JAVASCRIPT-PLUS-S22-Modules

modules
Training Clarusway
Pear Deck - March 2, 2022 at 10:31AM

Part 1 - Summary

Use this space to summarize your thoughts on the lesson

Part 2 - Responses

Slide 1

JavaScript Modules
Javascript Plus Session-22

CLARUSWAY
WAY TO REINVENT YOURSELF

Use this space to take notes:

Slide 2

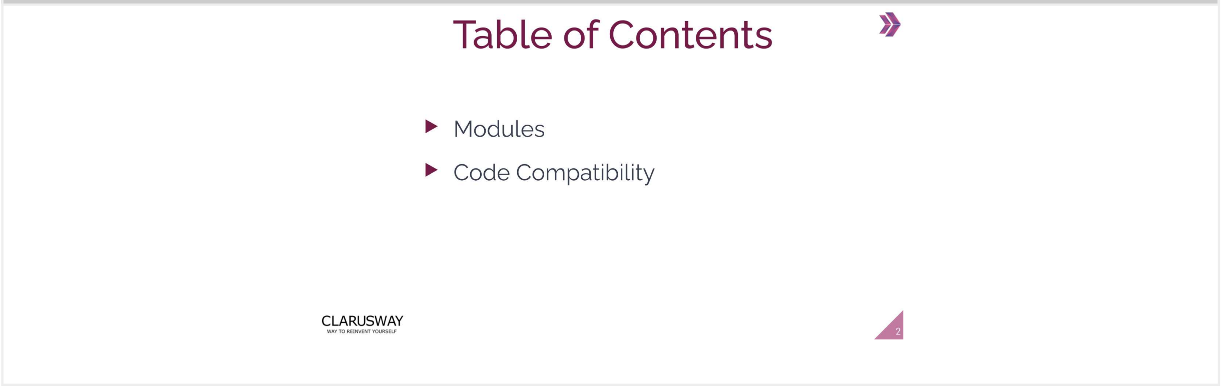


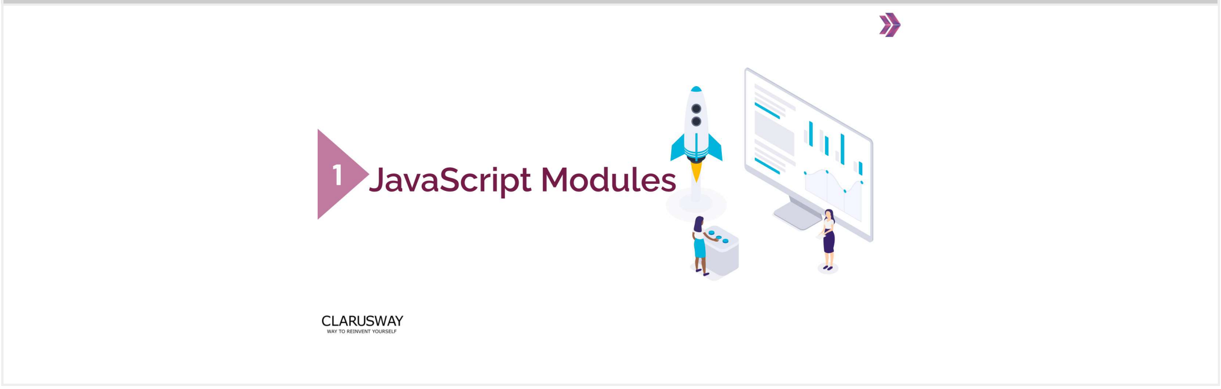
Table of Contents

CLARUSWAY
WAY TO REINVENT YOURSELF

2

Use this space to take notes:

Slide 3

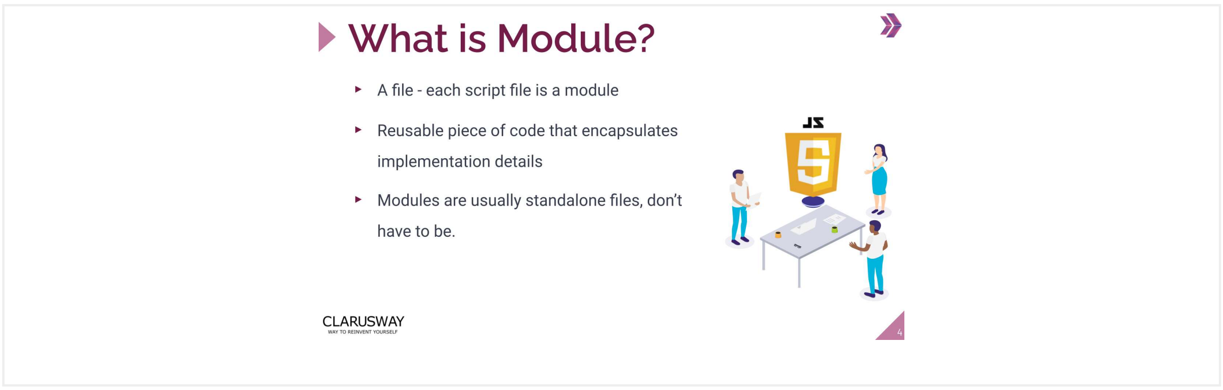


1 JavaScript Modules

CLARUSWAY
WAY TO REINVENT YOURSELF

Use this space to take notes:

Slide 4



What is Module?

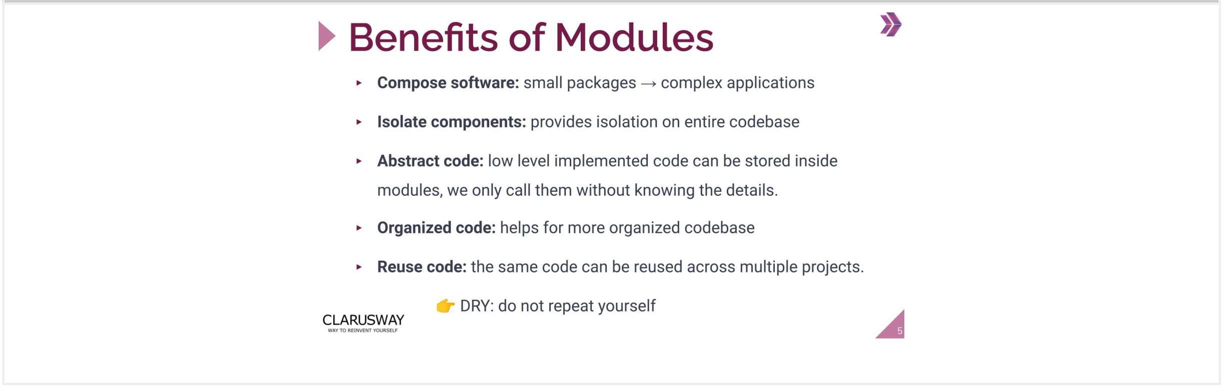
- A file - each script file is a module
- Reusable piece of code that encapsulates implementation details
- Modules are usually standalone files, don't have to be.

CLARUSWAY
WAY TO REINVENT YOURSELF

4

Use this space to take notes:

Slide 5



Benefits of Modules

- **Compose software:** small packages → complex applications
- **Isolate components:** provides isolation on entire codebase
- **Abstract code:** low level implemented code can be stored inside modules, we only call them without knowing the details.
- **Organized code:** helps for more organized codebase
- **Reuse code:** the same code can be reused across multiple projects.

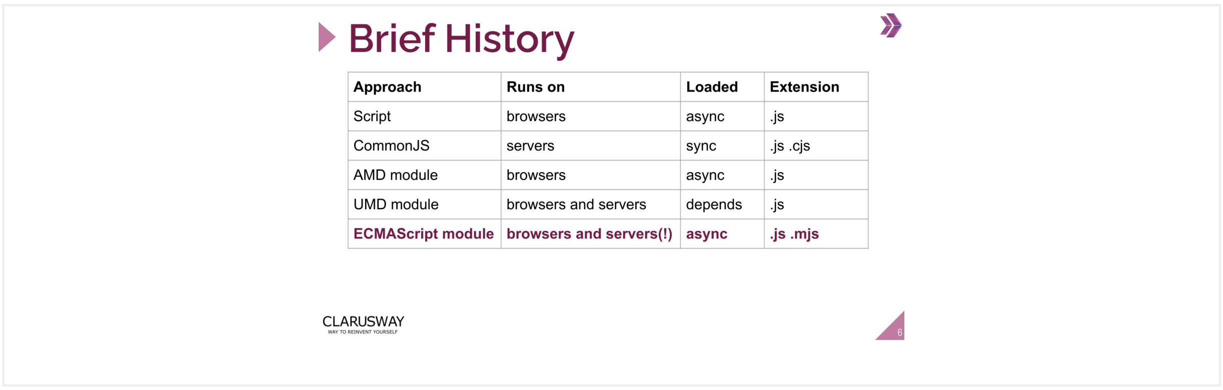
👉 DRY: do not repeat yourself

CLARUSWAY
WAY TO REINVENT YOURSELF

5

Use this space to take notes:

Slide 6

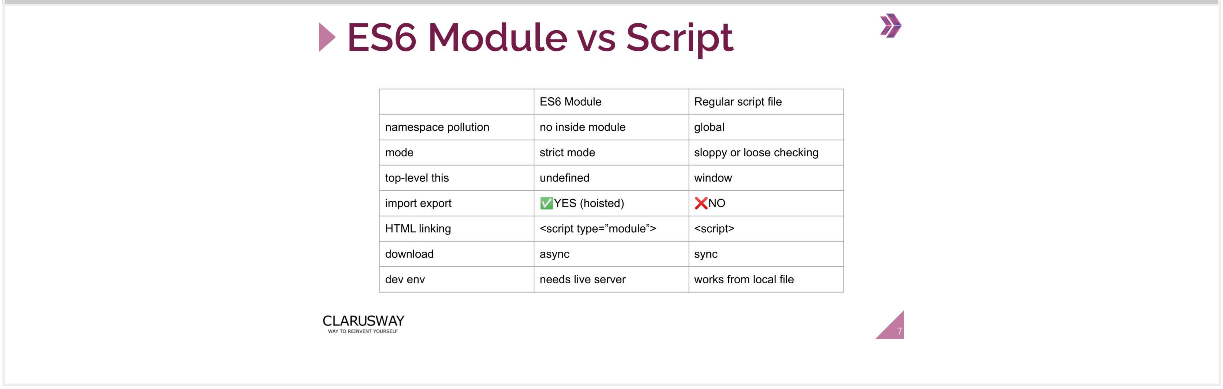


Brief History

Approach	Runs on	Loaded	Extension
Script	browsers	async	.js
CommonJS	servers	sync	.js .cjs
AMD module	browsers	async	.js
UMD module	browsers and servers	depends	.js
ECMAScript module	**browsers and servers(!)**	**async**	**.js .mjs**

Use this space to take notes:

Slide 7

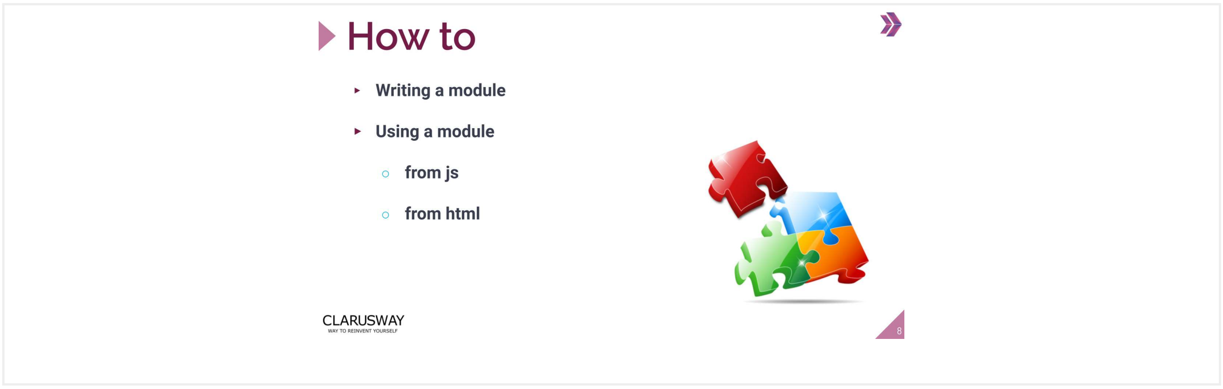


ES6 Module vs Script

	ES6 Module	Regular script file
namespace pollution	no inside module	global
mode	strict mode	sloppy or loose checking
top-level this	undefined	window
import export	✅YES (hoisted)	❌NO
HTML linking	<script type="module">	<script>
download	async	sync
dev env	needs live server	works from local file

Use this space to take notes:

Slide 8



How to

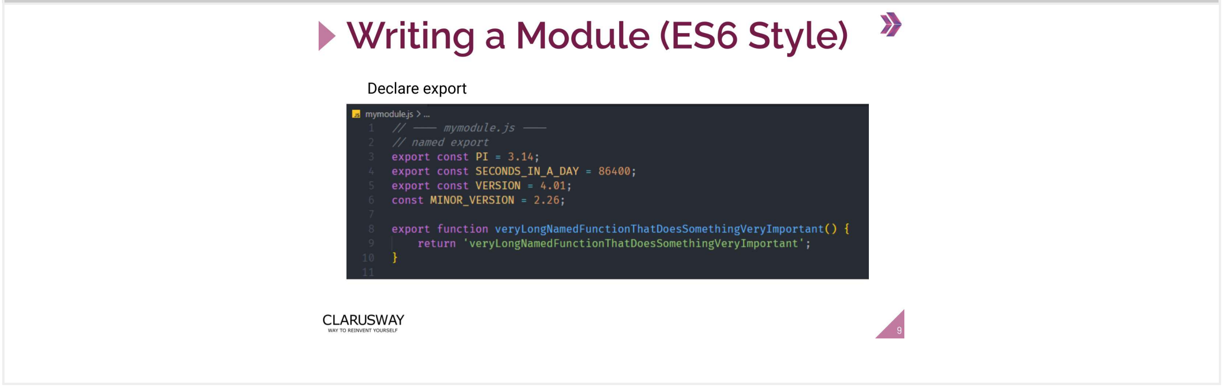
- Writing a module
- Using a module
    - from js
    - from html

CLARUSWAY
WAY TO REINVENT YOURSELF

8

Use this space to take notes:

Slide 9



Writing a Module (ES6 Style)

Declare export

mymodule.js > ...

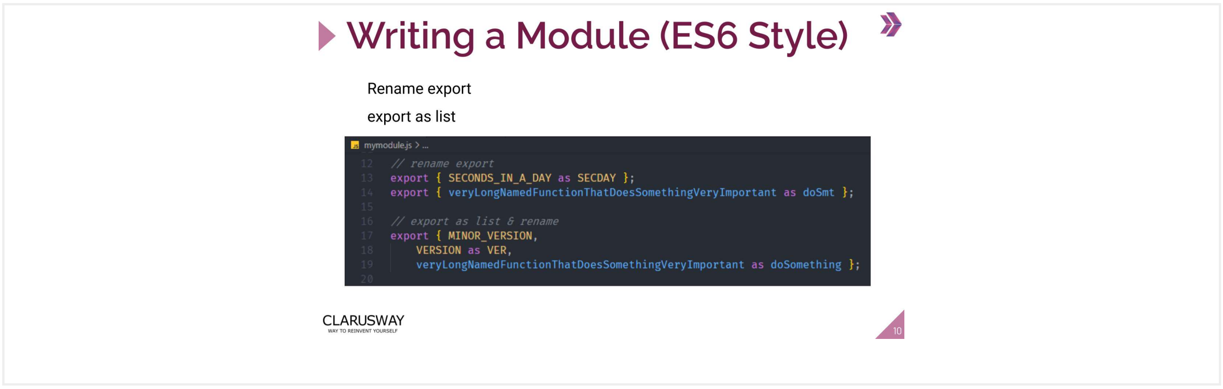
```
// ——— mymodule.js ———
// named export
export const PI = 3.14;
export const SECONDS_IN_A_DAY = 86400;
export const VERSION = 4.01;
const MINOR_VERSION = 2.26;

export function veryLongNamedFunctionThatDoesSomethingVeryImportant() {
    return 'veryLongNamedFunctionThatDoesSomethingVeryImportant';
}
```

CLARUSWAY
WAY TO REINVENT YOURSELF

9

Use this space to take notes:

Slide 10



Writing a Module (ES6 Style)

Rename export

export as list

mymodule.js > ...

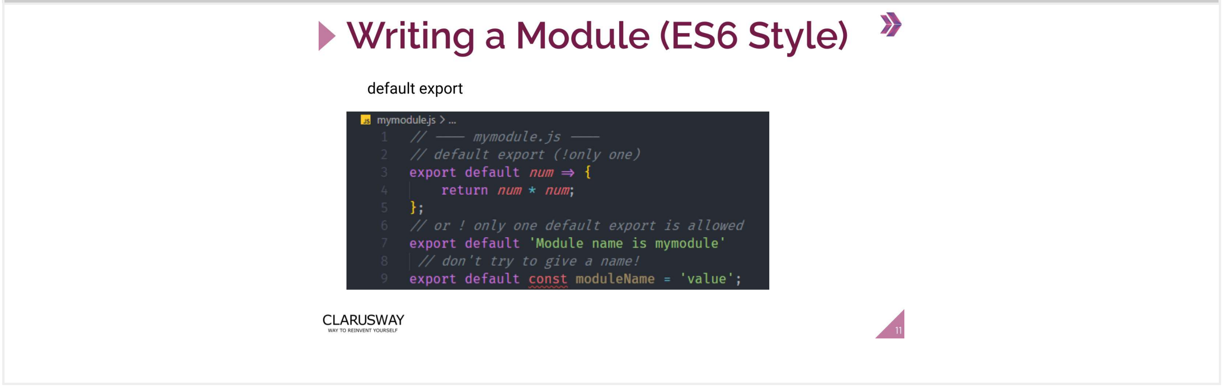
```
// rename export
export { SECONDS_IN_A_DAY as SECDAY };
export { veryLongNamedFunctionThatDoesSomethingVeryImportant as doSmt };

// export as list & rename
export { MINOR_VERSION,
    VERSION as VER,
    veryLongNamedFunctionThatDoesSomethingVeryImportant as doSomething };
```

CLARUSWAY
WAY TO REINVENT YOURSELF

10

Use this space to take notes:

Slide 11



Writing a Module (ES6 Style)

default export

mymodule.js > ...

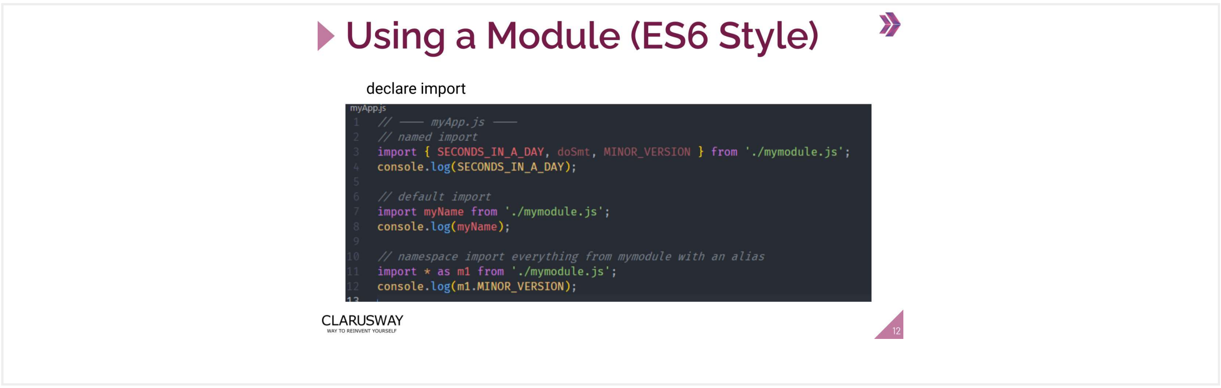
```
// ─── mymodule.js ───
// default export (!only one)
export default num => {
    return num * num;
};
// or ! only one default export is allowed
export default 'Module name is mymodule'
 // don't try to give a name!
export default const moduleName = 'value';
```

CLARUSWAY
WAY TO REINVENT YOURSELF

11

Use this space to take notes:

Slide 12



Using a Module (ES6 Style)

declare import

myApp.js

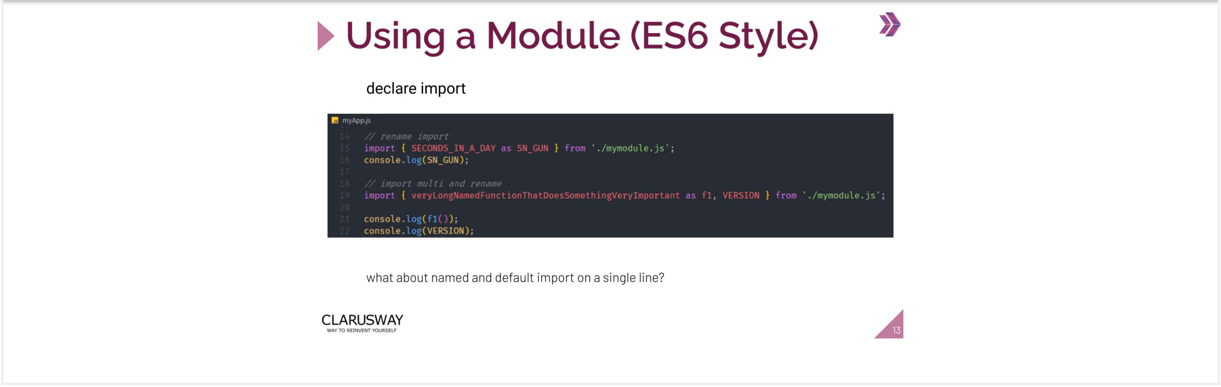
```javascript
// —— myApp.js ——
// named import
import { SECONDS_IN_A_DAY, doSmt, MINOR_VERSION } from './mymodule.js';
console.log(SECONDS_IN_A_DAY);

// default import
import myName from './mymodule.js';
console.log(myName);

// namespace import everything from mymodule with an alias
import * as m1 from './mymodule.js';
console.log(m1.MINOR_VERSION);
```

CLARUSWAY
WAY TO REINVENT YOURSELF

12

Use this space to take notes:

Slide 13



Using a Module (ES6 Style)

declare import

myApp.js

```javascript
// rename import
import { SECONDS_IN_A_DAY as SN_GUN } from './mymodule.js';
console.log(SN_GUN);

// import multi and rename
import { veryLongNamedFunctionThatDoesSomethingVeryImportant as f1, VERSION } from './mymodule.js';

console.log(f1());
console.log(VERSION);
```

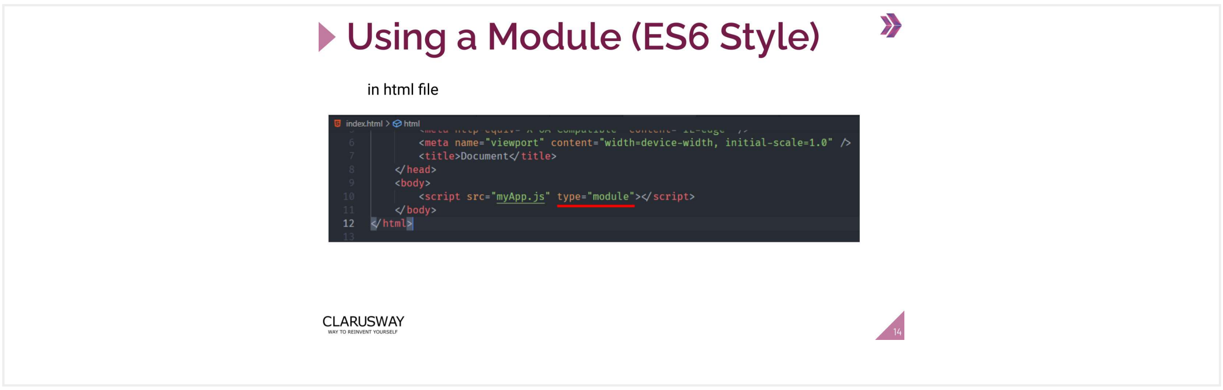
what about named and default import on a single line?

CLARUSWAY
WAY TO REINVENT YOURSELF

13

Use this space to take notes:

Slide 14



Using a Module (ES6 Style)

in html file

index.html > html

```
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Document</title>
    </head>
    <body>
        <script src="myApp.js" type="module"></script>
    </body>
</html>
```

CLARUSWAY
WAY TO REINVENT YOURSELF

14

Use this space to take notes:

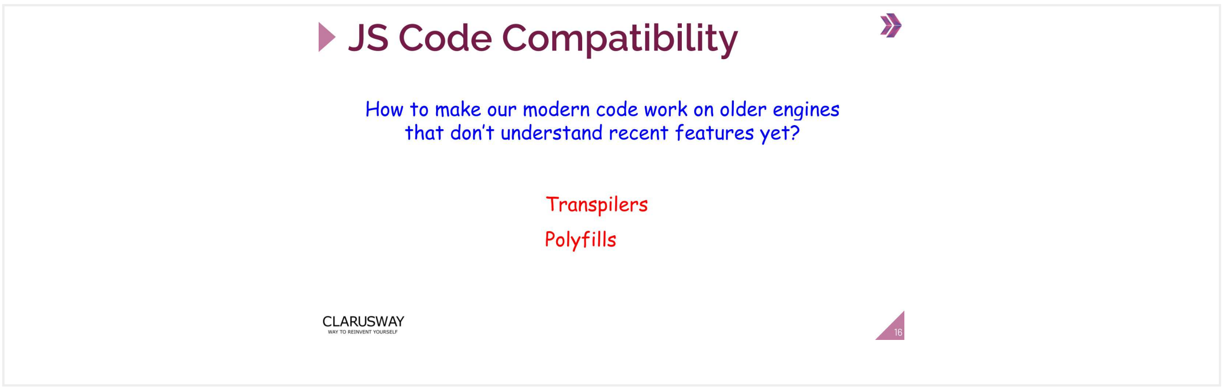
Slide 15

1 JavaScript Code Compatibility

CLARUSWAY
WAY TO REINVENT YOURSELF

Use this space to take notes:

Slide 16



JS Code Compatibility

How to make our modern code work on older engines
that don't understand recent features yet?

Transpilers

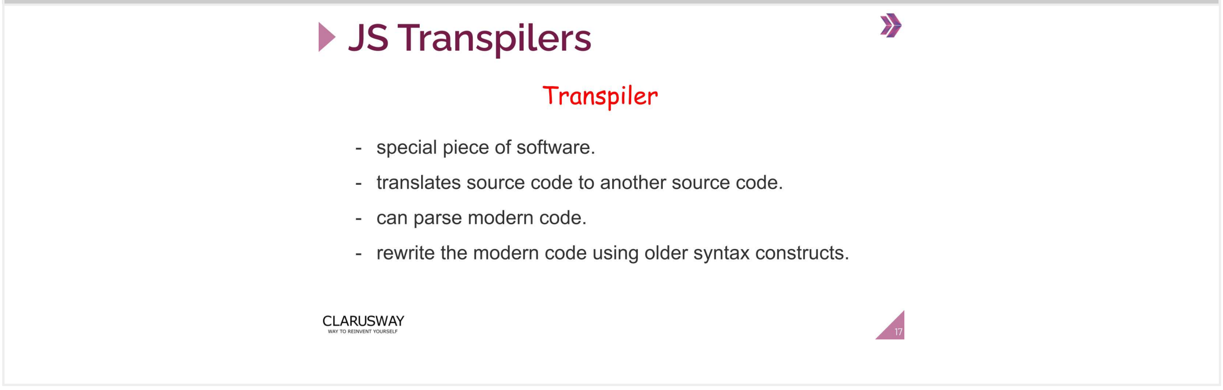
Polyfills

CLARUSWAY
WAY TO REINVENT YOURSELF

16

Use this space to take notes:

Slide 17



JS Transpilers

Transpiler

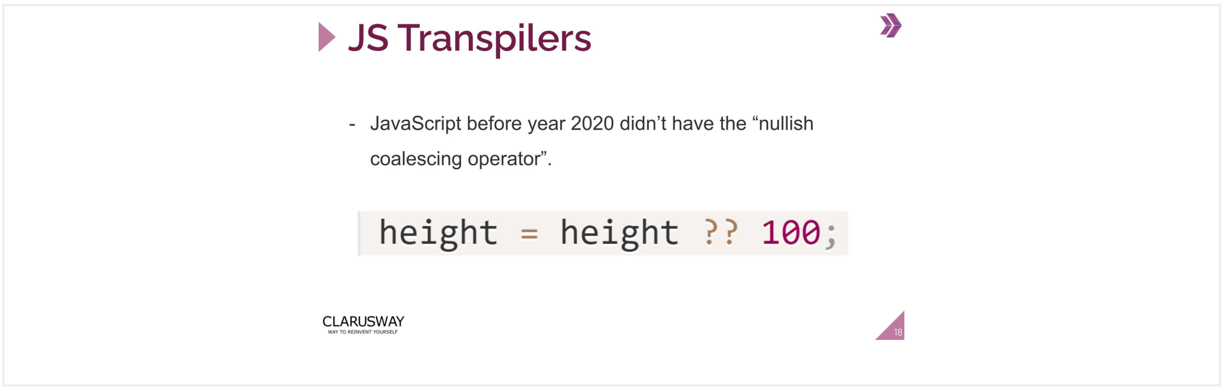
- special piece of software.
- translates source code to another source code.
- can parse modern code.
- rewrite the modern code using older syntax constructs.

CLARUSWAY
WAY TO REINVENT YOURSELF

17

Use this space to take notes:

Slide 18



JS Transpilers

- JavaScript before year 2020 didn’t have the “nullish coalescing operator”.

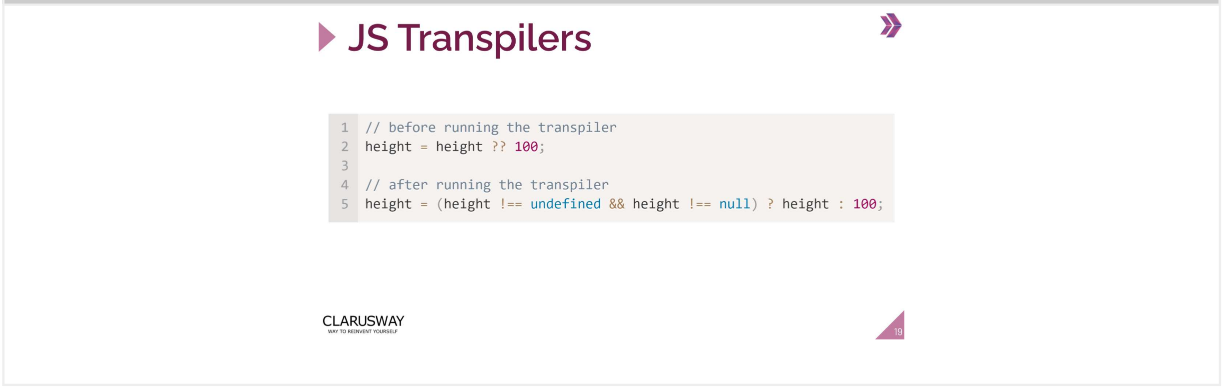
```
height = height ?? 100;
```

CLARUSWAY
WAY TO REINVENT YOURSELF

18

Use this space to take notes:

Slide 19



JS Transpilers

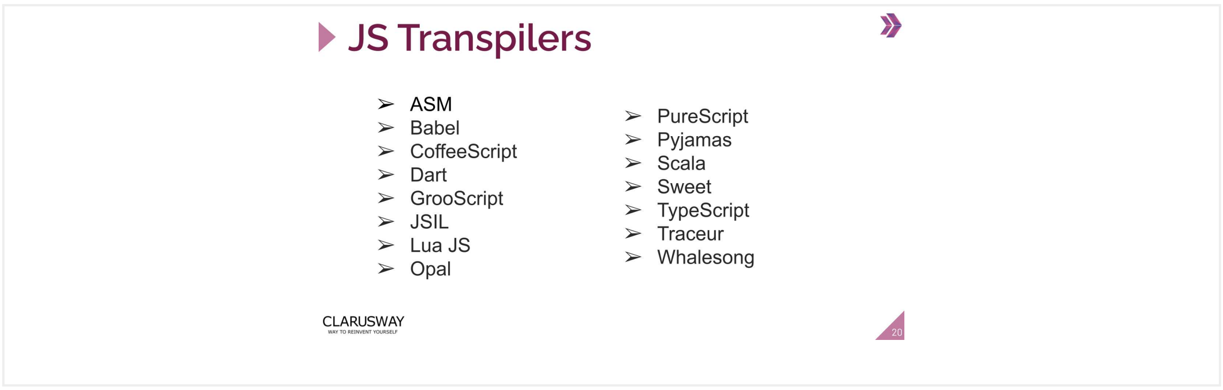
```
// before running the transpiler
height = height ?? 100;

// after running the transpiler
height = (height !== undefined && height !== null) ? height : 100;
```

CLARUSWAY
WAY TO REINVENT YOURSELF

19

Use this space to take notes:

Slide 20



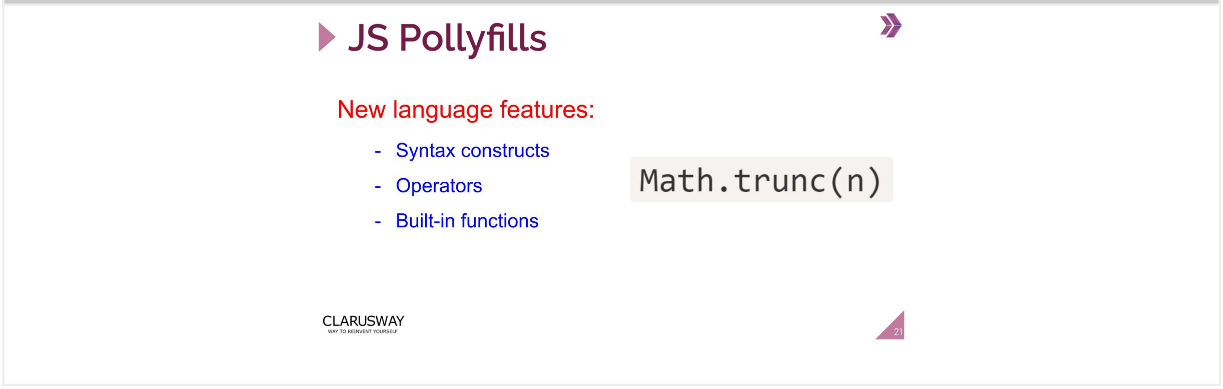
JS Transpilers

- ASM
- Babel
- CoffeeScript
- Dart
- GrooScript
- JSIL
- Lua JS
- Opal
- PureScript
- Pyjamas
- Scala
- Sweet
- TypeScript
- Traceur
- Whalesong

CLARUSWAY
WAY TO REINVENT YOURSELF
20

Use this space to take notes:

Slide 21



JS Pollyfills

New language features:

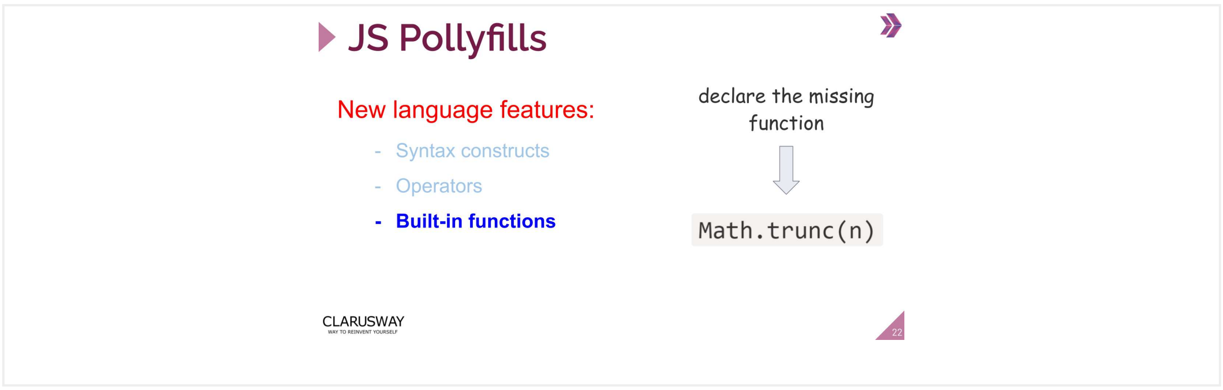
- Syntax constructs
- Operators
- Built-in functions

```
Math.trunc(n)
```

CLARUSWAY
WAY TO REINVENT YOURSELF
21

Use this space to take notes:

Slide 22



JS Pollyfills

New language features:

- Syntax constructs
- Operators
- **Built-in functions**

declare the missing function

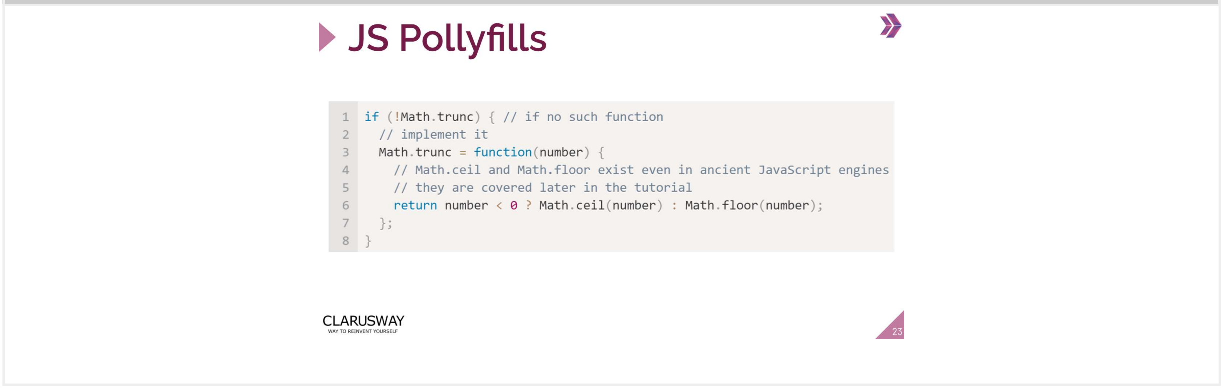
`Math.trunc(n)`

CLARUSWAY
WAY TO REINVENT YOURSELF

22

Use this space to take notes:

Slide 23



JS Pollyfills

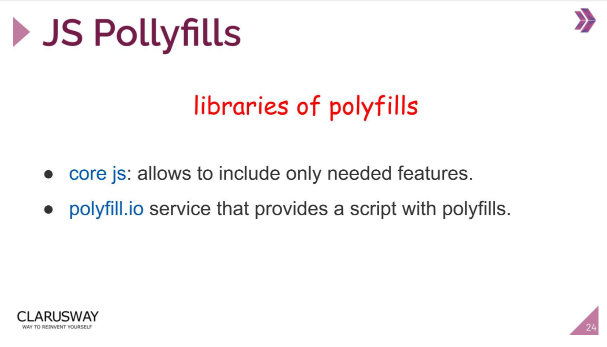
```
if (!Math.trunc) { // if no such function
  // implement it
  Math.trunc = function(number) {
    // Math.ceil and Math.floor exist even in ancient JavaScript engines
    // they are covered later in the tutorial
    return number < 0 ? Math.ceil(number) : Math.floor(number);
  };
}
```

CLARUSWAY
WAY TO REINVENT YOURSELF

23

Use this space to take notes:

Slide 24



JS Pollyfills

libraries of polyfills

- core js: allows to include only needed features.
- polyfill.io service that provides a script with polyfills.

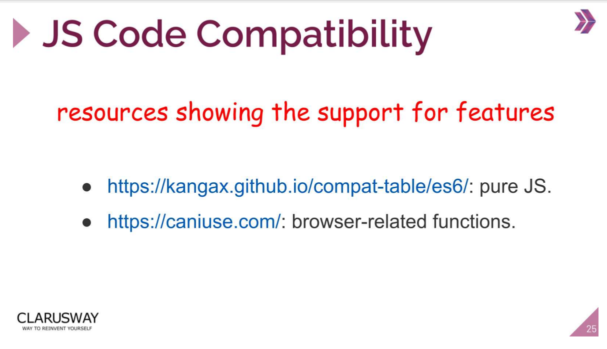
CLARUSWAY
WAY TO REINVENT YOURSELF

24

Link(s) on this slide:
- https://github.com/zloirock/core-js
- http://polyfill.io/

Use this space to take notes:

Slide 25



JS Code Compatibility

resources showing the support for features

- https://kangax.github.io/compat-table/es6/: pure JS.
- https://caniuse.com/: browser-related functions.

CLARUSWAY
WAY TO REINVENT YOURSELF

25

Link(s) on this slide:
- https://kangax.github.io/compat-table/es6/
- https://caniuse.com/

Use this space to take notes:

Slide 26

Link(s) on this slide:

- https://create.kahoot.it/details/02-node-js-module/12e6f3f0-0ecc-44f1-8887-f950d944cce2

Use this space to take notes:

Slide 27

Use this space to take notes: