JAVA SCRIPT MODULE BUNDLER

What is Bundler?

A bundler is a development tool used to combine multiple JavaScript code files into a single file that is optimized and production-ready, making it loadable in the browser. Bundlers help manage dependencies, reduce the number of HTTP requests, and improve performance

Purpose: Bundlers reduce the number of HTTP requests and improve performance by consolidating code and dependencies.

Key Features:
Combines multiple JavaScript files.
Manages dependencies.
Optimizes code for production.

Popular Bundlers:

- Webpack
- Parcel
- Rollup
- Browserify

Parcel: Parcel is a zero-configuration bundler, which means it requires no configuration file to start bundling. It automatically handles various types of files out of the box, allowing developers to focus more on writing code and less on configuring their build tools.

Purpose:
Bundles JavaScript files.
Requires no configuration.

Key Features:
Zero-configuration setup.
Supports various types of assets out of the box.
Fast and reliable.

How does it operate?

- **Asset Tree construction:** Using an entry point asset as a starting point, Parcel iterates through the file to find the dependencies needed to build an asset tree that resembles the dependency graph.
- **Bundel Tree Construction:** Building a bundle tree involves combining individual items from the asset tree with the dependencies that link them together.
- **Packaging:** This last step involves associating each bundle in the bundle tree with the appropriate packager file type and converting it into a final compiled file.

You can then use a single entry asset against Parcel after that. Remember that Parcel allows for several points of entry.

To get started, run npm i parcel.

Let’s say you have a sample HTML boilerplate.

```
<html>
  <body>
    <script src="./index.js"></script>
  </body>
</html>
```

Run parcel index.html to begin building the HTML file using Parcel. Impressively, Parcel will compile both the index.js that the HTML links to and the HTML file that is linked to it.

Example Usage:
parcel build index.html.

Rollup: Rollup is a module bundler for JavaScript which compiles small pieces of code into something larger and more complex, such as a library or application. It differs from other tools in that it uses the new standardized format for code modules included in the ES6 revision of JavaScript, instead of using CommonJS or AMD like its alternatives.

Purpose:
Bundles JavaScript files.
Optimizes code for production.

Key Features:
Uses the ES module format.
Tree-shaking for dead code elimination.
Faster build times.

How does it operate?

Rollup arranges dependencies, does tree shaking, and examines the entry point file. It is customized using rollup.config.js. It creates an efficient bundle by combining specified functions into a single global scope and guaranteeing that there are no name collisions.

To get started, run npm i rollup to install rollup. You can perform the bundling process either through a CLI with the aid of a config file or via the bundling JavaScript API.

Here’s a sample config file containing the entry point, output file destination, and format type.

```
export default {  input: 'src/app.js',  output: {  file: 'bundle.js',  format:
```

Browserify: A JavaScript bundler called Browserify is especially helpful for projects that make use of the CommonJS (a Node.js-style module system).

Purpose: Bundles JavaScript files.

Key Features: Converts Node-style modules (CommonJS) into a single bundle.

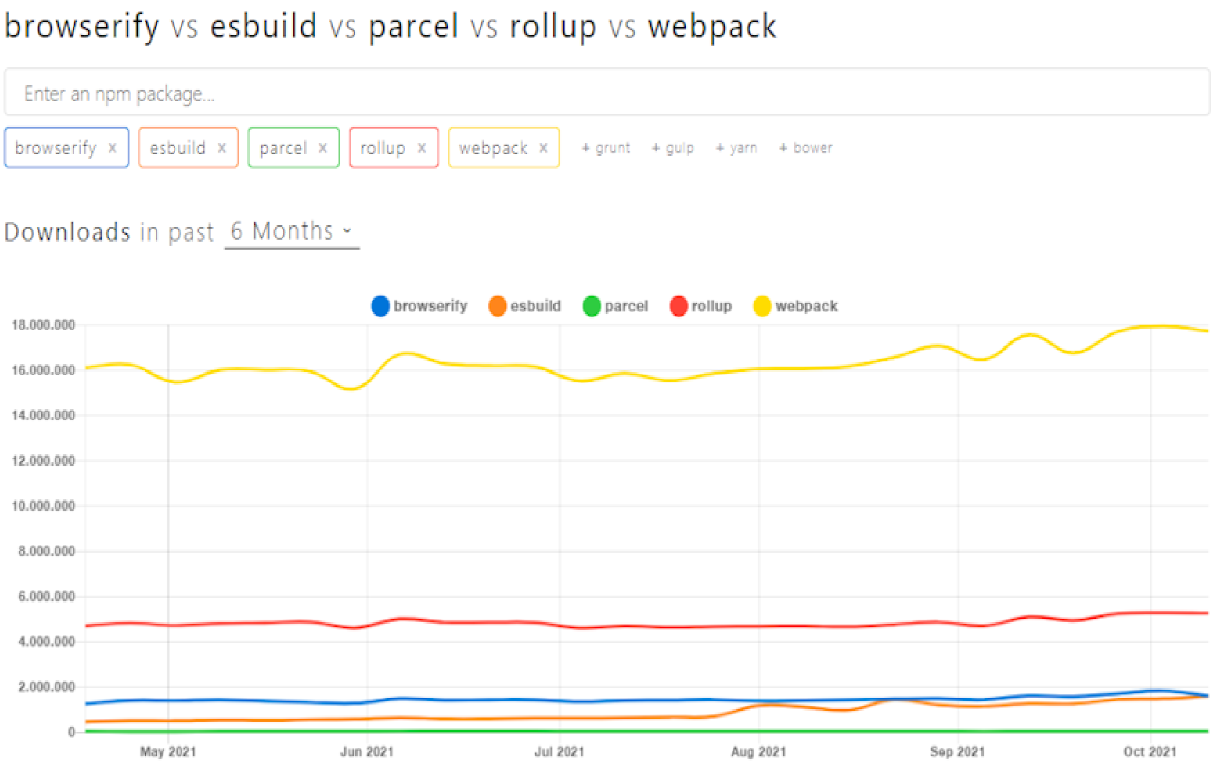
How dose it work?

When bundling modules, Browserify follows predetermined steps, just like all other JavaScript bundlers. The creation of the dependency graph comes first. At this point, Browserify examines your files recursively for all require() calls, starting from the entry point files you've selected. A file path is returned for each require() call, and additional require() calls are made by iteratively exploring each file path.
You can then place the final bundle in a single <script> for eventual browser loading. Getting started with Browserify is as simple as running npm i webpack and running Browserify against your entry file.

```
$ browserify main.js > bundle.js
```

The bundler also provides s ome inbuilt options like **--debug** and **--ignore-missing**.

Statistical View:


browserify vs esbuild vs parcel vs rollup vs webpack