**TUTORIAL FRONTEND**

**WEBPACK**

**Agenda**

1. **What is Webpack?**
2. **Should I Use Webpack?**
3. **Dependency Graph**
4. **What Does Webpack Actually Do?**
5. **The Good**
6. **Get Started with WEBPACK**

**Let’s GO**

1. **What is Webpack**

Webpack is a **build tool that puts all of your assets, including Javascript, images, fonts, and CSS, in a dependency graph.**

Webpack lets you use require() in your source code to point to local files, like images, and decide how they're processed in your final Javascript bundle, like replacing the path with a URL pointing to a CDN.

1. **Should I Use Webpack**

If you're building a complex Front End application with many **non-code static assets** such as CSS, images, fonts, etc, then **yes, Webpack will give you great benefits.**

If your application is fairly small, and you don't have many static assets and you only need to build one Javascript file to serve to the client, then **Webpack might be more overhead than you need.**

1. **Dependency Graph:**

In the early days, we "managed" JavaScript dependencies by including files in a specific order:

<script src="jquery.min.js"></script>

<script src="jquery.some.plugin.js"></script>

<script src="main.js"></script>

This was too slow because of excess HTTP requests. **Or**

// build-script.js

var scripts = [

'jquery.min.js',

'jquery.some.plugin.js',

'main.js'

].concat().uglify().writeTo('bundle.js');

// Everything our app needs!

<script src="bundle.js"></script>

This still **relied on the order of concatenated files.** Even worse, the code could only communicate through global variables! The first script declared a global jQuery variable, then jquery.some.plugin.js created a new global, or modified the global jQuery object

And with **WEBPACK**

We use CommonJS or ES6 modules to put our Javascript in a true **dependency graph.** We make small files that explicitly describe what they need.

// Version.js

module.exports = { version: 1.0 };

// App.js

var config = require('./Version.js');

console.log('App Version:', config.version);

1. **What Does Webpack Actually Do?**

Webpack lets you use require() on **local "static assets,"** meaning non-code files.

<img src={ require('../../assets/logo.png') } />

When you run Webpack, it searches through all of your code for require() calls. It compares the path string ../../assets/logo.png to the **"loader" configuration** you specify.

In this example, when you require() file paths ending in .png (matching the above regular expression), Webpack sends that file to the **file loader**.

The file loader does two things.

+ In the bundled Javascript code, it replaces the require() call with a URL string, making it valid Javascript. The string depends on how you configure Webpack.

+ The file loader also spits out logo.png into some local folder you specify, like dist/.

**Key concept:** The require('logo.png') source code never actually gets executed in the browser (nor in Node.js). Webpack builds a new Javascript file, replacing require() calls with valid Javascript code, such as URLs. The bundled file is what's executed by Node or the browser.

## What About Browserify, Grunt, Gulp…?

Webpack puts your static assets (and source code) in a true dependency graph. **Grunt and Gulp** are only tools for working with files, and have no concept of a depdency graph.

[Browserify](http://browserify.org/) is mainly a tool to transform require() calls that work in Node.js into calls that work in the browser. It's a dependency graph for your source code only

Webpack's core idea of a dependency graph is what makes it so powerful and useful.

1. **The Good**

* **Dead asset elimination.**
* **Easier code splitting**
* **You control how assets are processed**
* **Stable production deploys.**  You can't accidentally deploy code with images missing, or outdated styles.
* Webpack will slow you down at the start, but give you **great speed benefits** when used correctly. You get hot page reloading
* Webpack is the **main build tool adopted by the React community**

Webpack comes with a built in "[dev server](https://webpack.github.io/docs/webpack-dev-server.html)"; a small [express](https://www.npmjs.com/package/express) app for local development. You simply include one Javascript tag pointed to the server, like localhost:8080/assets/bundle.js, and get live code updating and asset management for free.

1. **Let’s get Started with WEBPACK**