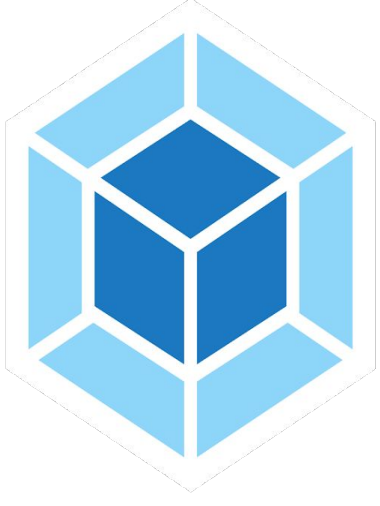


webpack ACADEMY

Mastering Plugins

https://github.com/thelarkinn/webpack-workshop-2018

https://frontendmasters.com/courses/webpack-plugins/

What is Tapable?

What is Tapable?

- ~200 line plugin library

What is Tapable?

- ~200 line plugin library
- The backbone of the plugin system


Compiler.js
458
459 Compiler.prototype.compile = function(callback) {
460 var self = this;
461 var params = self.newCompilationParams();
462 self.applyPluginsAsync("before-compile", params, function(err) {
463 if(err) return callback(err);
464
465 self.applyPlugins("compile", params);
466
467 var compilation = self.newCompilation(params);
468
469 self.applyPluginsParallel("make", compilation, function(err) {
470 if(err) return callback(err);
471
472 compilation.finish();
473
474 compilation.seal(function(err) {
475 if(err) return callback(err);
476
477 self.applyPluginsAsync("after-compile", compilation, function(err) {
478 if(err) return callback(err);
479
480 return callback(null, compilation);
481 });
482 });
483 });
484 });
485 };
486
compiler event
compiler event

```
class BasicPlugin {
    constructor() {}

    apply(compiler) {
        compiler.plugin("make", (compilation) => {
            console.log("I now have access to the compilation!!!!!");
        });
    }
}

module.exports = BasicPlugin;
```

What is Tapable?

```
class SomePlugin {
  ① apply(compiler){
      compiler.plugin("run", (c, cb)=>{


      });
  }
}
```

```
class Compiler extends Tapable {
  someFunct() {
    this.applyPluginsAsync("run", this, ()=>
  } );
```

① Plugin is registered via compilers inherited apply fn. (from Tapable)

What is Tapable?

```
class SomePlugin {
  ① apply(compiler){
      compiler.plugin("run", (c, cb)=>{

      });
  }
}
```

```
class Compiler extends Tapable {
  someFunct() {
    ② this.applyPluginsAsync("run", this, ()=>
    } );
```

① Plugin is registered via compilers inherited apply fn. (from Tapable)

② Tapable instances call "applyPlugins()" and pass the event and state through to plugin.

What is Tapable?

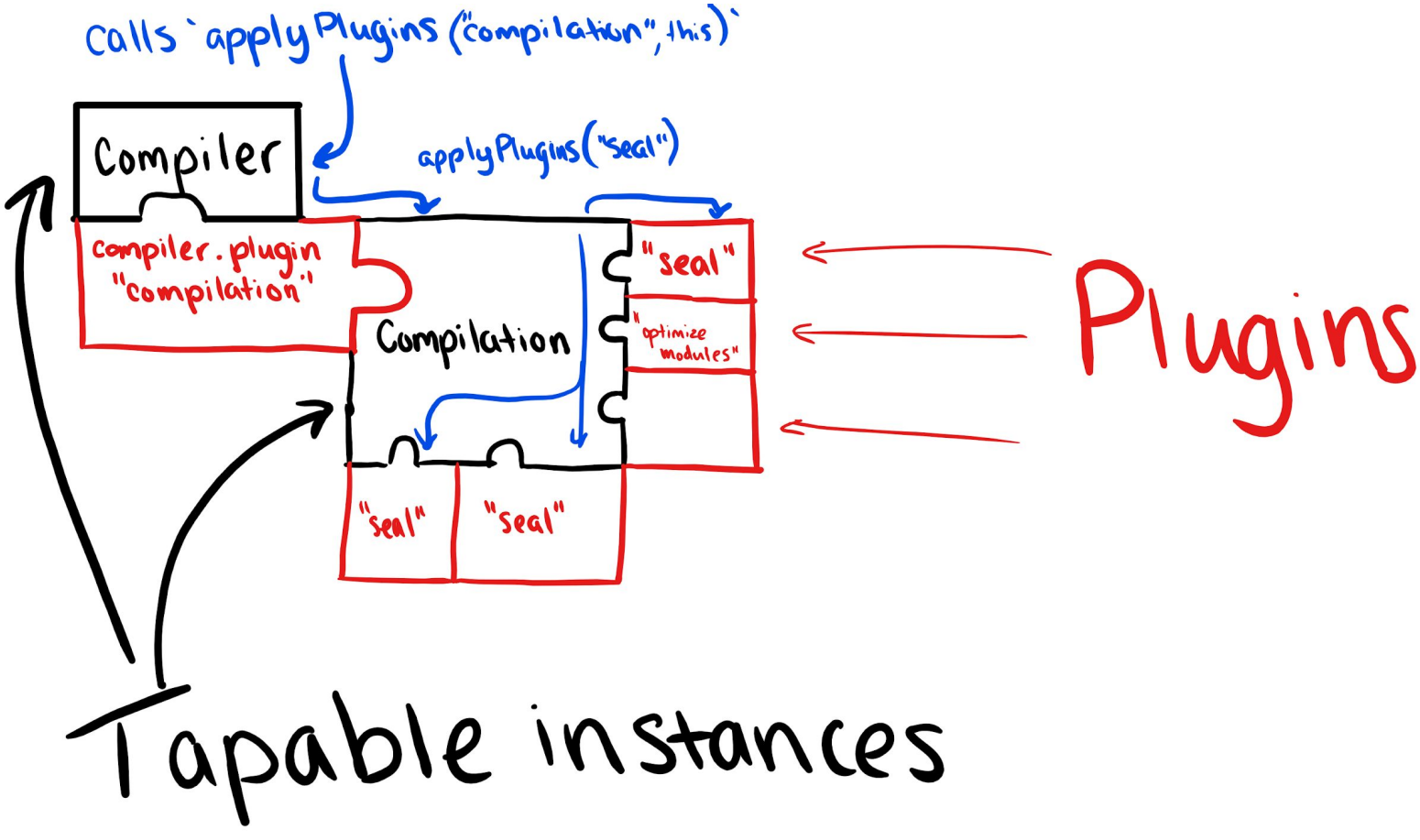
- Tapable instance

What is Tapable?

- Tapable instance

  A class/object that extends Tapable

  (aka something you can plug into!!)


Calls `applyPlugins("compilation", this)`
Compiler
applyPlugins("seal")
compiler.plugin
"compilation"
Compilation
"seal"
"optimize modules"
"seal"
"seal"
Plugins
Tapable instances

7ish Tapable Instances (aka classes)

Compiler!

7ish Tapable Instances (aka classes)

Compiler!

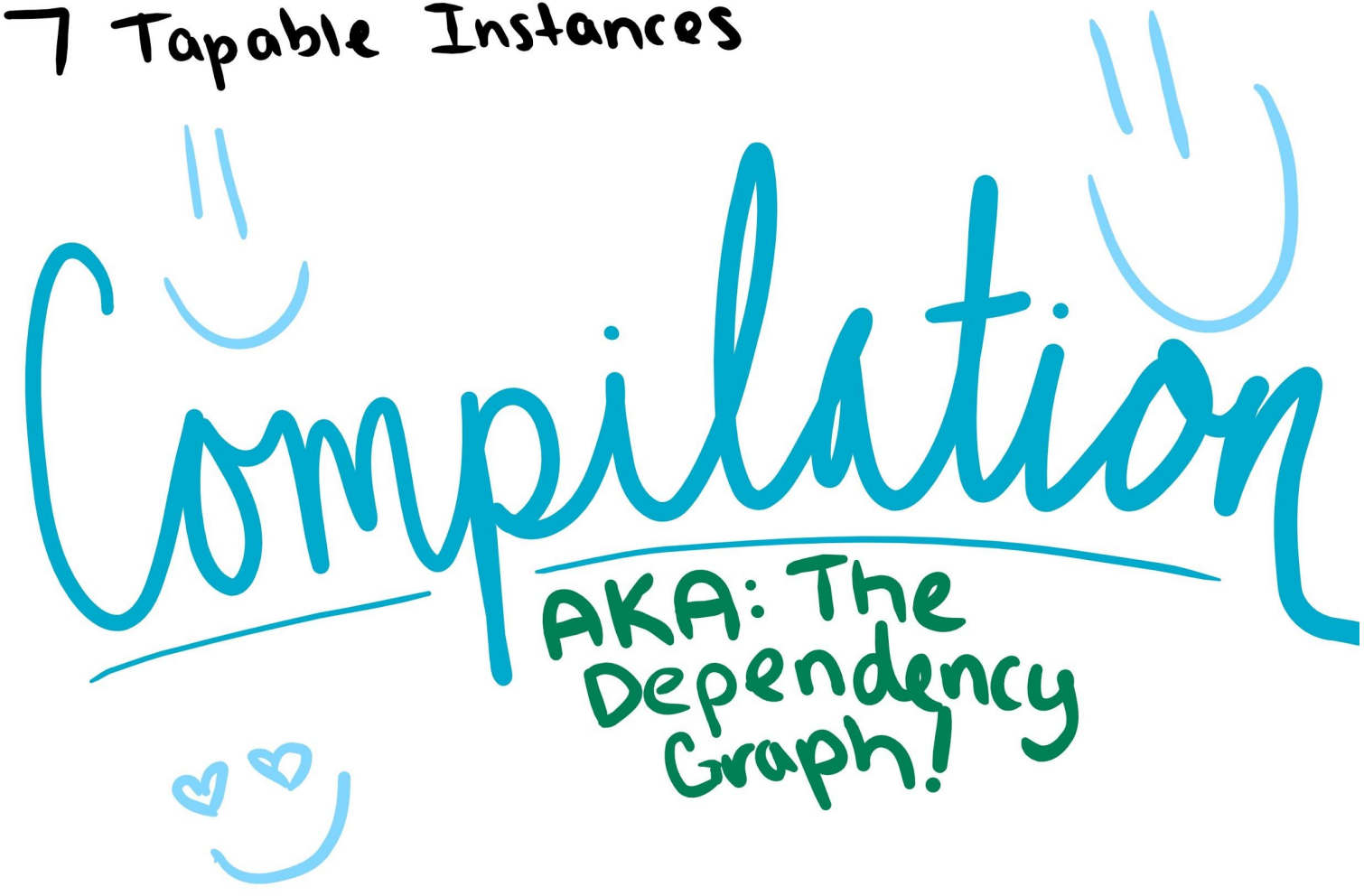
- Exposed via Node API
- Central Dispatch
- Start / Stop

7 Tapable Instances

Compilation

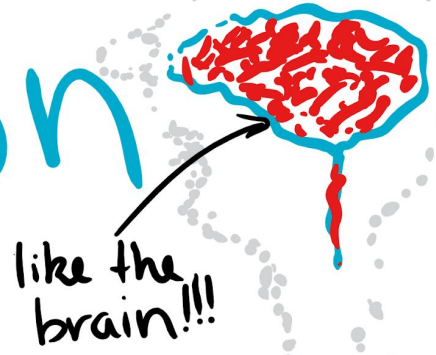
AKA: The Dependency Graph!

7 Tapable Instances



7 Tapable Instances

Compilation



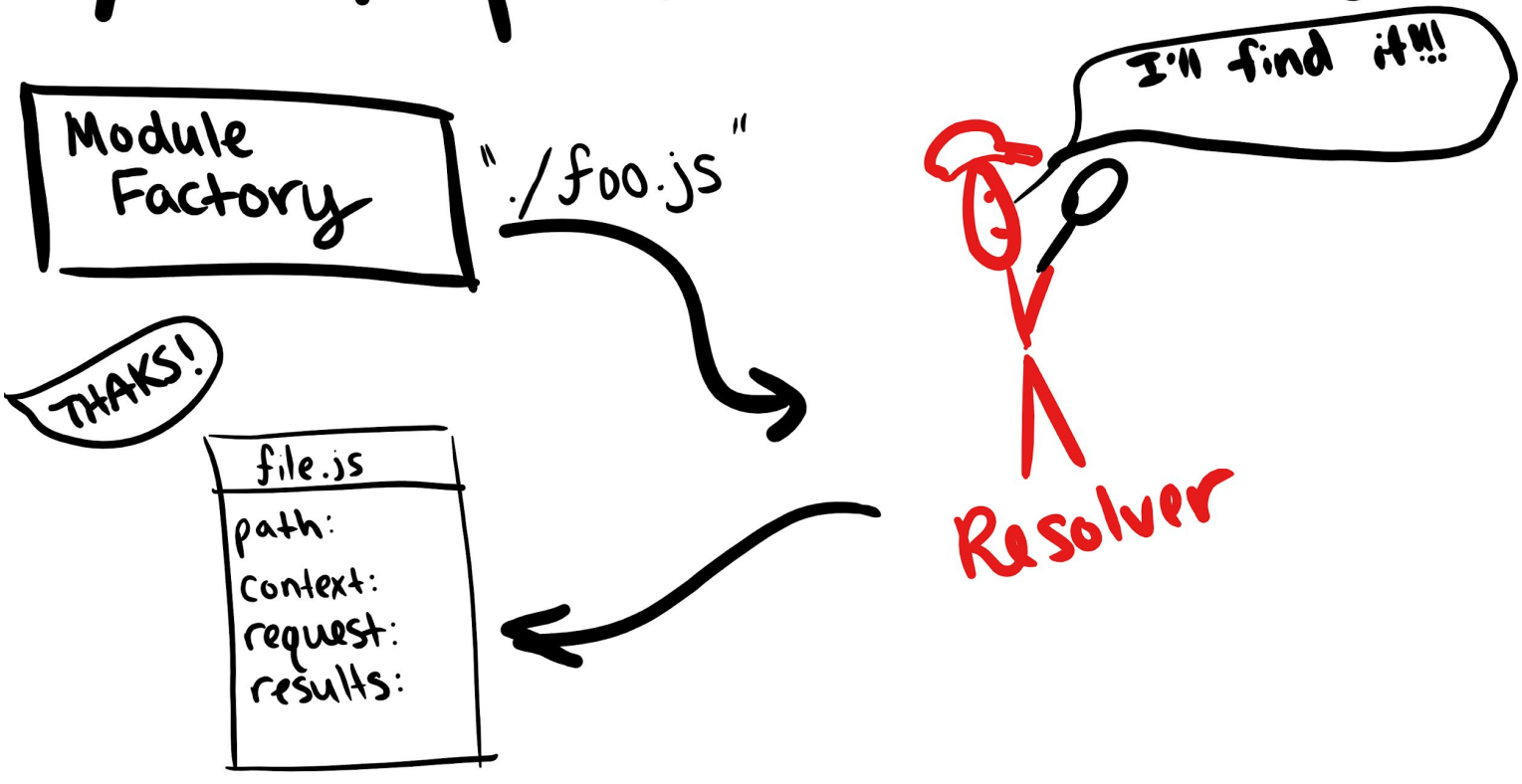
- Created by the Compiler.
- Contains dep graph traversal algo.

7ish Tapable instance

The "Bloodhound"

The Resolver

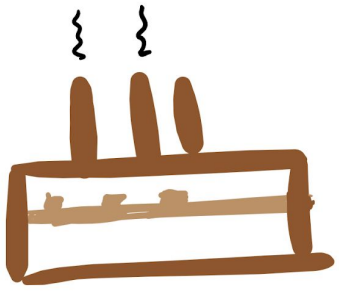
7ish Tapable Instances



ResolverFactory.js ×

```
*/
var Resolver = require("./Resolver");

var SyncAsyncFileSystemDecorator = require("./SyncAsyncFileSystemDecorator");

var ParsePlugin = require("./ParsePlugin");
var DescriptionFilePlugin = require("./DescriptionFilePlugin");
var NextPlugin = require("./NextPlugin");
var TryNextPlugin = require("./TryNextPlugin");
var ModuleKindPlugin = require("./ModuleKindPlugin");
var FileKindPlugin = require("./FileKindPlugin");
var JoinRequestPlugin = require("./JoinRequestPlugin");
var ModulesInHierachicDirectoriesPlugin = require("./ModulesInHierachicDirectoriesPlugin");
var ModulesInRootPlugin = require("./ModulesInRootPlugin");
var AliasPlugin = require("./AliasPlugin");
var AliasFieldPlugin = require("./AliasFieldPlugin");
var ConcordExtensionsPlugin = require("./ConcordExtensionsPlugin");
var ConcordMainPlugin = require("./ConcordMainPlugin");
var ConcordModulesPlugin = require("./ConcordModulesPlugin");
var DirectoryExistsPlugin = require("./DirectoryExistsPlugin");
var FileExistsPlugin = require("./FileExistsPlugin");
var SymlinkPlugin = require("./SymlinkPlugin");
var MainFieldPlugin = require("./MainFieldPlugin");
var UseFilePlugin = require("./UseFilePlugin");
var AppendPlugin = require("./AppendPlugin");
var ResultPlugin = require("./ResultPlugin");
var ModuleAppendPlugin = require("./ModuleAppendPlugin");
var UnsafeCachePlugin = require("./UnsafeCachePlugin");

exports.createResolver = function(options) {

    //// OPTIONS ////
```

7ish Tapable Instances



Module Factories

7ish Tapable Instances

Module Factories

- Takes Succesfully Resolved Requests.

7ish Tapable Instances

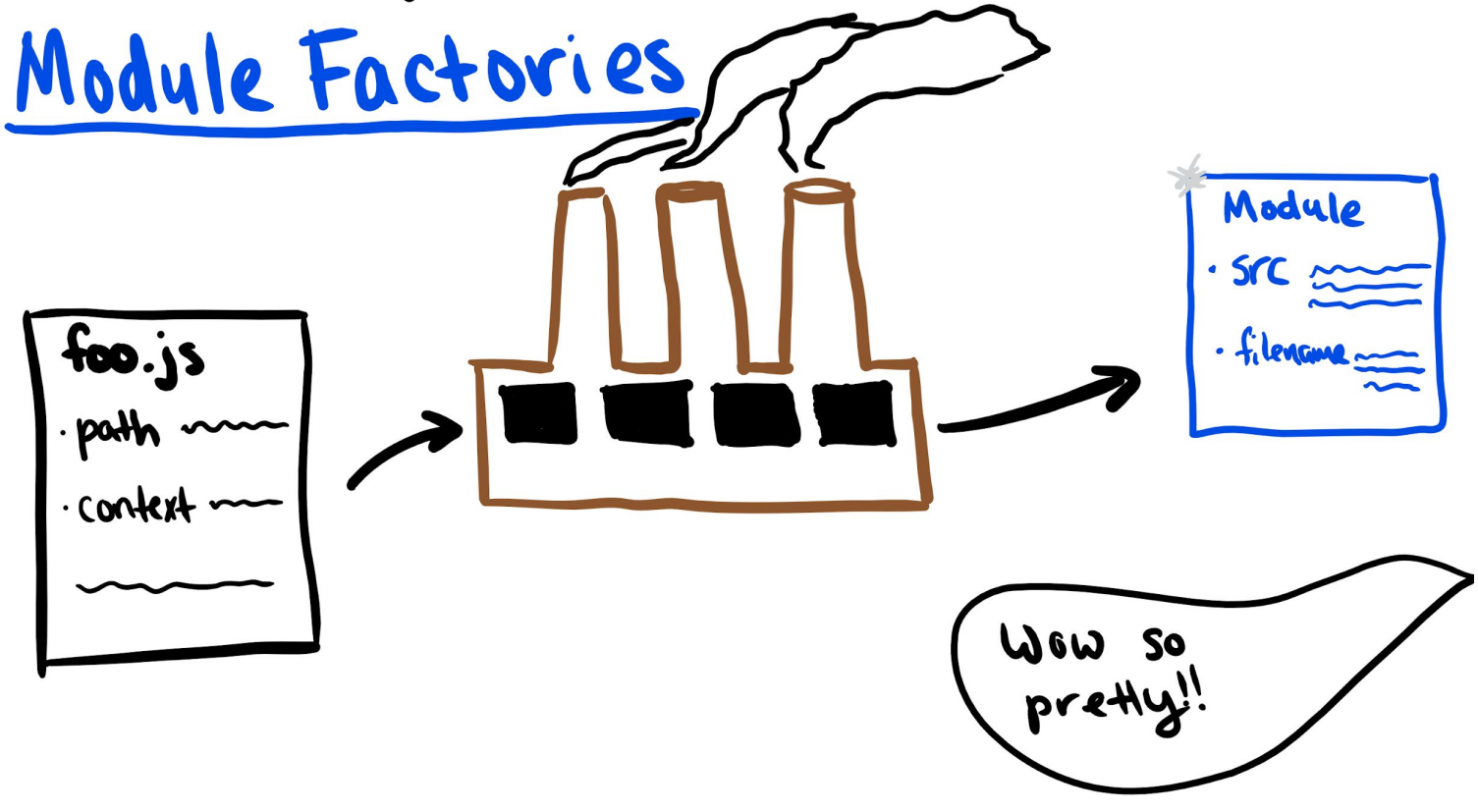
Module Factories

- Takes Succesfully Resolved Requests.

- Collects source for that file. Creates a Module obj.

7 ish Tapable Instances

Module Factories



7ish Tapable Instances

PARSER

I can haz ASTs

7ish Tapable Instances

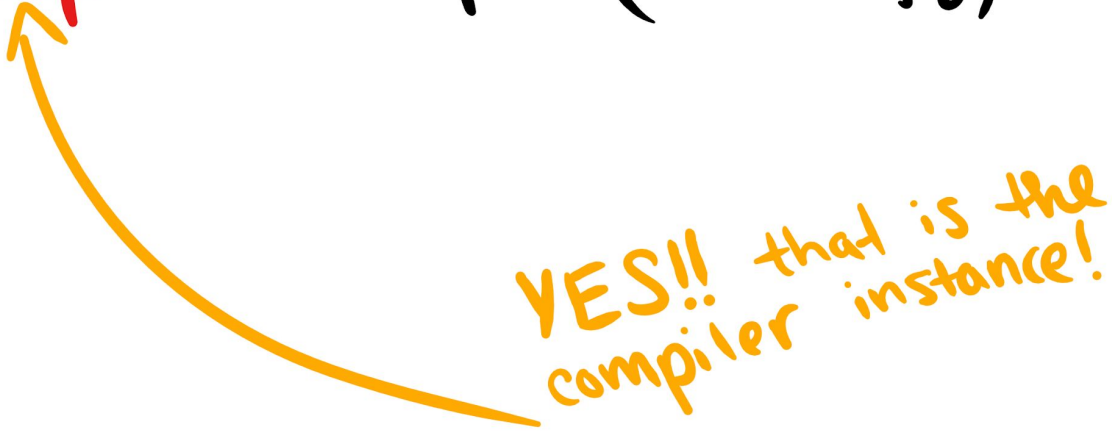
Parser

- Parses!!!!
- Takes a Module Object, turns into AST to parse

7ish Tapable Instances (aka classes)

COMPILER?

```
const webpack = require("webpack");
const compiler = webpack(someConfig);
```



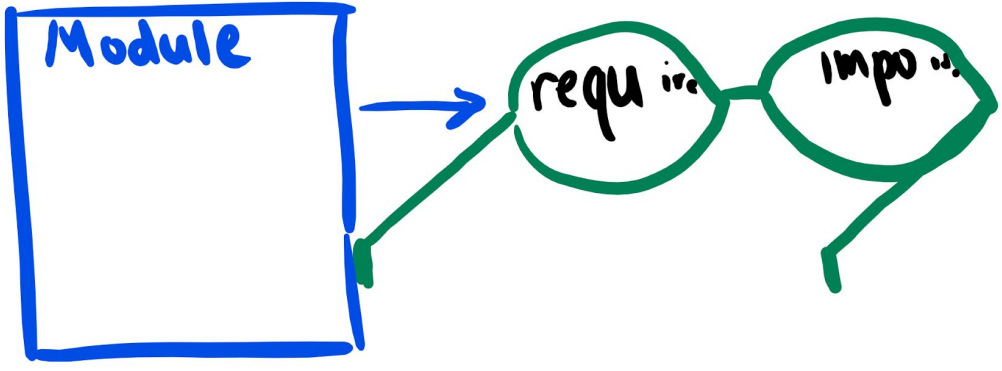
7ish Tapable Instances

Parser

- Parses!!!!
- Takes a Module Object, turns into AST to parse
- Find all require's imports, create Dependency's

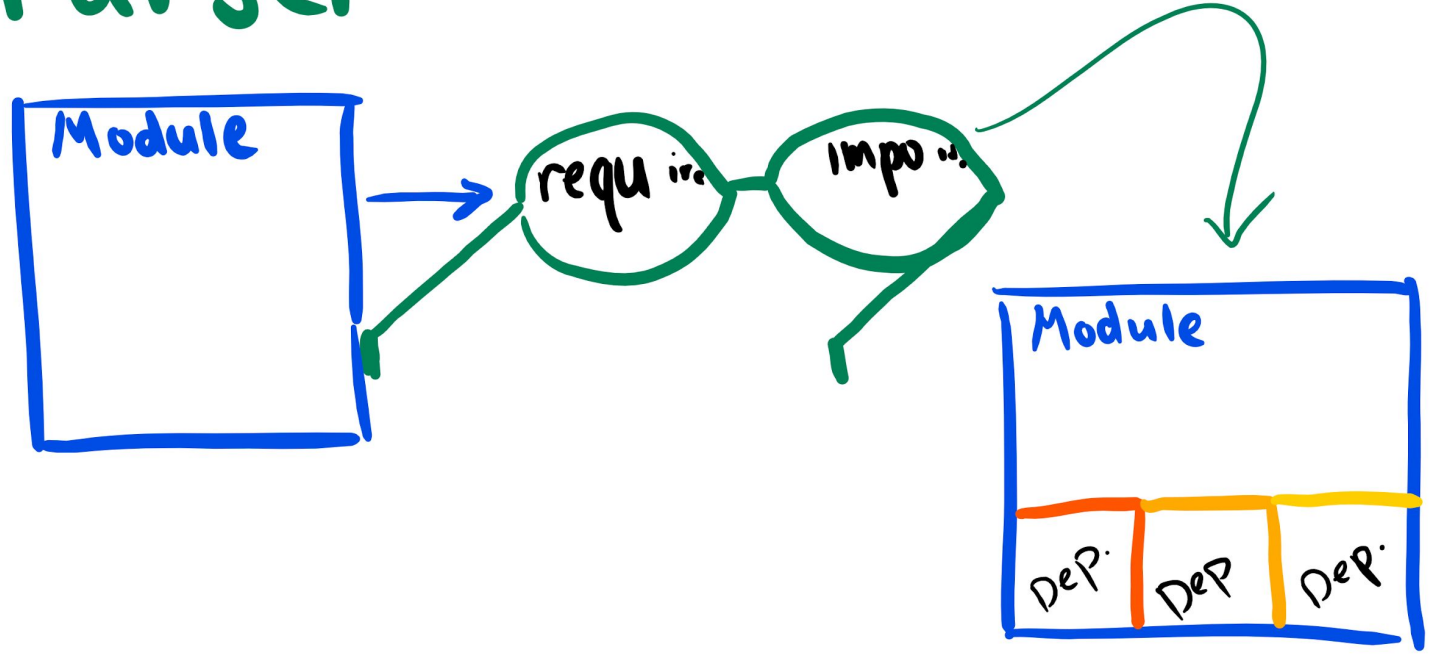
7^ish Tapable Instances

Parser



7ish Tapable Instances

Parser



7ish Tapable Instances

{TEMPLATE}

7ish Tapable Instances

Template

- Data binding for your modules

7^ish Tapable Instances

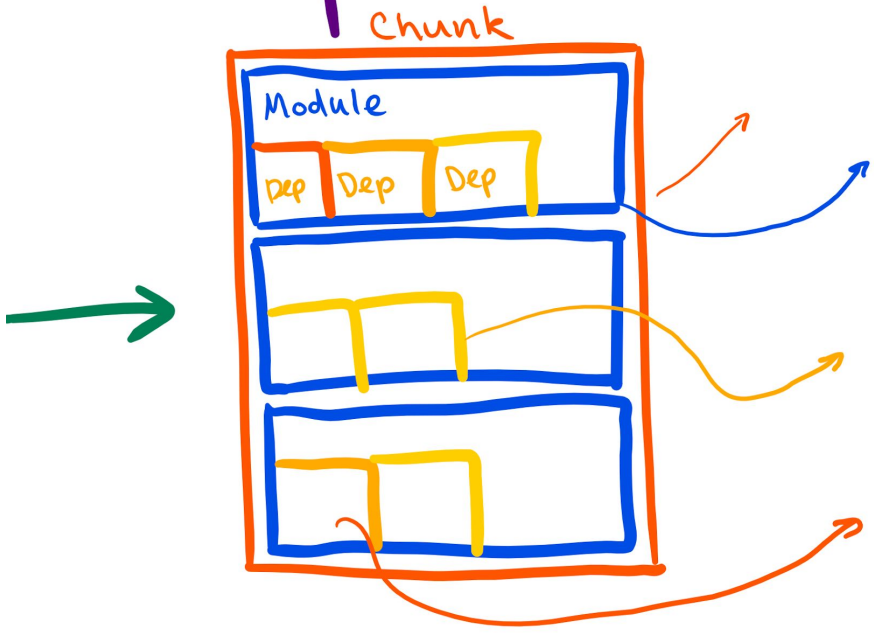
Template

- Data binding for your modules
- Creates the code you see in your bundles.

7ish Tapable Instances

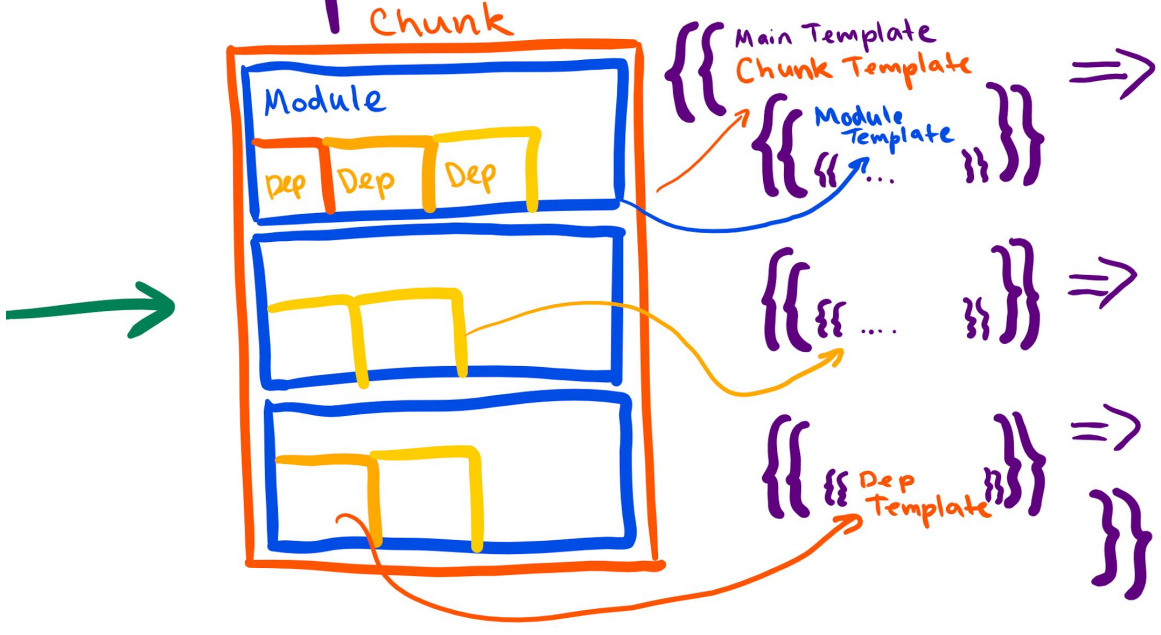
Template

"render"



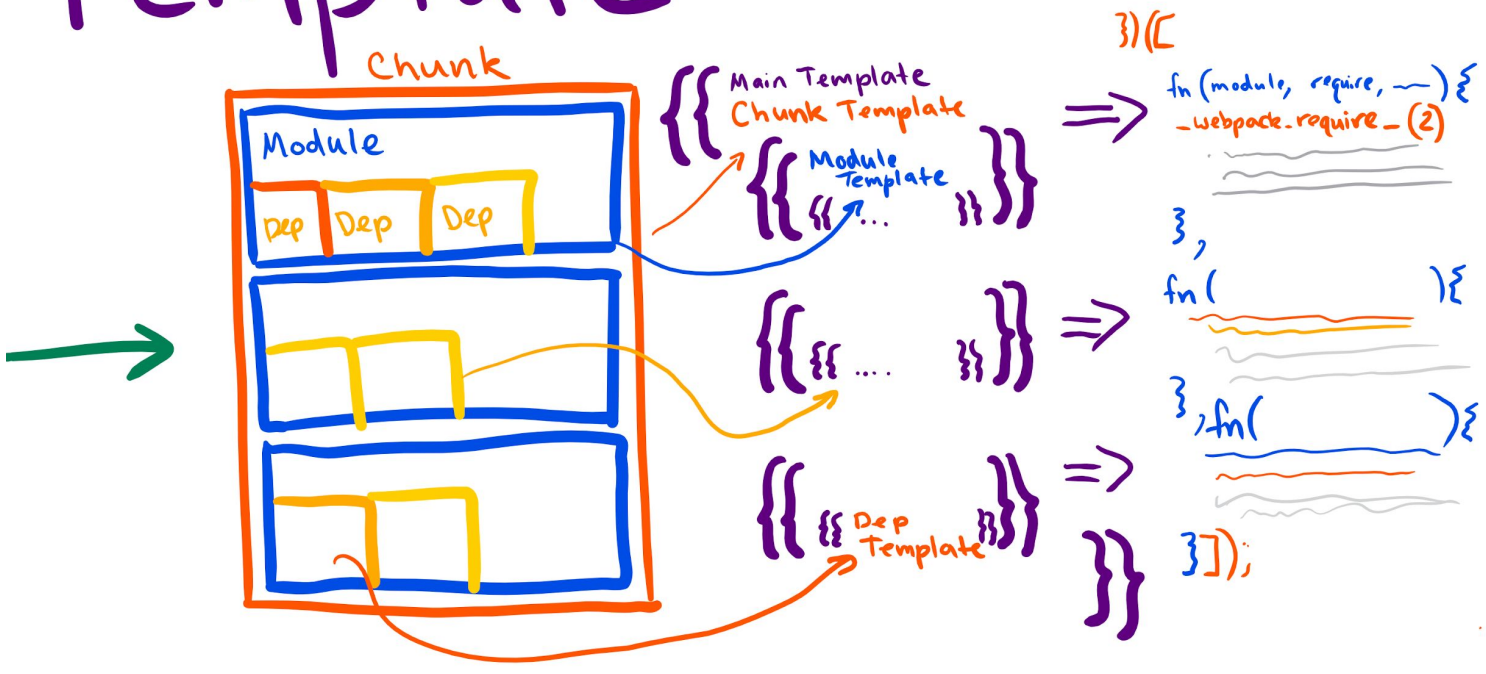
7ish Tapable Instances

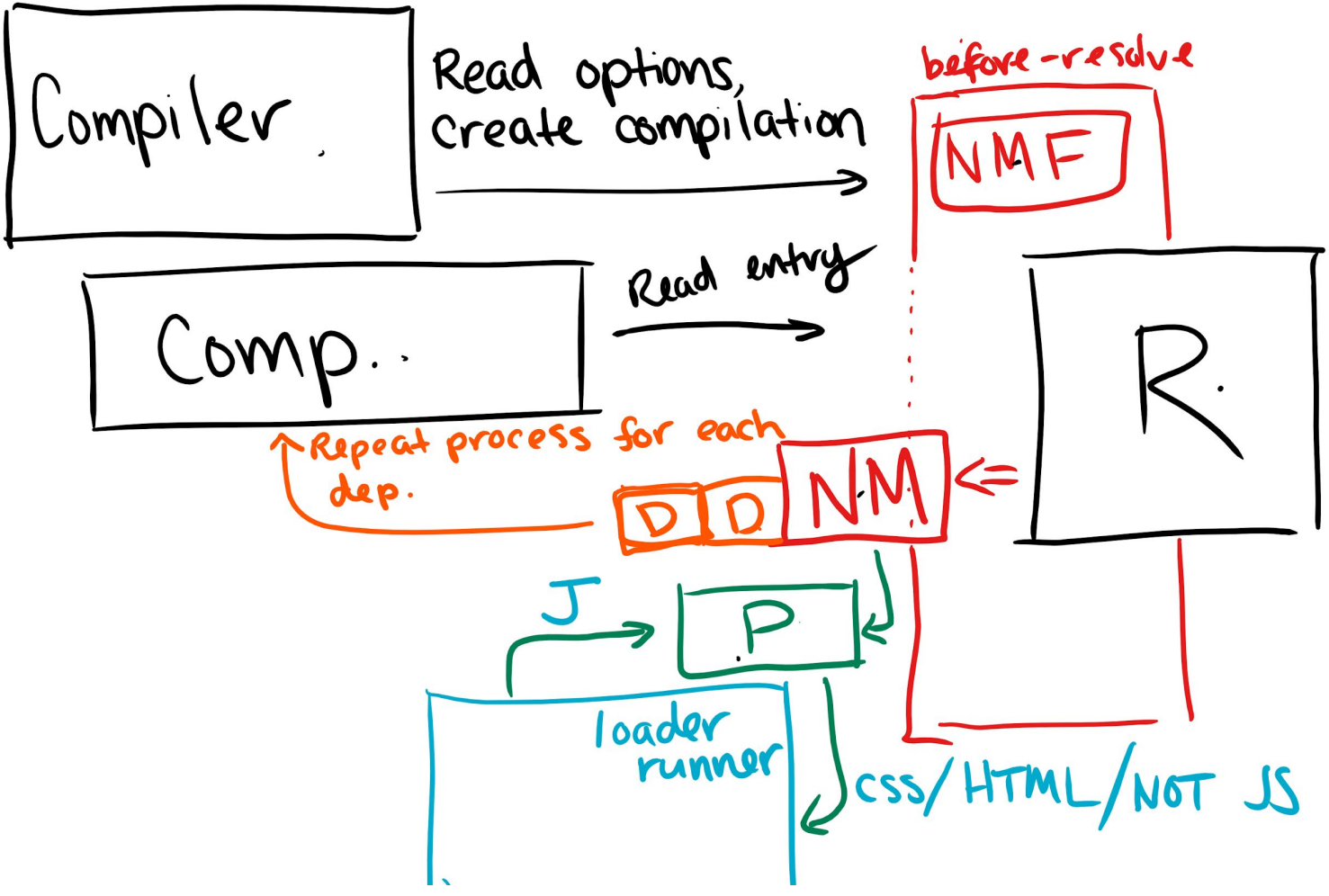
Template



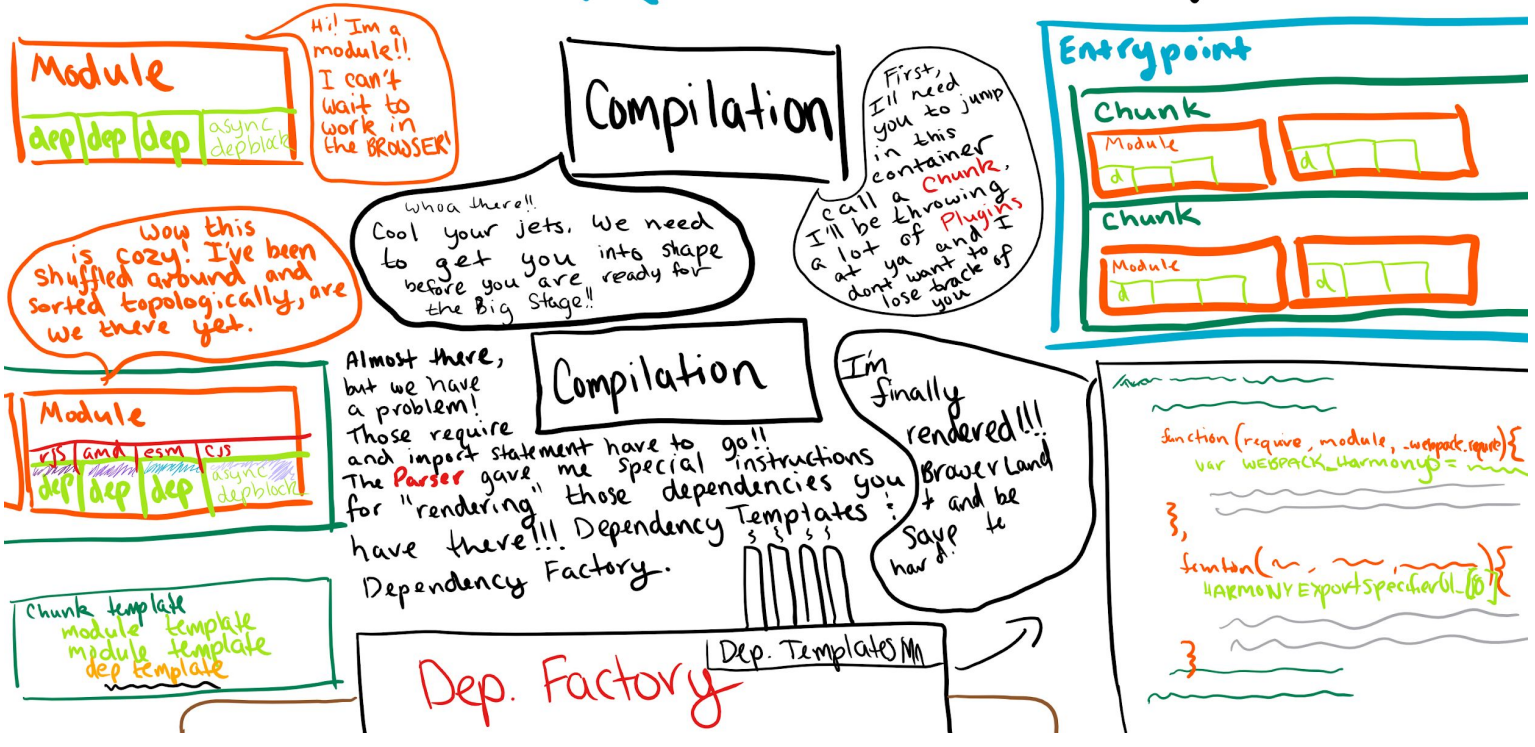
7ish Tapable Instances

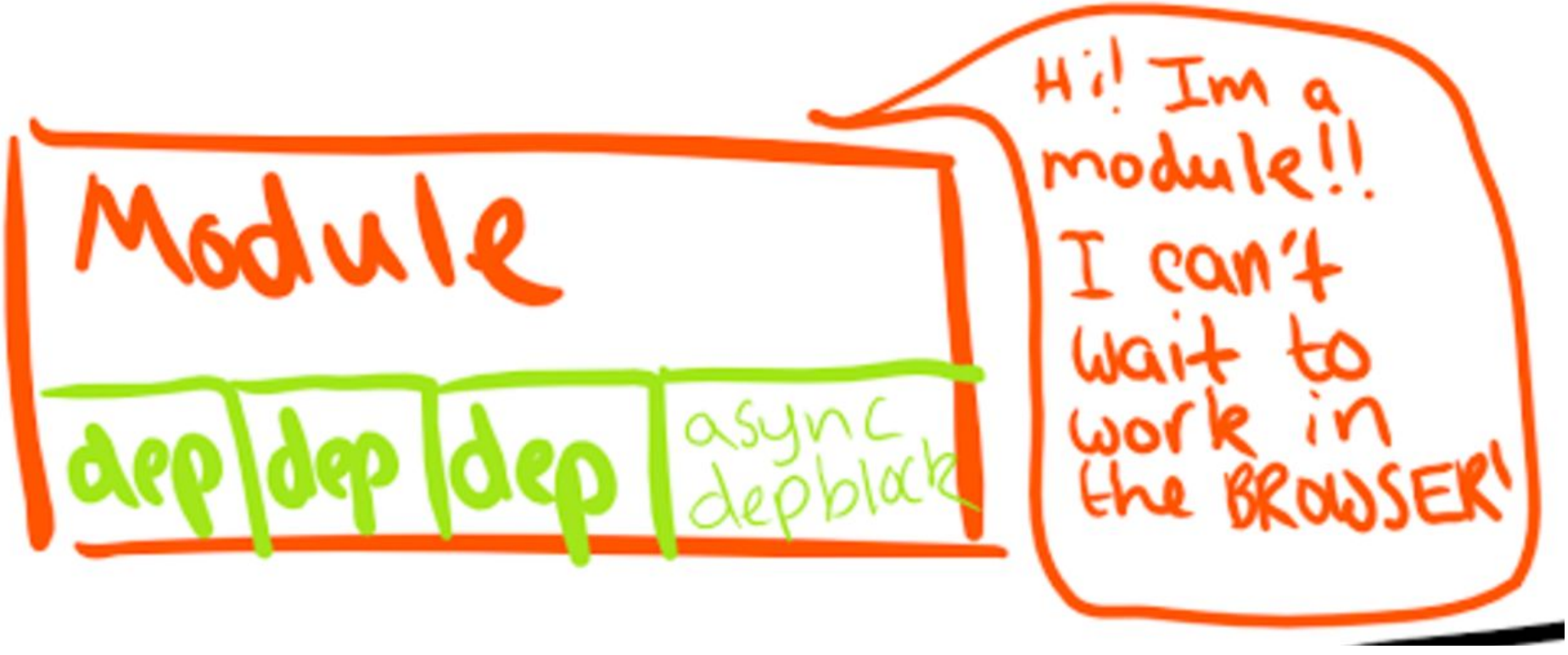
Template




Compiler
Read options, create compilation
before-resolve
NMF
Comp..
Read entry
R.
Repeat process for each dep.
D
D
NM
J
P
loader runner
CSS/HTML/NOT JS

How a Module gets to the BROWSER with webpack!



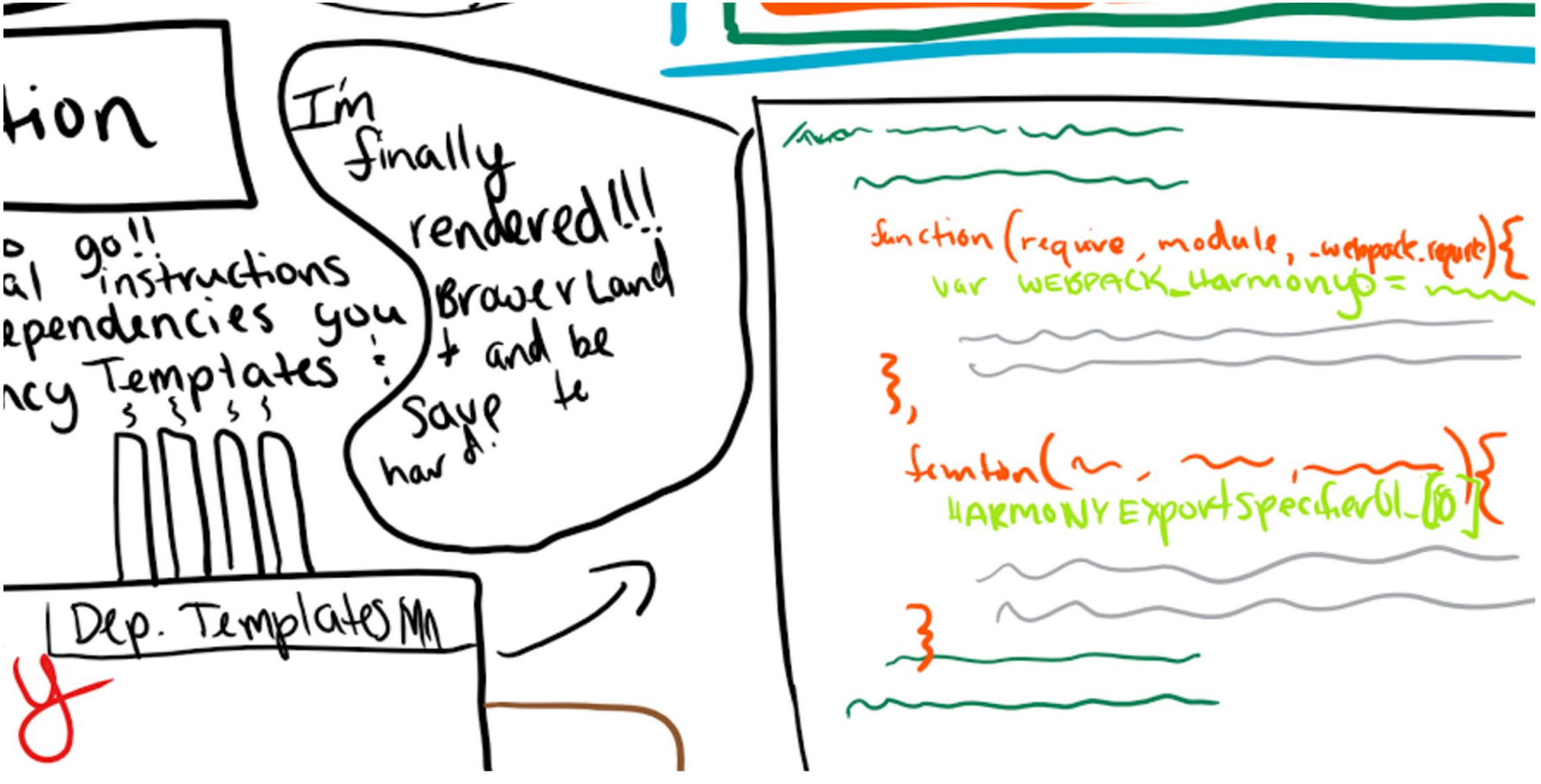

Module
dep
dep
dep
async dep block
Hi! Im a module!! I can't wait to work in the BROWSER!


Im a
dule!!
can't
it to
rk in
e BROWSER!
Compilation
First,
I'll need
you to jump
in this
container
call a Chunk,
I'll be throwing
a lot of Plugins
at ya and I
dont want to
lose track of
you
Whoa there!!
Cool your jets. We need
to get you into shape
before you are ready for
the Big Stage!!

Compilation

Almost there, but we have a problem! Those require and import statement have to go!!

The Parser gave me special instructions for "rendering" those dependencies you have there!!! Dependency Templates : Dependency Factory.


tion
go!!
al instructions
ependencies you
ncy Templates :
I'm finally rendered!!!
Brower Land
t and be
Save te
haw d!
Dep. Templates
function (require, module, __webpack_require__){
var WEBPACK_HARMONY_
},
function(
HARMONY EXPORT SPECIFIER_[0]

Chapter 6: Custom Plugins

Additional Resources: https://github.com/TheLarkInn/everything-is-a-plugin
Additional Workshop Video https://www.youtube.com/watch?v=4tQiJaFzuJ8

You just learned how webpack works entirely under the hood.