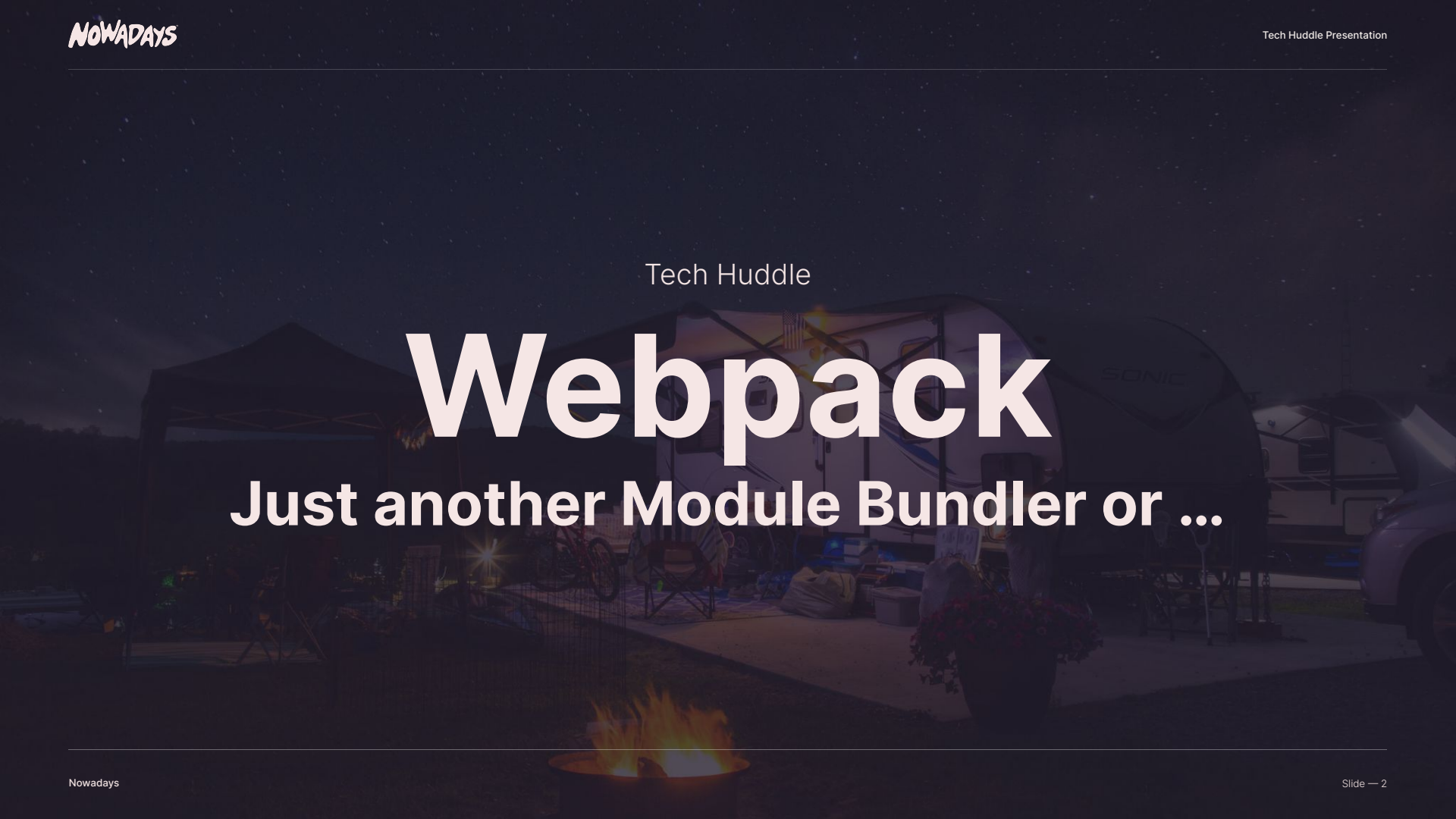

# Webpack

2022/06/10



Tech Huddle

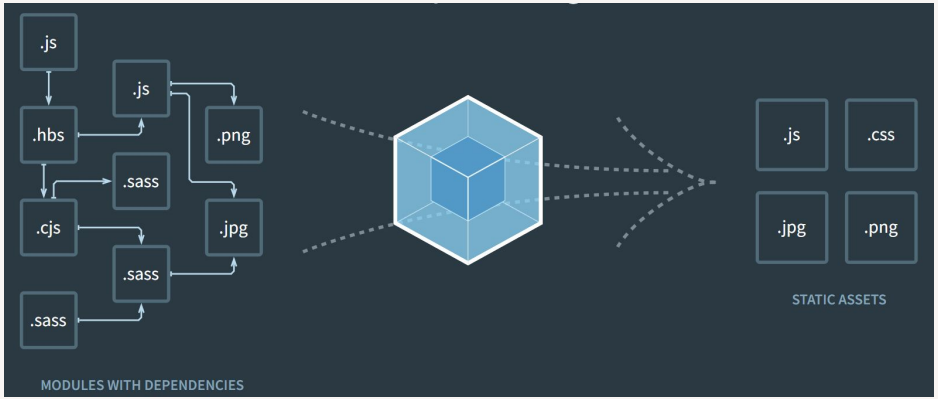
# Webpack

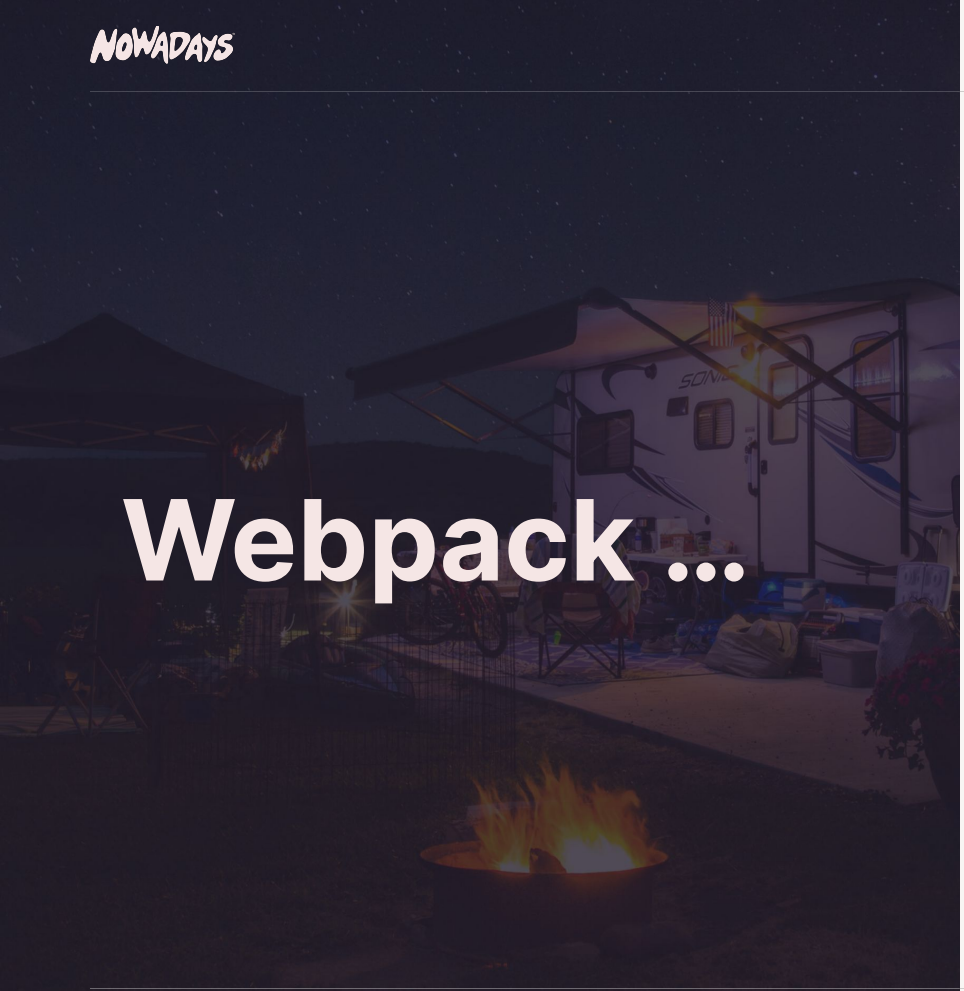
## Just another Module Bundler or ...

Tech Talk

# Focus

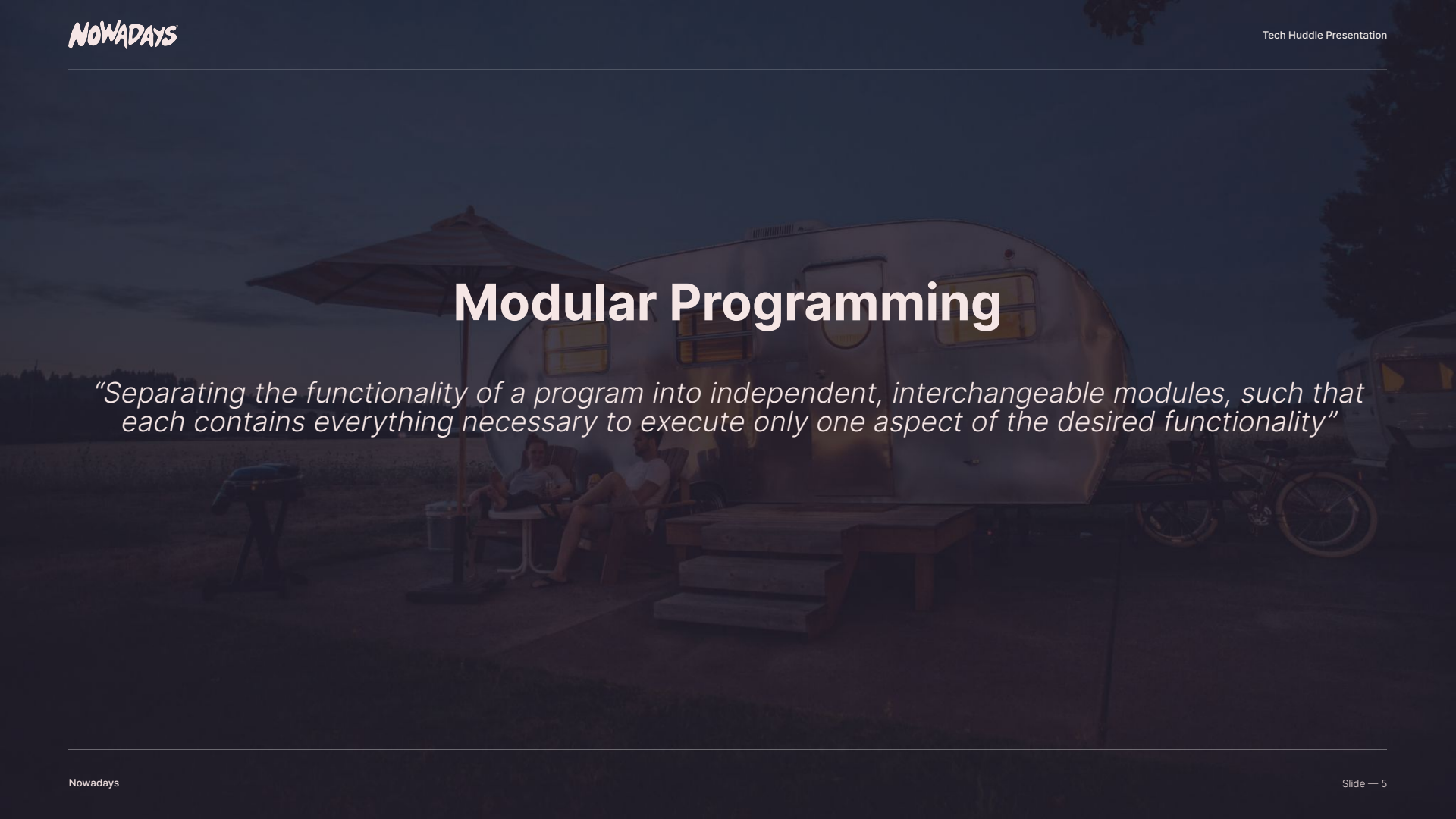
What is Webpack?
The role it plays and how can we harness its “power” for good?
And some other small tricks to make life easier!





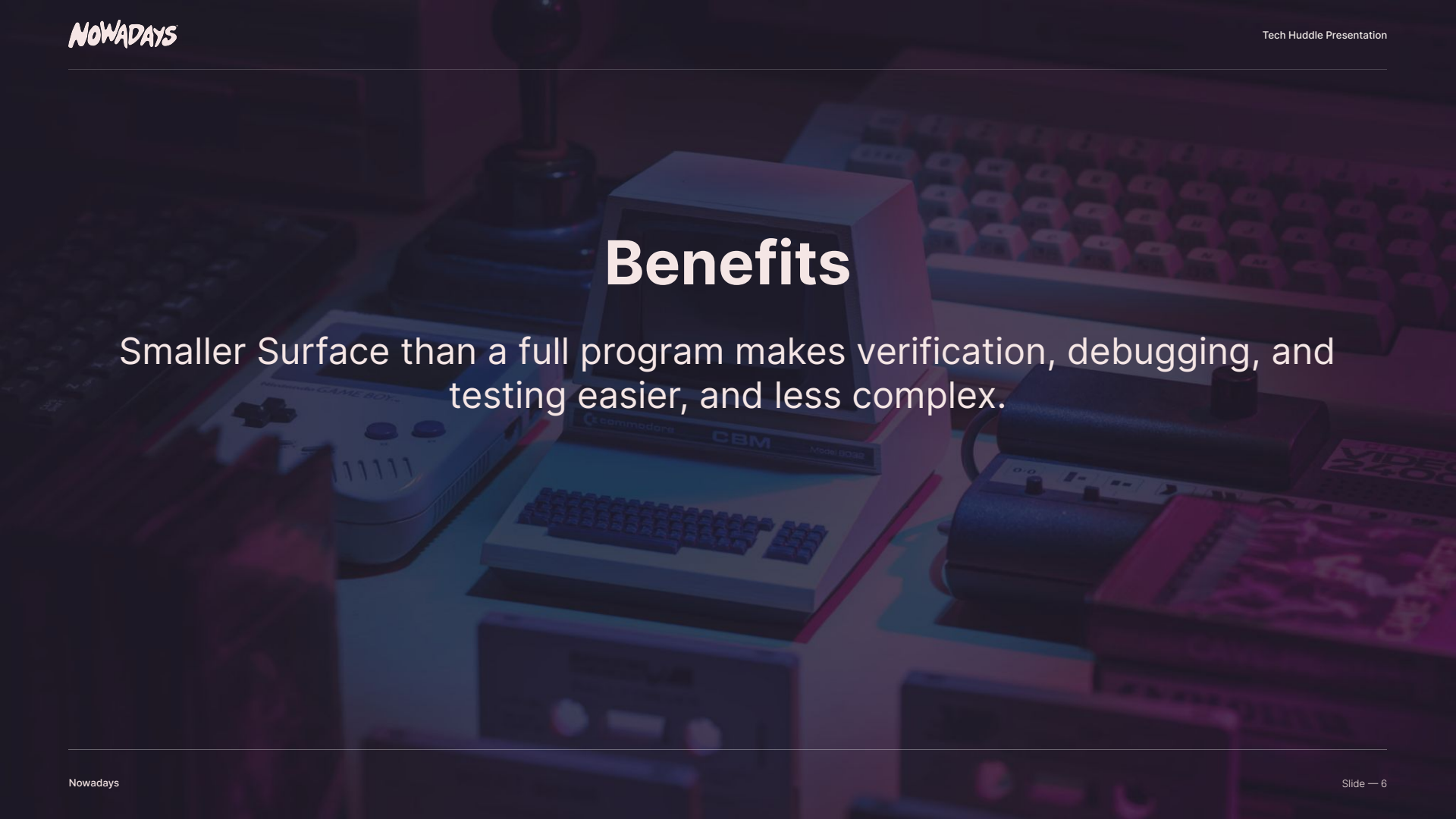
# Webpack ...

- ✔ ... is a Static Module Bundler
- ✔ ... serves Modern JavaScript applications
- ✔ ... supports the Modular Programming Software Design technique
- ✔ ... builds a Dependency Graph from an entry point
- ✔ ... bundles needed modules into static assets



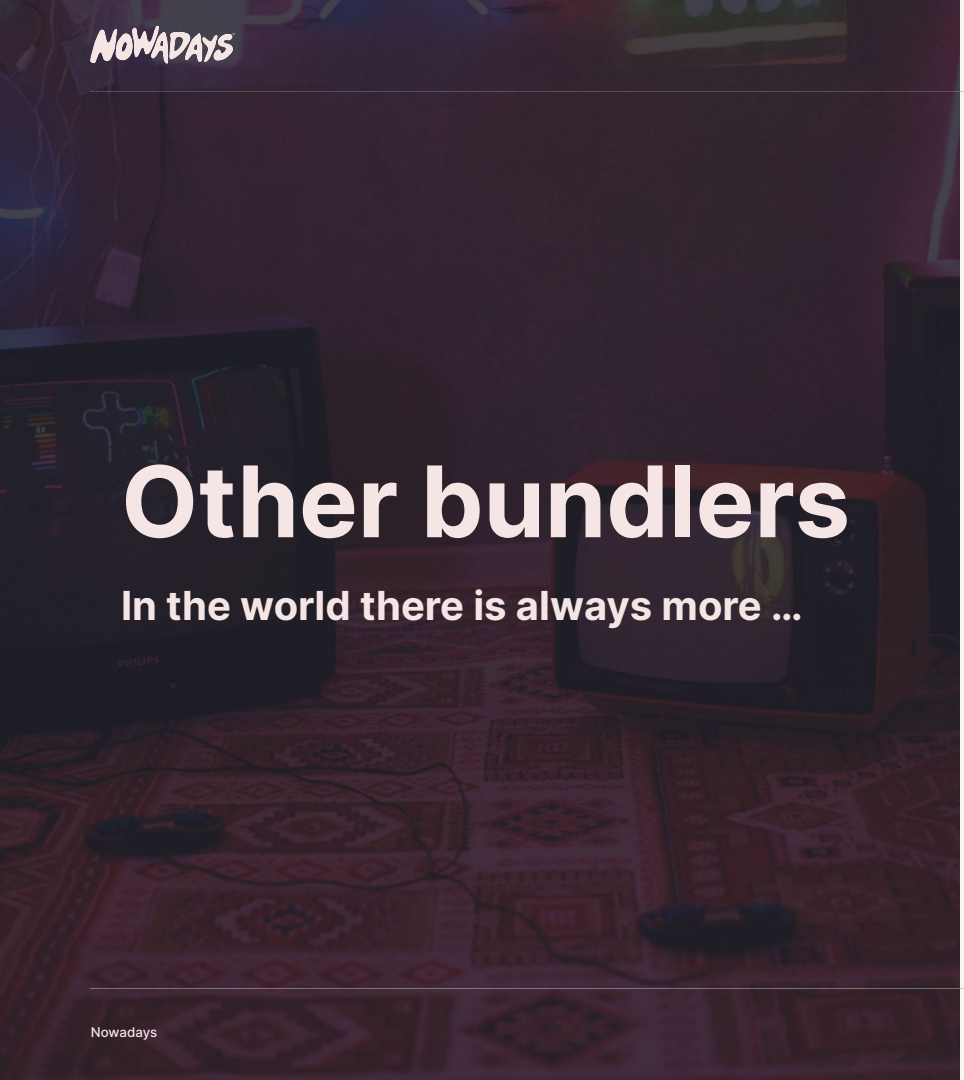
# Modular Programming

*“Separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality”*



# Benefits

Smaller Surface than a full program makes verification, debugging, and testing easier, and less complex.



# Other bundlers

## In the world there is always more ...

- ✔ Vite
- ✔ Browserify
- ✔ FuseBox
- ✔ Rollup
- ✔ Parcel
- ✔ ...

# Webpack modules

Webpack supports ...

- ✔ ECMAScript modules (ESM)
- ✔ CommonJS modules
- ✔ AMD modules
- ✔ Assets
- ✔ WebAssembly modules

# Webpack modules

Express dependencies using …

- ✔ An ES2015 **import** statement
- ✔ A CommonJS **require()** statement
- ✔ An AMD **define** and **require** statement
- ✔ An **@import** statement inside of a css/sass/less file
- ✔ An image url in a stylesheet url(...) or HTML <img src=...>

Webpack

# Core Concepts

These are the core concept worth knowing ...

- ✔ Entry
- ✔ Output
- ✔ Loaders
- ✔ Plugins

There are two more that we will skip for now ...

- ✔ Mode
- ✔ Browser Compatibility

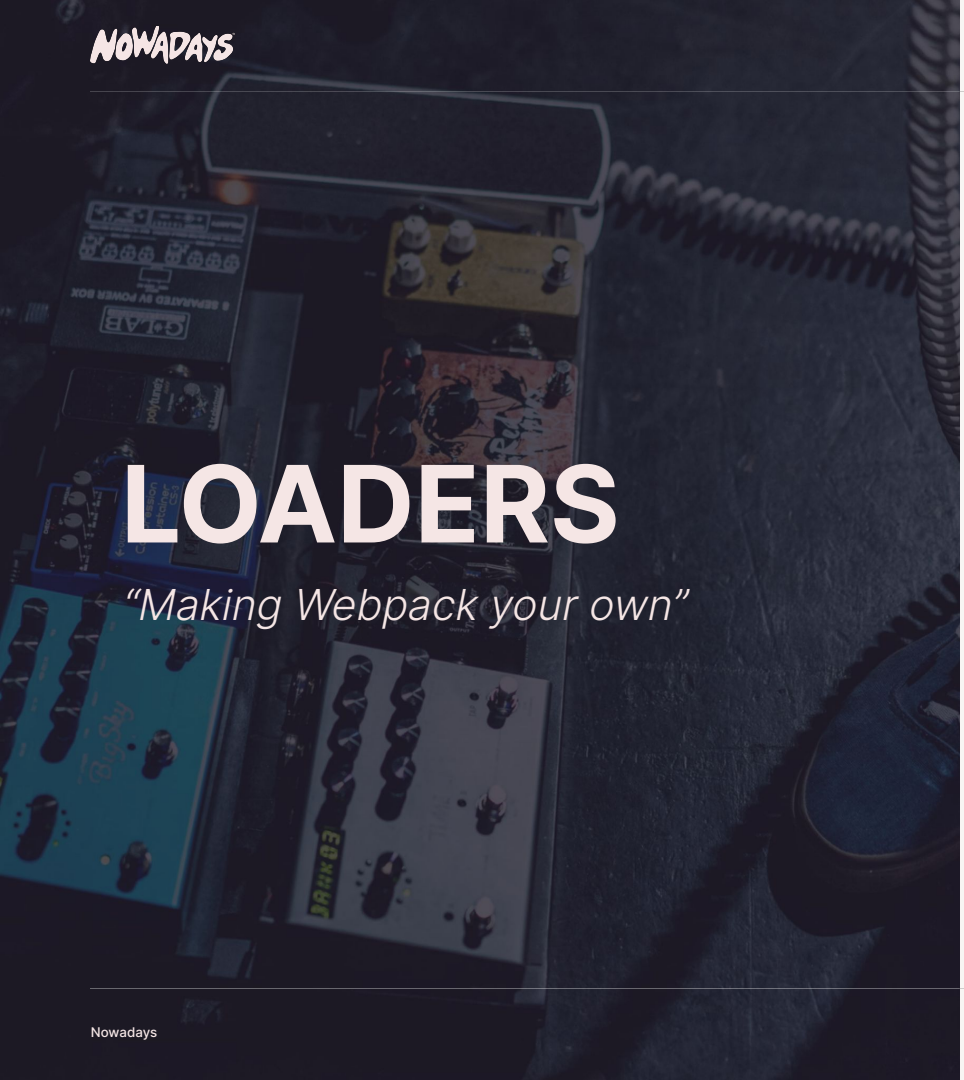
**Entry**
An entry point indicates which module webpack should use to begin building out its internal dependency graph

**Output**
An entry point indicates which module webpack should use to begin building out its internal dependency graph

**Loaders**
Out of the box, webpack only understands JavaScript and JSON files. Loaders allow webpack to process other types of files and convert them into valid modules that can be consumed by your application and added to the dependency graph

**Plugins**
While loaders are used to transform certain types of modules, plugins can be leveraged to perform a wider range of tasks like bundle optimization, asset management and injection of environment variables

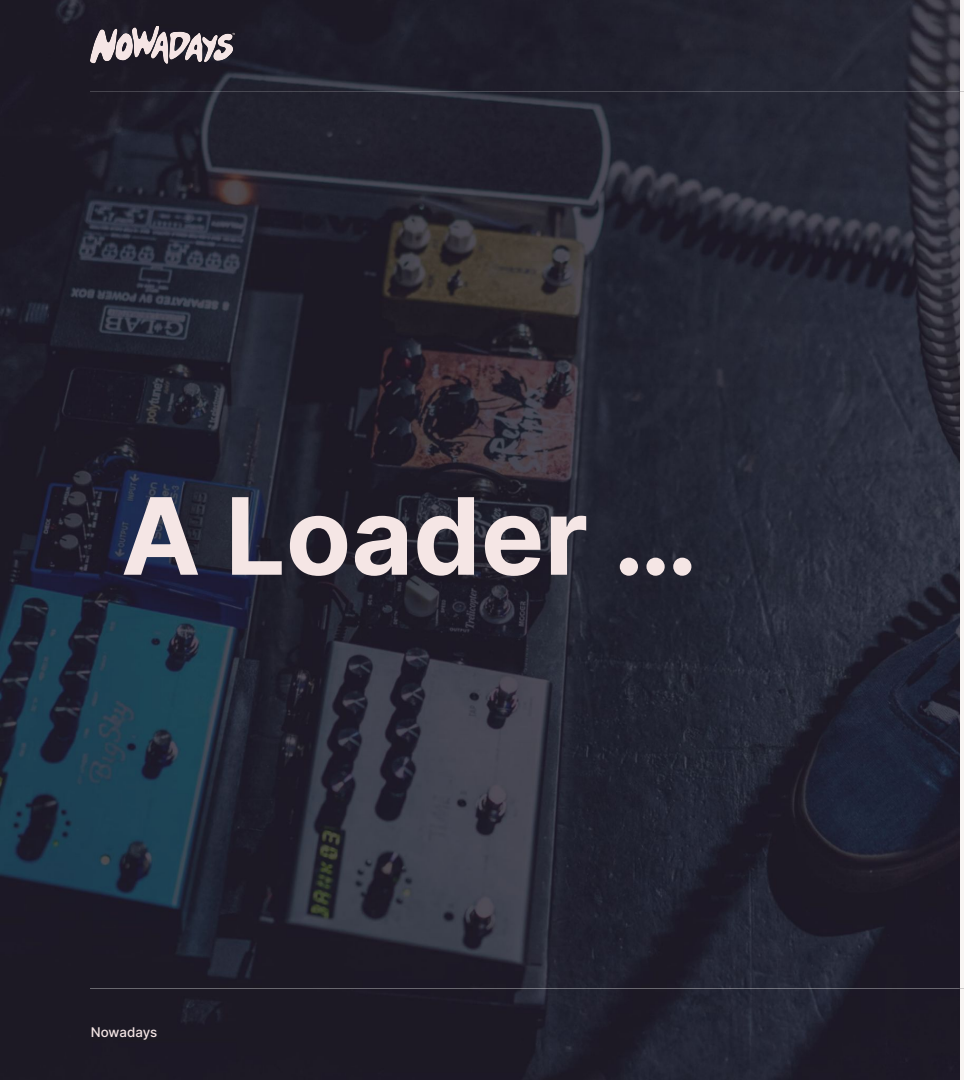
# LOADERS



# LOADERS

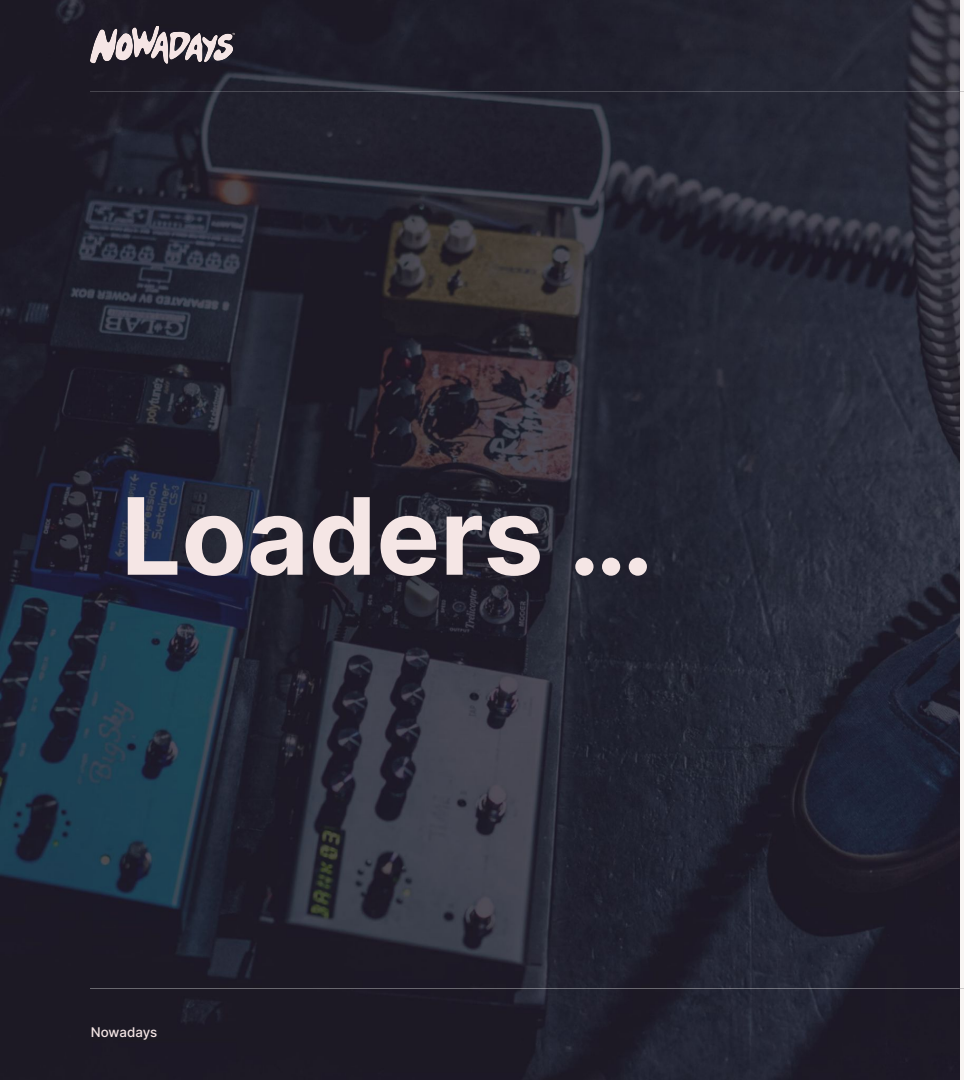
*“Making Webpack your own”*

- ✔ **Transformations applied to the source code of a module**
- ✔ **Allow to pre-process files when “import” or “load” them**
- ✔ **Work as Tasks in other build tools**
- ✔ **Handles front-end build steps**



# A Loader ...

- ... is a Node module that exports a function
- ... has access to the Loader API through “this” context
- ... takes a single parameter that is passed a string containing the content of the resource file
- ... can be synchronous that will return a single value representing the transformed module



# Loaders ...

- ✔ ... can be chained.
- ✔ ... are executed in reverse order in a chain
- ✔ ... results are passed as an argument to the next in the chain
- ✔ ... last in a chain is expected to return JavaScript
- ✔ ... can be synchronous or asynchronous
- ✔ ... runs in Node.js
- ✔ ... can emit additional arbitrary files

# Loaders are executed individually

# PLUGINS

# Plugins ...

- ✔ ... are the backbone of Webpack
- ✔ ... system is used internally in Webpack
- ✔ ... do everything else that Loaders cannot do
- ✔ ... are added and configured in webpack.config.js
- ✔ ... can also be added through the Node API

# A Plugin ...

- ✔ ... is a JavaScript Object
- ✔ ... object has one method, apply
- ✔ ... is initialized by Webpack by calling the apply method in the initialization
- ✔ ... has access to the entire compilation lifecycle

“While loaders are used to transform certain types of modules, plugins can be leveraged to perform a wider range of tasks like bundle optimization, asset management and injection of environment variables.”

Webpack

# Webpack provides a lot out of the box

https://webpack.js.org/plugins
https://webpack.js.org/awesome-webpack/#webpack-plugins

# How to Hook into Webpack

# Plugin API

Provides access to all parts of the Webpack lifecycle through tapable hooks.

# Tapable

Small package built by Webpack

Many objects in Webpack extends the Tapable class

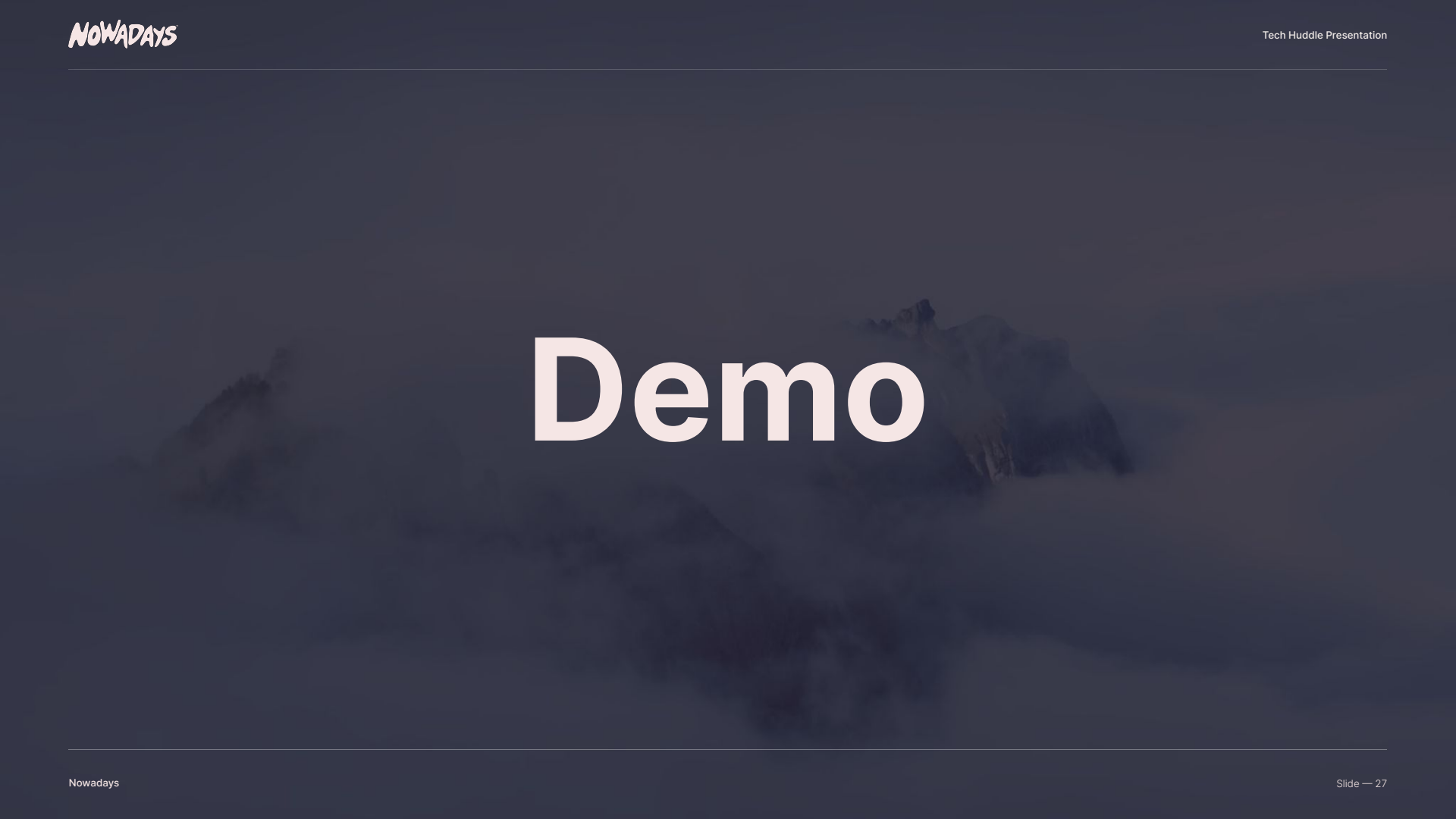
Tapable expose Hook classes

Through “tap” you can attach a hook to different parts of Webpacks lifecycle

# Tapables

- ✔ SyncHook,
- ✔ SyncBailHook,
- ✔ SyncWaterfallHook,
- ✔ SyncLoopHook,
- ✔ AsyncParallelHook,
- ✔ AsyncParallelBailHook,
- ✔ AsyncSeriesHook,
- ✔ AsyncSeriesBailHook,
- ✔ AsyncSeriesWaterfallHook


NOWADAYS
Reporting Progress
To report progress, a plugin must tap into a hook using the context: true option.
It is possible to customize the printed output by passing different arguments to the ProgressPlugin.
reportProgress
--progress argument when running the webpack CLI.
Plugins can report progress via ProgressPlugin, which prints progress messages to stderr by default. In order to enable progress reporting, pass a
Hooks
Compiler Hooks
...
...
rebuildModule
...
NormalModuleFactory Hooks
Custom Hooks
The moral of the story is that there are


Demo

# Why Scaffolding?

NOWADAYS

Tack så mycket!