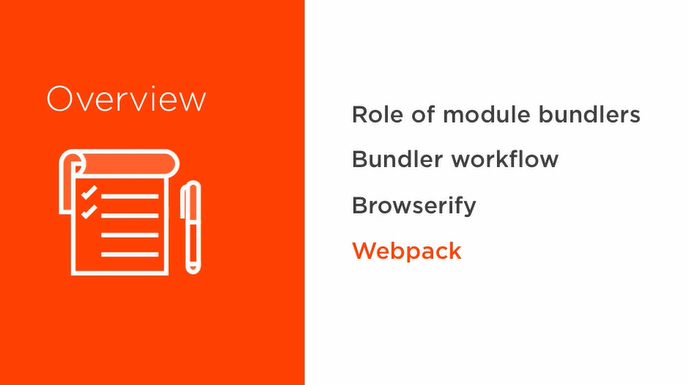
**Module Bundlers**

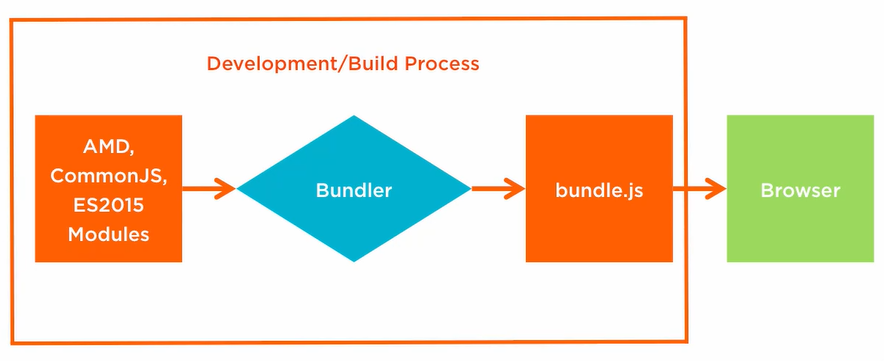
Module bundlers solve the same problem as module loaders (which load AMD + CommonJS modules in a browser), but they do it as a build step, rather than at runtime.



**The role of a module bundler**

* Do what loaders do *but as a build step, not at runtime*.
* Rather than downloading a dependency when needed at runtime, it just *adds them to the bundle in the right location and order*.
* Means far fewer files because modules used by multiple libraries aren’t redownloaded – they all point to the same instance of the code.
* Sometimes this can decrease start-up time.

**Module Bundler Workflow**



A very similar workflow to transpiling code with Babel. It’s a build step, and something that only happens once before the build is deployed. There may also be a transpilation step, particularly if your modules are in ES2015, in which case the modules are transpiled into AMD or CommonJS.

**Browserify**

* *Developed to make modules built for Node.js backends available to browser-based applications.*
* Because Node.js uses CommonJS, it bundles CommonJS modules.

