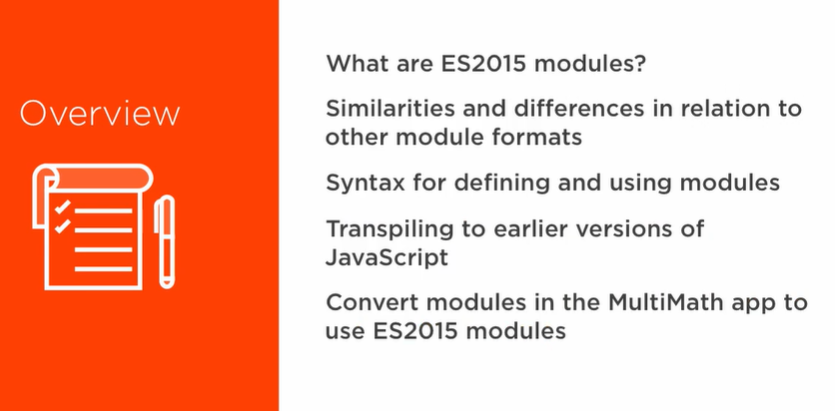
**ES2015 modules**



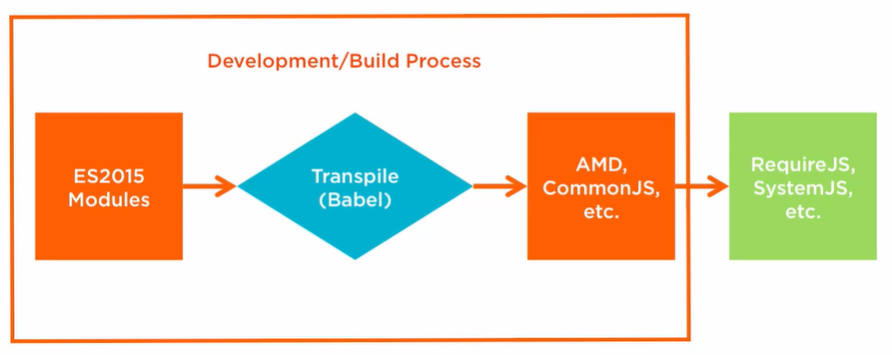
**What are ES2015 modules? How do we create and load them?**

Unlike AMD or CommonJS, they’re natively implemented modules, added well after the others were added by third parties. No loader required. Very similar to the others:

* encapsulate implementation details
* explicitly expose a public API
* support for dependency management

Note that we sometimes need to **transpile** (convert from one language to another) from the modern JS to an older version because older browsers can fail to support our new module syntax.

1. Write modern ES2015 modules.
2. Transpile those modules down to an old version (AMD, CommonJS, etc) with Babel.
3. Load those translated modules with RequireJS or SystemJS



**ES2015 Module imports:**

**Imported** items are dependencies.

* May import an entire module, or just part of it (say a sigle function).
* Can import an item under an alias – say it has a confusing name, or a name that might conflict with your own.

**Exporting** exposes the API of a module.

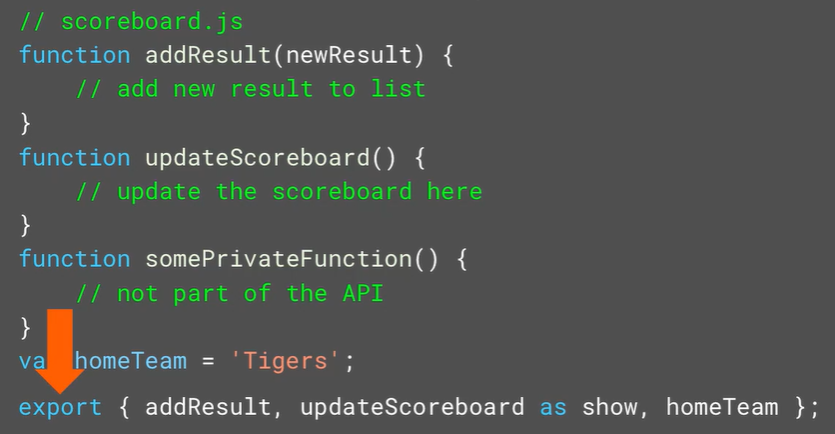
* Exposes the API of a module. Conceptually the same as the previous systems.
* Can either export the item at declaration or use it once at the end and pass it a list of things to export at the end.
* May specify a default export.

**Export Syntax**

*Line by line exporting*

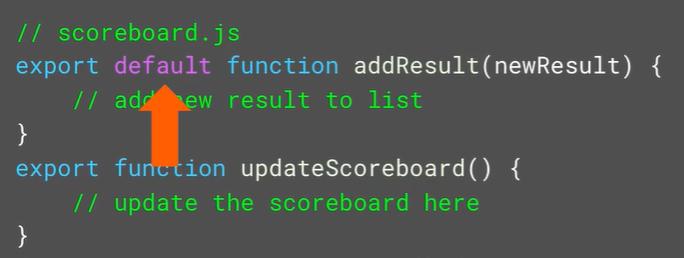
**

*End of document exporting*



* You can export things line by line – functions, variables, objects, everything 😊
* You can also export them all at the end. This is good when you have a big module, because you can very quickly and easily see what you’re making available.

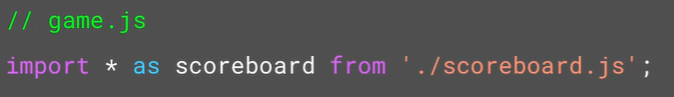
Default Exports



* You can also set one member as a default export. This lets people import the default without specifying the name. This is great if you only have one export in your module.

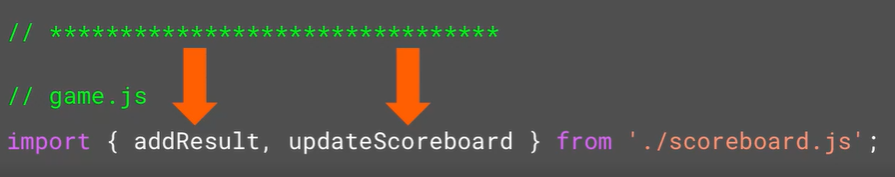
**Import Syntax**

*Simple module import*



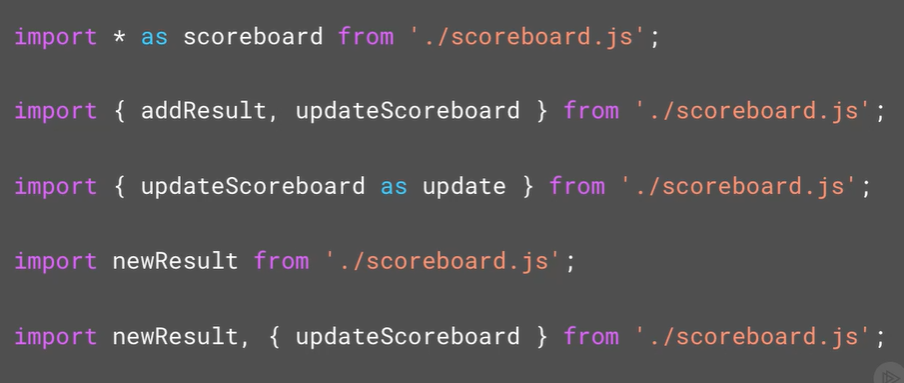
* Simple as cheese! Import the whole thing with star. Reference mbmers on that object by calling the module DOT imported component (scoreboard.createNewScoreboard()).

*Importing members*



* You can also import individual members. From here on out you don’t need to prefix it with a parent object like above, you can just call it directly.

*Other syntax*



* 1. Import an entire module
* 2. Import specific members by listing them
* 3. Import specific members and assign them an alias – similar to exporting syntax.
* 4. If your module has a default export, you can specify the name you want to give to the default module (newResult) – the member in the scoreboard module exported as default will be available now as newResult.
* 5. If you want to import the default export plus others, do the last option. You’re listing the name of the default export, then the additional members you want.