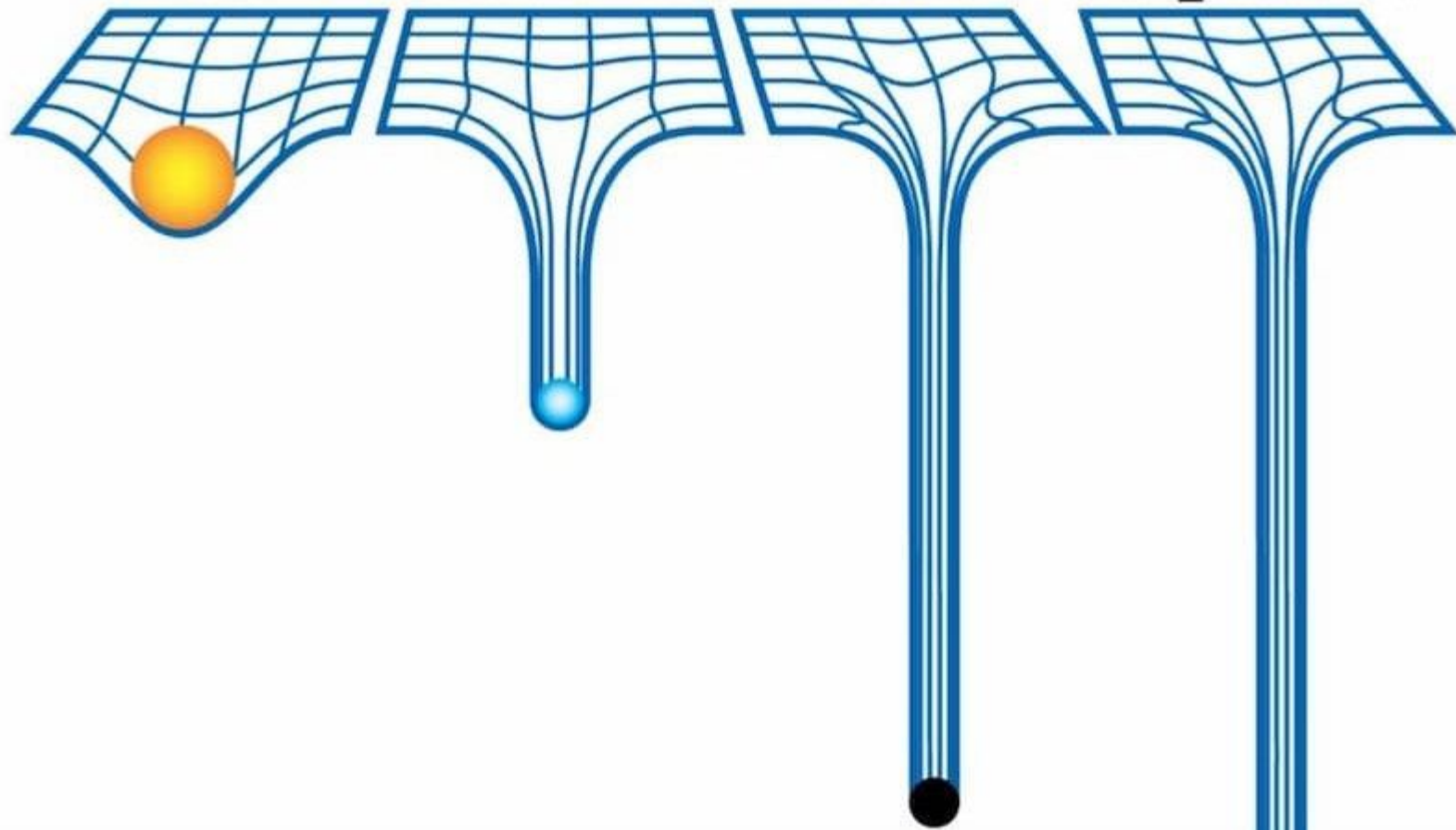
node_modules – creates

Sun

Neutron star

Black hole

node_modules



.gitignore

.DS_Store

Thumbs.db

***.log**

node_modules/ – too heavy

public/ – version for prod

Build commands for start

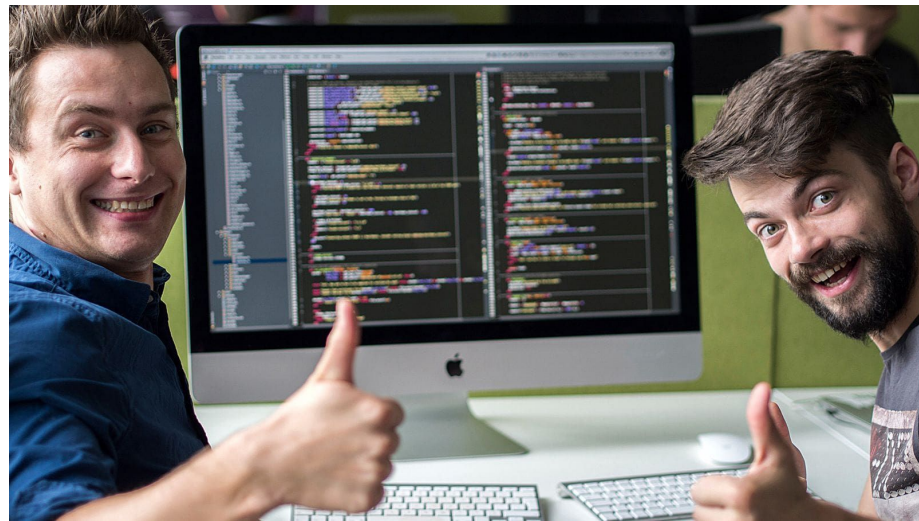
npm start – will generate site (static build)

npm run dev – will generate site, show **lint errors**,
watch files and **re-run build on change(live server)**

ctrl + c – stop watch

Summary – Now We Know:

- ✓ What a bundler is and why we need it
- ✓ What is Gulp
- ✓ How to install and use them
- ✓ How to automate work with your project



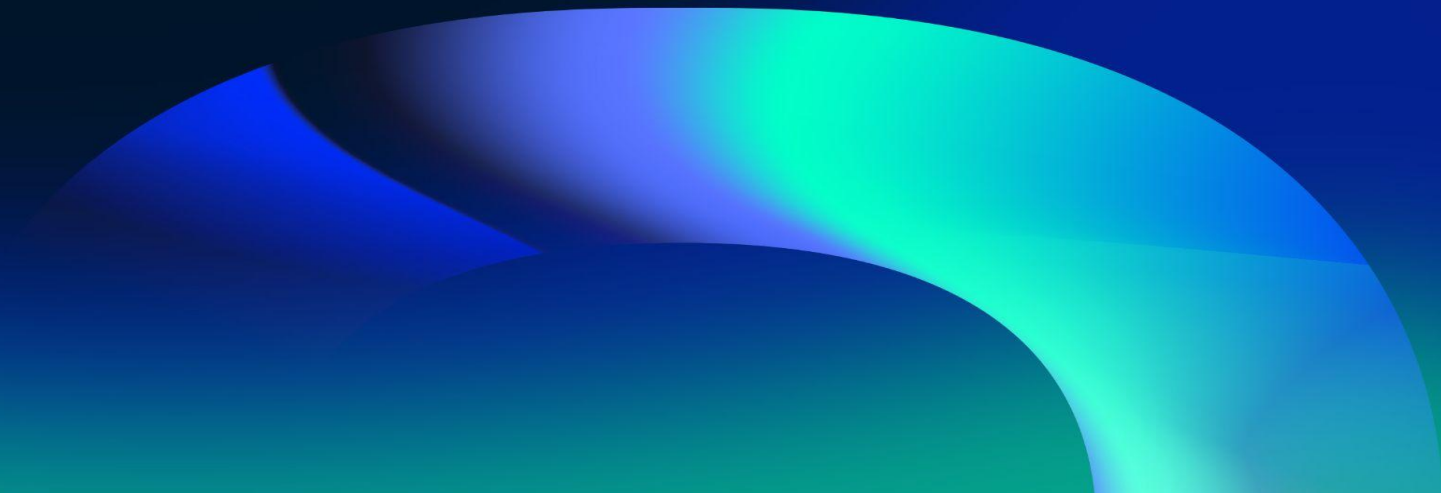
Code easier, faster, and more efficiently 🚀

B Academy
RO



QUESTIONS?

Please fill out the feedback form
It's very important for us





THANK YOU!

Have a good evening!