

Professional JS

Features You Need to Know



JS

Maximiliano Firtman

firt.dev

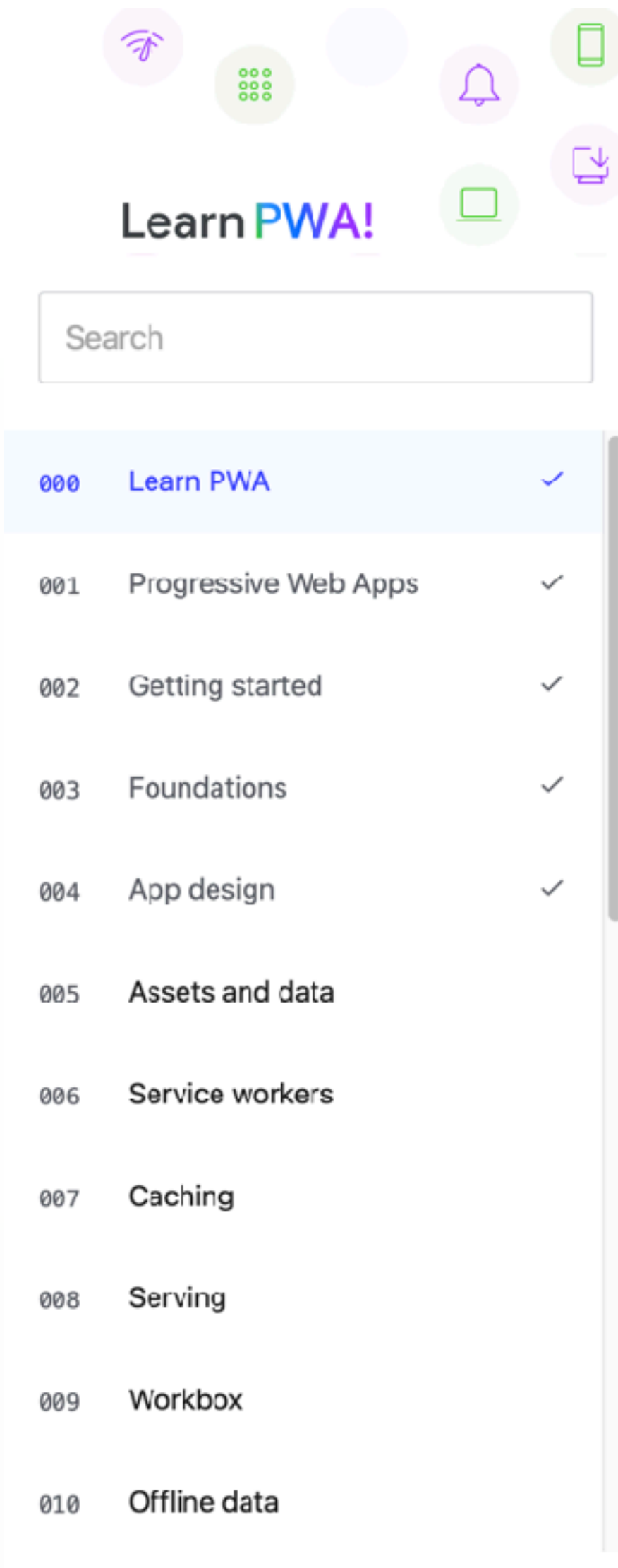
 x.com/firt

 linkedin.com/in/firtman

 github.com/firtman

About me

Maximiliano Firtman



000	Learn PWA	✓
001	Progressive Web Apps	✓
002	Getting started	✓
003	Foundations	✓
004	App design	✓
005	Assets and data	
006	Service workers	
007	Caching	
008	Serving	
009	Workbox	
010	Offline data	

web.dev > Learn > Learn PWA!

000

Learn PWA

A course that breaks down every aspect of modern progressive web app development.

Welcome to Learn Progressive Web Apps!

Welcome to Learn Progressive Web Apps!

This course covers the fundamentals of Progressive Web App development in easy-to-understand pieces. Over the following modules, you'll learn what a Progressive Web App is, how to create one or upgrade your existing web content, and how to add all the pieces for an offline, installable app. Use the menu pane to navigate the modules. (The menu is at left on desktop or behind the hamburger menu on mobile.)

You'll learn PWA fundamentals like the Web App Manifest, service workers, how to design with an app in mind, how to use other tools to test and debug your PWA. After these fundamentals, you'll learn about integration with the platform and operating system, how to enhance your PWA's installation and usage experience, and how to offer an offline experience.

About me

Maximiliano Firtman

The logo for Frontend Masters, featuring the word "Frontend" in a grey sans-serif font and "Masters" in a red, stylized script font.

Frontend *Masters*

Frontend Courses

Mobile App Courses

Backend Courses

What we will cover today



Introduction to ECMAScript

Recap of ES2015 (ES6)

Language Enhancements

Array and Collection Enhancements

Asynchronous Programming

Advanced Techniques

Pre-requisites



Basic JavaScript experience

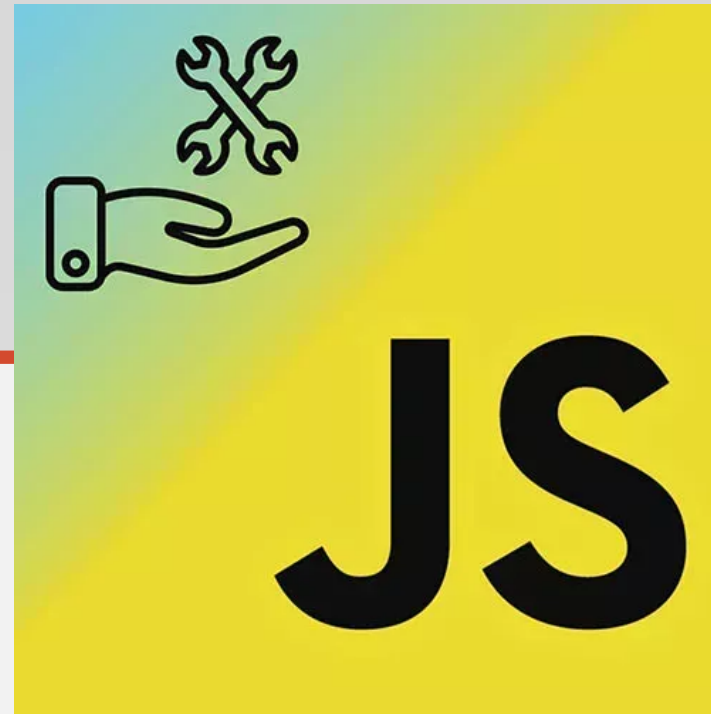
OOP experience

A web browser

Workshop Site

firtman.github.io/projs

Questions?



Introduction

Question for You...



Which version of JavaScript are you currently using?

Warning



JavaScript is not versioned

Using JavaScript in the browser

index.html



```
<script src="app.js" defer></script>
```

Using JavaScript in Node or modules

app.js



```
console.log("We don't specify a version");
```

Terminal >



```
node app.js
```

Deprecated syntax for JS versioning

It was available for Mozilla (Netscape and Firefox) only, from 1996 to 2010

index.html



```
<script language="JavaScript1.1">  
</script>
```

```
<script language="JavaScript1.8.5">  
</script>
```

Brief History of JavaScript

- **1995:** Brendan Eich created JavaScript™
- **1996:** Netscape 2 added JS™ 1.0
- **1996:** IE 3 added support for JScript™
- **1997:** JS 1.0 became an ECMA Standard, known as ECMA-Script (ES)
- **1997:** IE 4 supported ES1
- **1999:** ES3 was released
- **2000-2009:** The dark ages
- **2009:** ES5 was released
- **2015:** A new ES process started with ES6

Warning



JavaScript is a trademark of Oracle Corporation in the United States.



r/javascript · 6 yr. ago
imacpro1



Oracle Owns "Javascript", so Apple is taking down my app!

help

Just received this email from Apple about my app(HTML, CSS, JavaScript snippet editor). Looks like you can't use "JavaScript" because Oracle owns it!

Any one has any idea how to fight it or just give up?

"As you are likely aware, Oracle owns US Trademark Registration No. 2416017 for JAVASCRIPT. The seller of this iTunes app prominently displays JAVASCRIPT without authorization from our client. The unauthorized display of our client's intellectual property is likely to cause consumers encountering this app to mistakenly believe that it emanates from, or is provided under a license from, Oracle. Use of our client's trademark in such a manner constitutes trademark infringement in violation of the Lanham Act. 15 U.S.C. § 1125(a)(1)(A). In order to prevent further consumer confusion and infringement of our client's intellectual property rights, we request that you immediately disable access to this app. We look forward to your confirmation that you have complied with this request."

Important



For legal issues, most companies use ECMAScript when implementing or talking about JavaScript.

ECMAScript is free to use and it's the name we use to version the language.

ECMA-262

Aka ECMAScript aka ES

Industry association for standardizing information and communication systems



[About Ecma](#) ▾ [Publications and standards](#) ▾

ECMA-262

ECMAScript® 2023 language specification

14th edition, June 2023

This Standard defines the ECMAScript 2023 general-purpose programming language.

ECMA-262

Aka ECMAScript aka ES



Technical Committee **TC39**

Online Archives

- [ECMA-262 5.1 edition, June 2011](#)
- [ECMA-262, 6th edition, June 2015](#)
- [ECMA-262, 7th edition, June 2016](#)
- [ECMA-262, 8th edition, June 2017](#)
- [ECMA-262, 9th edition, June 2018](#)
- [ECMA-262, 10th edition, June 2019](#)
- [ECMA-262, 11th edition, June 2020](#)
- [ECMA-262, 12th edition, June 2021](#)
- [ECMA-262, 13th edition, June 2022](#)

ECMAScript

- It's a standard for scripting languages
- TC-39 is its technical committee
- JavaScript used by browsers or Node are ECMAScript engines
- Other engines: ActionScript, JScript.NET
- Since ES2015 (or ES6) we have one version published per year
- As developers, we can't specify which version we want to use, it's up to the engine where the script is executed

Warning



If you use syntax of an ES version that is not supported on the engine running it you may get a syntax error or runtime exception.

Important



To know the ECMAScript version that your engine uses:

- Node: check node.green
- Browsers: check caniuse.com/ecmascript

TC-39 Process

- Every proposal goes through a process
 - **Stage-0:** Strawperson
 - **Stage-1:** Under Consideration
 - **Stage-2:** Draft
 - **Stage-2.7:** Approved
 - **Stage-3:** Candidate
 - **Stage-4:** Complete, ready for ES-next
- Backward compatibility is forced
- Most changes are sugar syntax from the previous version

Modern Versions of ECMAScript

- From ES1 to ES5 versions were using numbers
- From ES6, also known as ES2015, the year of release is also used as version.
- From ES6 TC-39 has an annual version process so there will be an ES version for every year since 2015.
- While ES14 do exist as a version number, the community uses the year version from ES7, so it's ES2023.

Warning



When we talk about ECMAScript we
are not talking about platform APIs

Warning



Most platform APIs are defined by the W3C, OpenJS Foundation, and other organizations.

Important



ECMA Internationalization API under the ECMA-402 is a separate ECMAScript-related spec separated from the core spec.

ES.Next

- It's a non-official name to talk about features that will be in next version of ES
- Stage-3 or Stage-4
- It's almost guaranteed they will be implemented in the spec
- Some browsers may already support some of those abilities

To use modern ES syntax on older engines



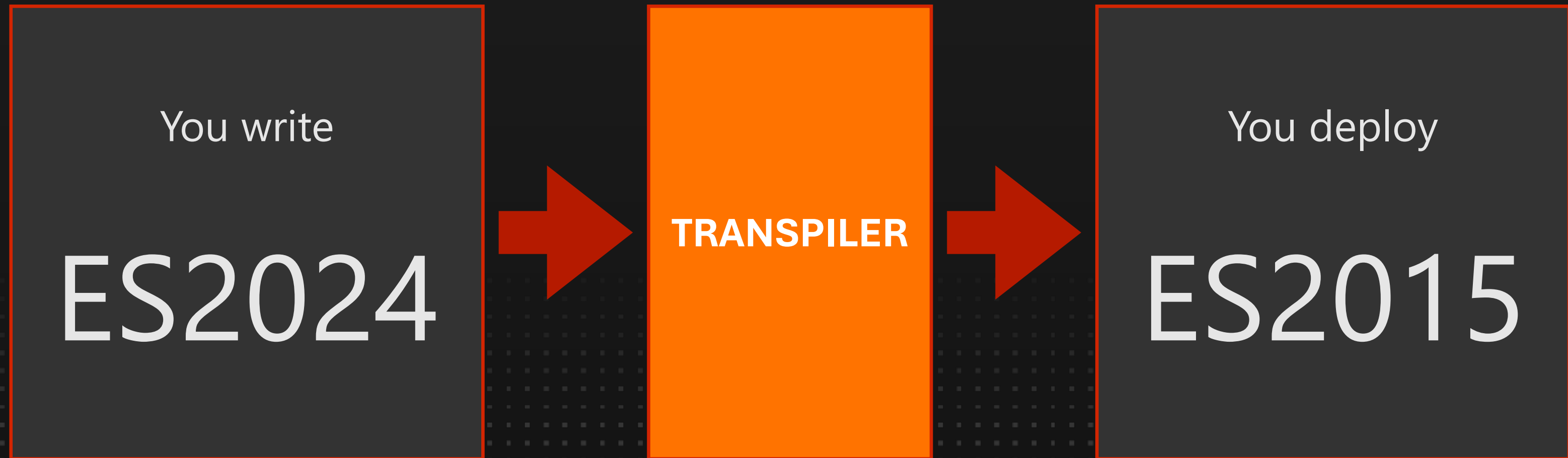
Polyfills

Transpilers

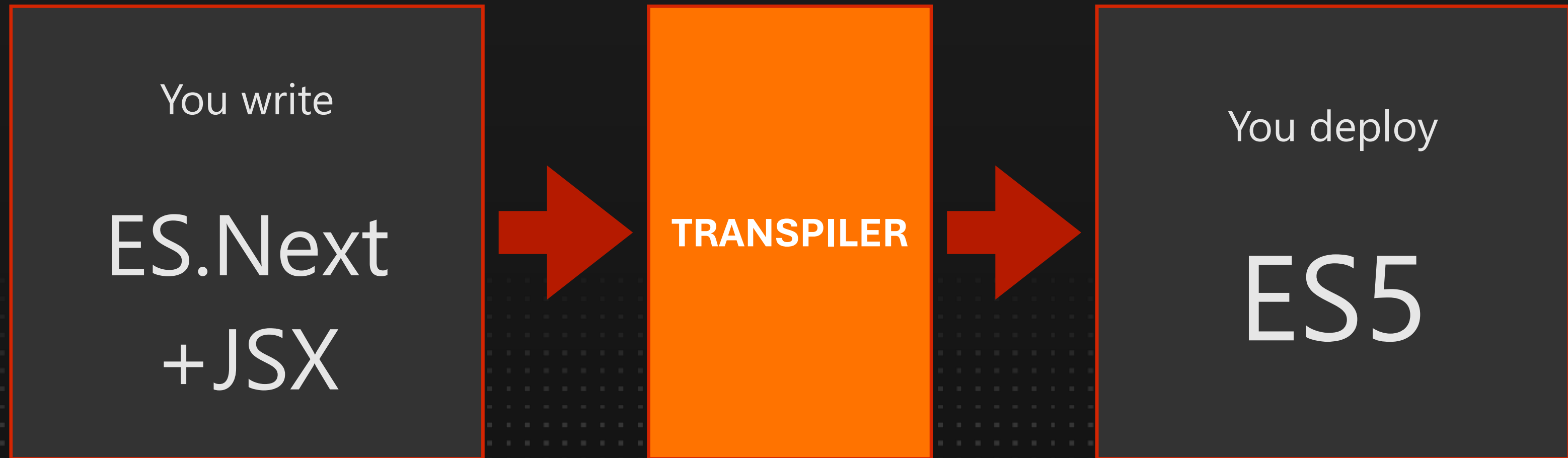
Transpilers

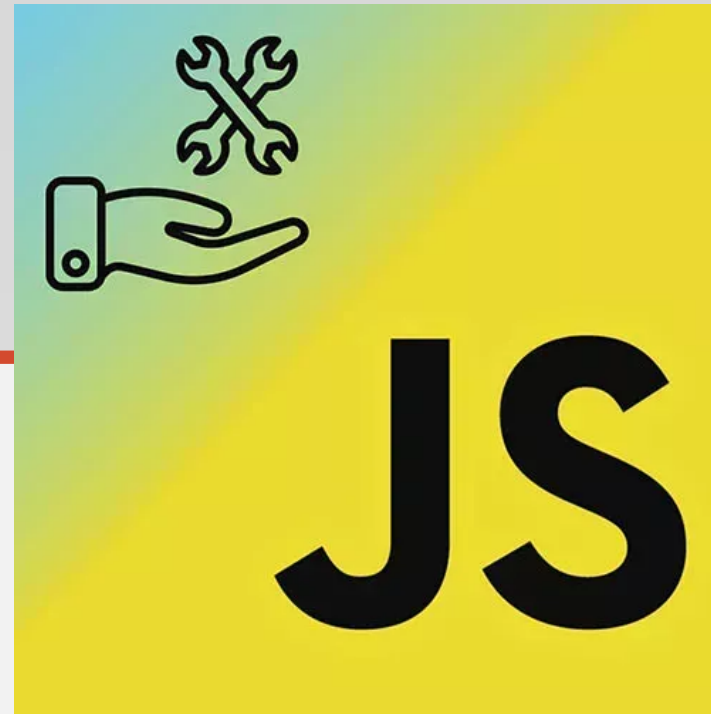
- They convert ES modern code into older ES code, such as ES5 or ES6 (2015) including polyfills, when needed.
- The most common solutions:
 - Babel
 - TypeScript
 - ESBuilder
- They may use plugins
 - ES.Next
 - JSX and non-standard supersets

With a Transpiler



With a Transpiler





Recap of ES2015 (ES6)

ES6 or ES2015

- It was one of the major upgrades to the language
- It's safe to use it on every browser today
- Class syntax for OOP
- Block scoped variable definitions
- ES Modules
- Arrow functions =>
- Promises
- And many more features!

Warning



We will leave some advanced ES6 topics for later, organized by topics

Definition

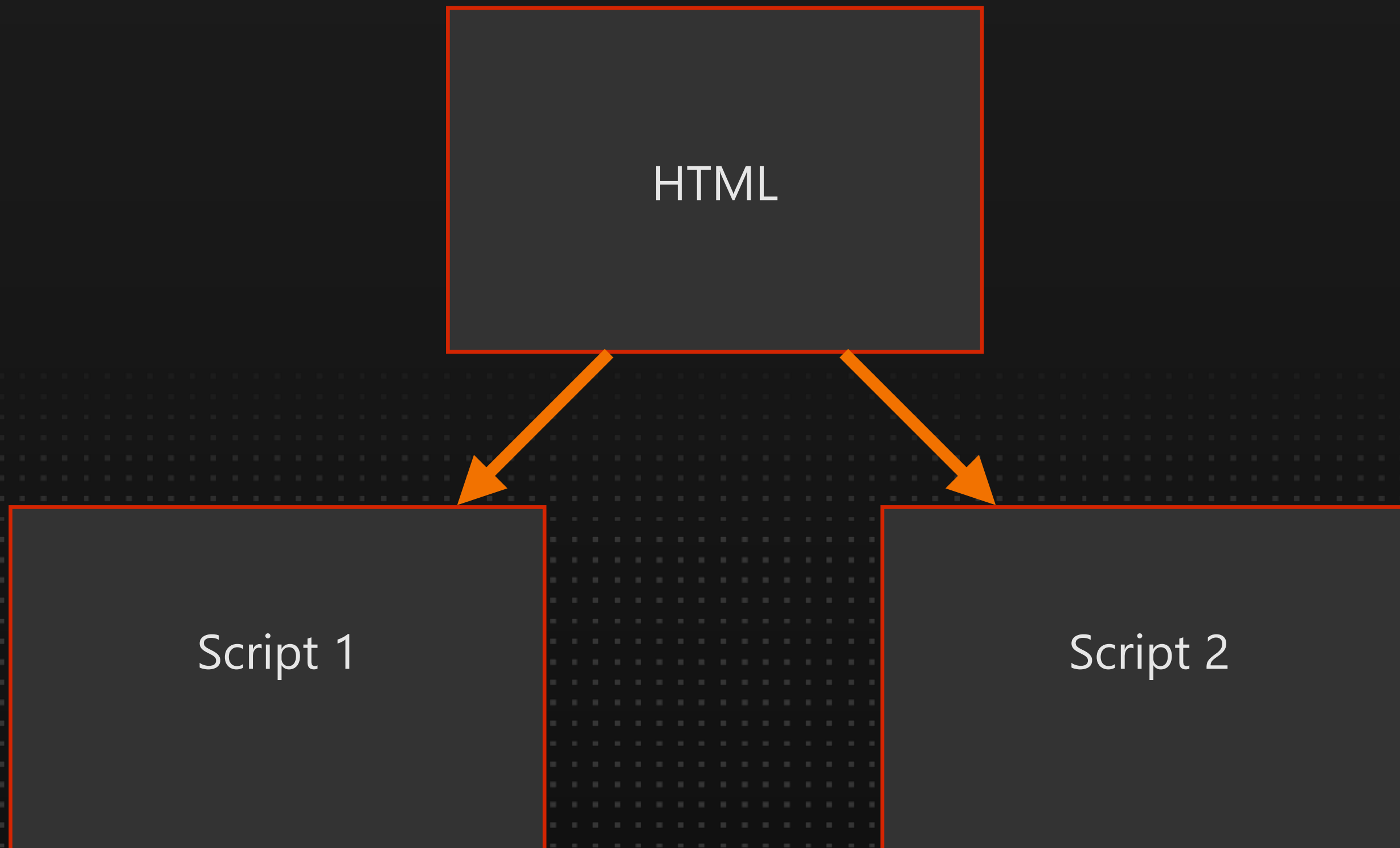
ES Modules



Standardized way to organize and reuse JavaScript code across different files using import and export statements for better modularity and maintainability.

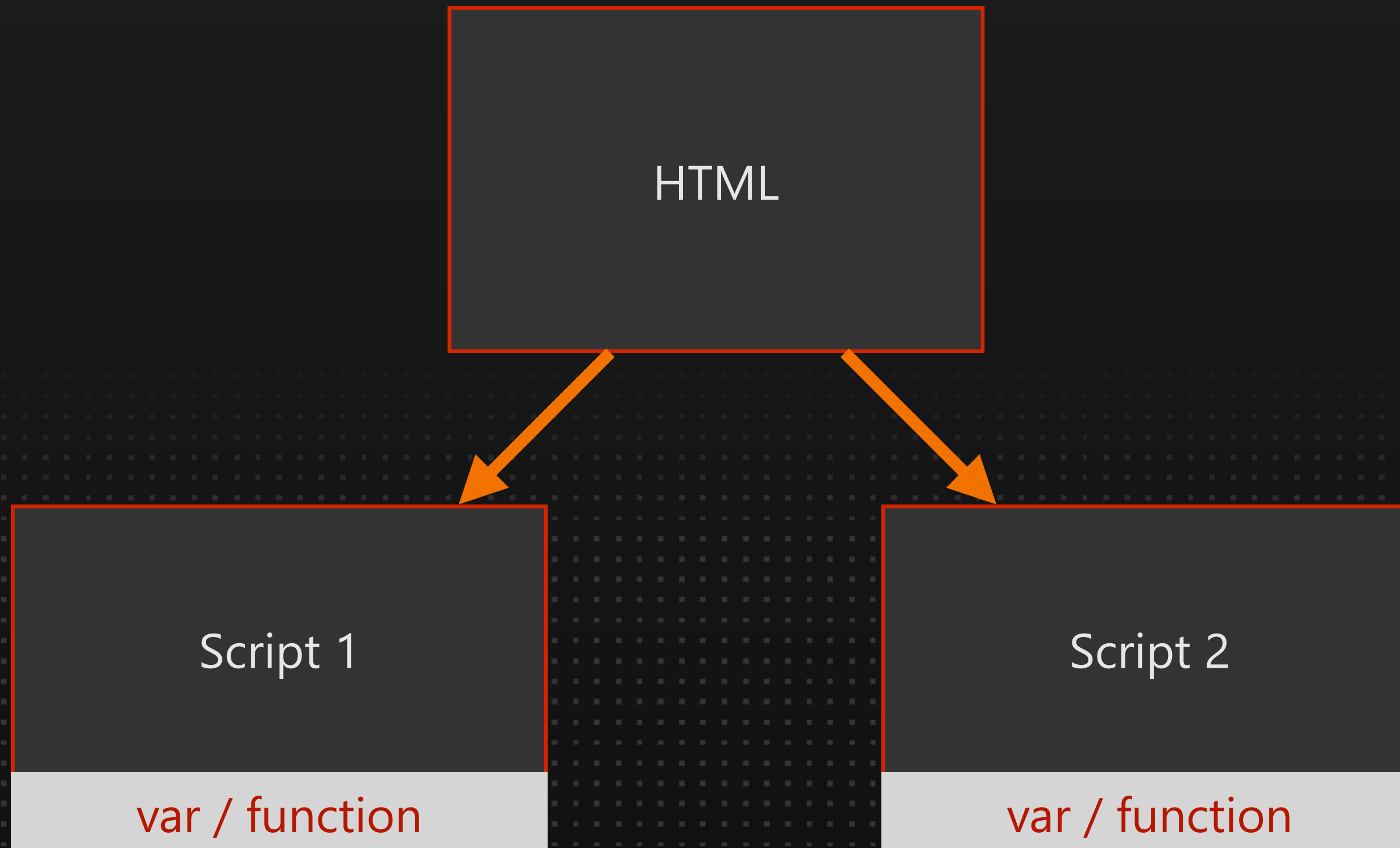
Working with Separate JavaScript Files

Historically, within the browser all the items were available globally to all files



Working with Separate JavaScript Files

Historically, within the browser all the items were available globally to all files



Working with different files in classic ES5 mode

- They use the same global context
- One script can't include or load other scripts (worker exception)
- Can't modularize behavior or data
- Node.js used the CommonJS pattern to emulate modules

Working with Separate JavaScript Files

Historically, within the browser all the items were available globally to all scripts loaded

script1.js



```
printUser(user);
```

script2.js



```
var user = {  
  firstName: "Brendan"  
}  
  
function printUser(u) {  
  console.log(u.firstName);  
}
```

ES Modules

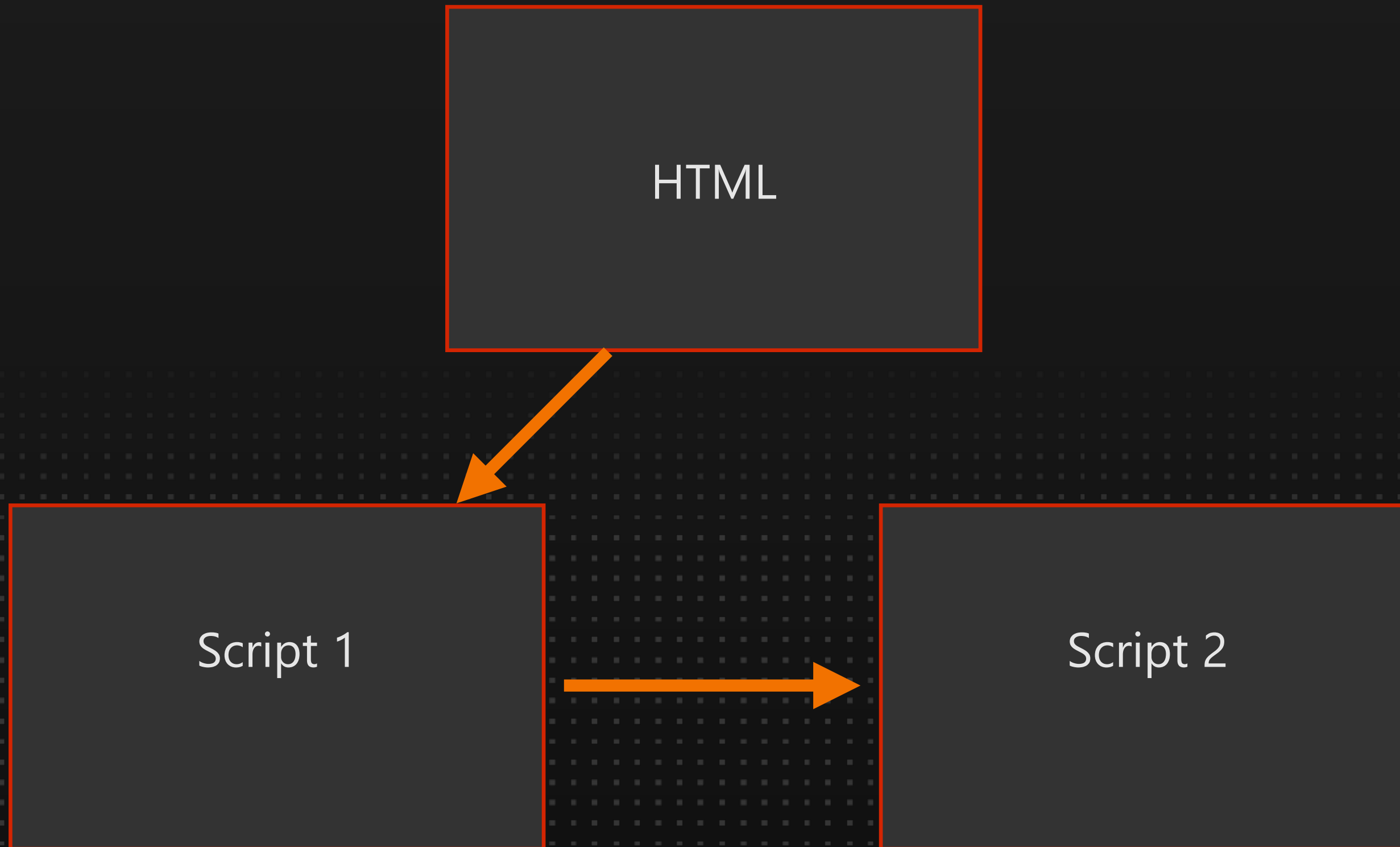
- ES6 included Modules
- They work as a container isolated from the global object (`window`, `global`, `self`)
- For node, it's replacing CommonJS modules using `require()`.
- Each module works in a separate file
 - For the browser ".js"
 - For node, ".mjs" by default
- The global scope creates a module import tree as soon as it's parsing modules

ES Modules

- A module can export items:
 - Variables
 - Functions
 - Class declarations
 - Objects
- It can have one default import
- A module can import other module's items totally or partially

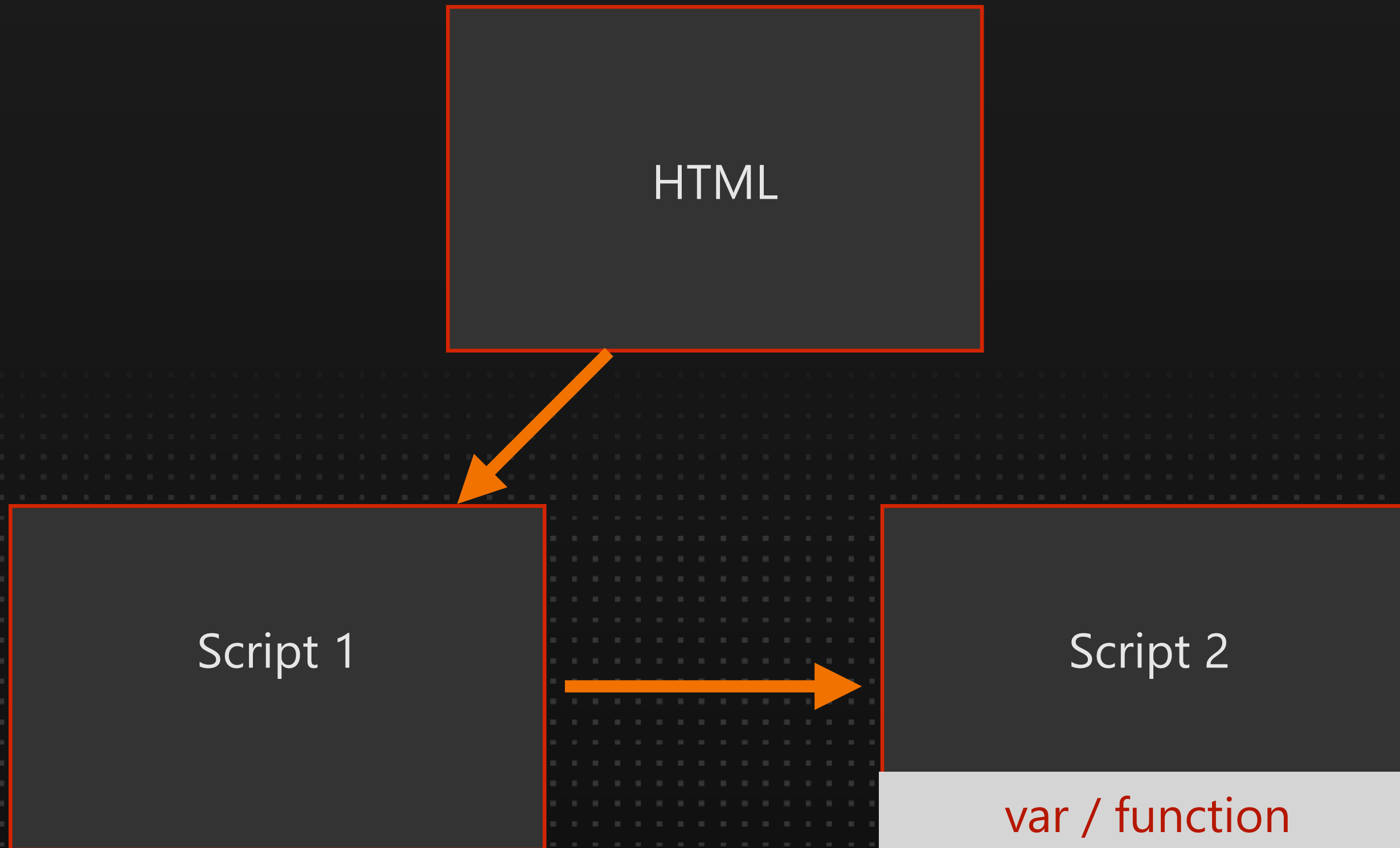
Working with Modules

The HTML `<script>` tag needs `type="module"`



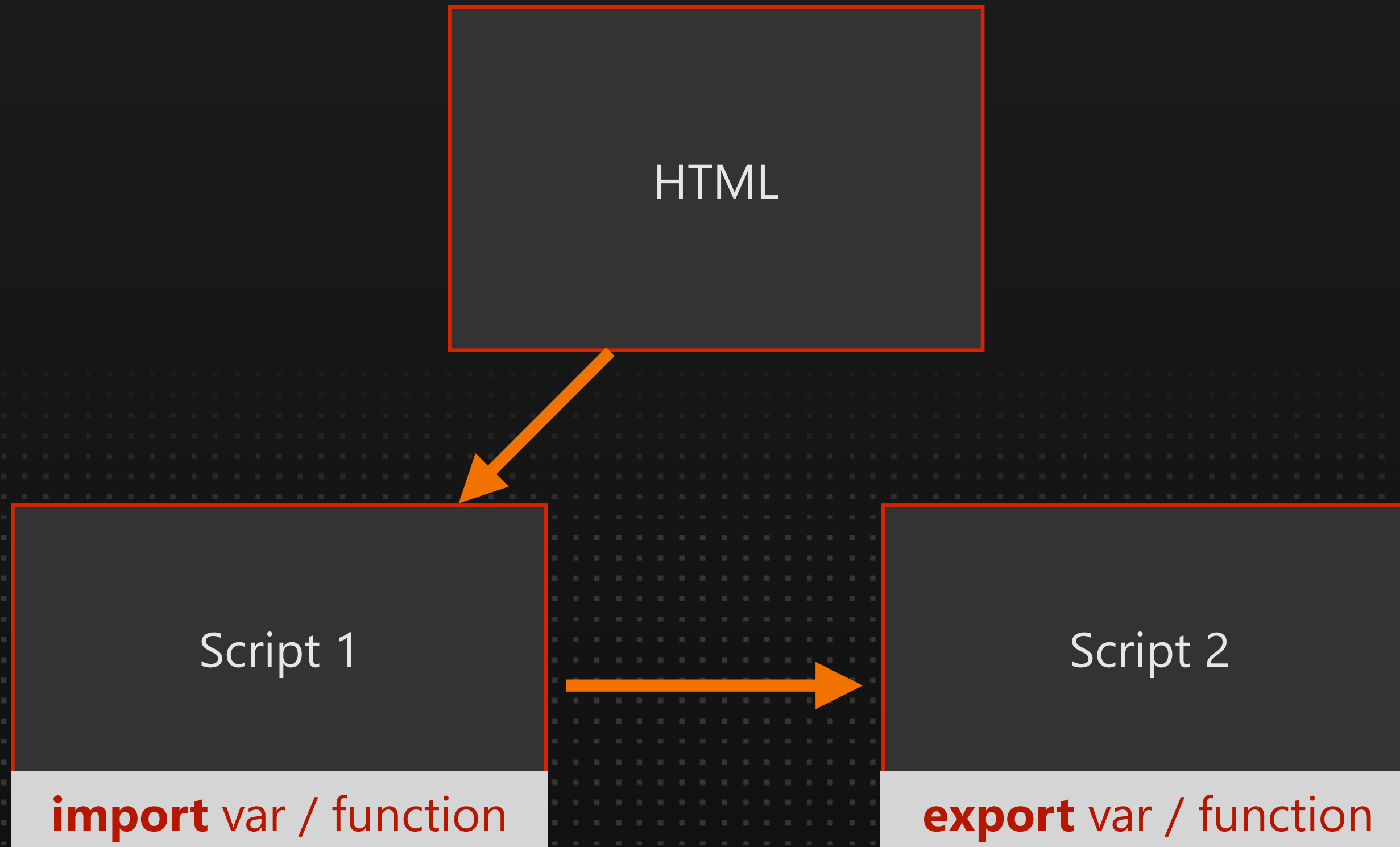
Working with Modules

The HTML `<script>` tag needs `type="module"`



Working with Modules

The HTML `<script>` tag needs `type="module"`



Working with Modules

From ES6 using `<script type="module" src="script1.js">`

script1.js



```
printUser(user);
```

script2.js



```
var user = {  
  firstName: "Brendan"  
}  
  
function printUser(u) {  
  console.log(u.firstName);  
}
```

Working with Modules

From ES6 using `<script type="module" src="script1.js">`

script1.js



~~printUser(user);~~ **✗ Error**

script2.js



```
var user = {
  firstName: "Brendan"
}

function printUser(u) {
  console.log(u.firstName);
}
```


Working with Modules

From ES6 using `<script type="module" src="script1.js">`

script1.js



~~printUser(user);~~ **✗** Error

script2.js



```
export var user = {  
  firstName: "Brendan"  
}  
  
export function printUser(u) {  
  console.log(u.firstName);  
}
```

Working with Modules

From ES6 using `<script type="module" src="script1.js">`

script1.js



```
import { user, printUser } from  
  './script2.js';
```

```
printUser(user);
```

script2.js



```
export var user = {  
  firstName: "Brendan"  
}
```

```
export function printUser(u) {  
  console.log(u.firstName);  
}
```

Working with Modules

From ES6 using `<script type="module" src="script1.js">`

script1.js



```
import Services from  
  './script2.js';  
  
Services.printUser(Services.user);
```

script2.js



```
var user = {  
  firstName: "Brendan"  
}  
  
function printUser(u) {  
  console.log(u.firstName);  
}  
  
default export { user, printUser }
```

Warning



When importing from modules path must start with `"/"`, `"/."` or `"/.."` and you should always use full URL (including in most cases `".js"`)



Lab time

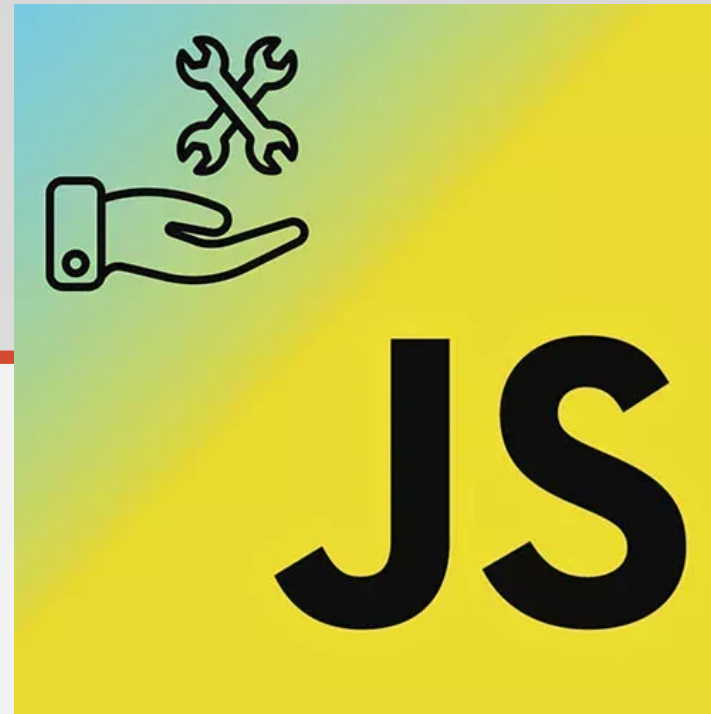
Workshop Site

firtman.github.io/projs

Time to see what's new after ES6 (ES2015)



We won't cover features by ES version but by category, explaining for each one from which version it's available (ES2016-ES2024).

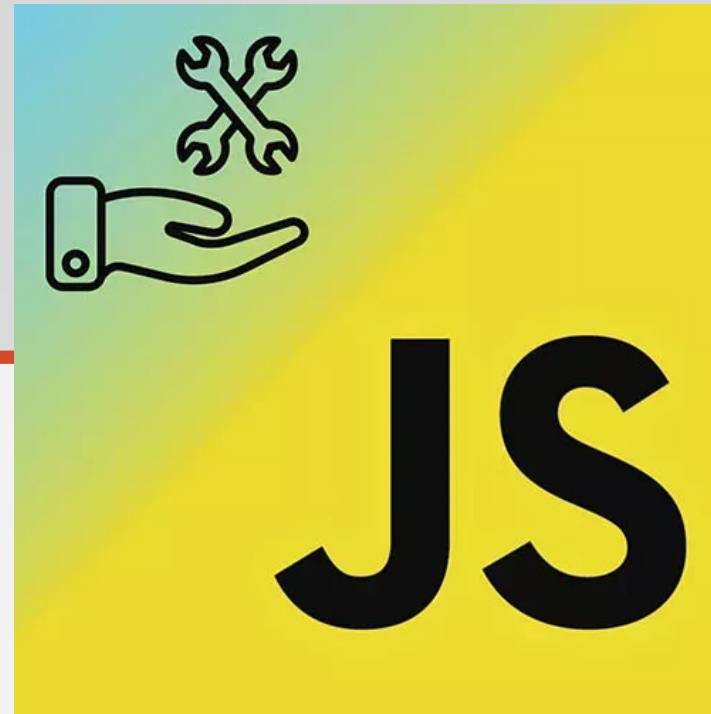


Language Enhancements

Language Enhancements

- Small changes
- GlobalThis
- Optional Catch Binding
- Function toString
- New Operators
- Class Declaration
- Object
- Strings
- Numbers

Lab time



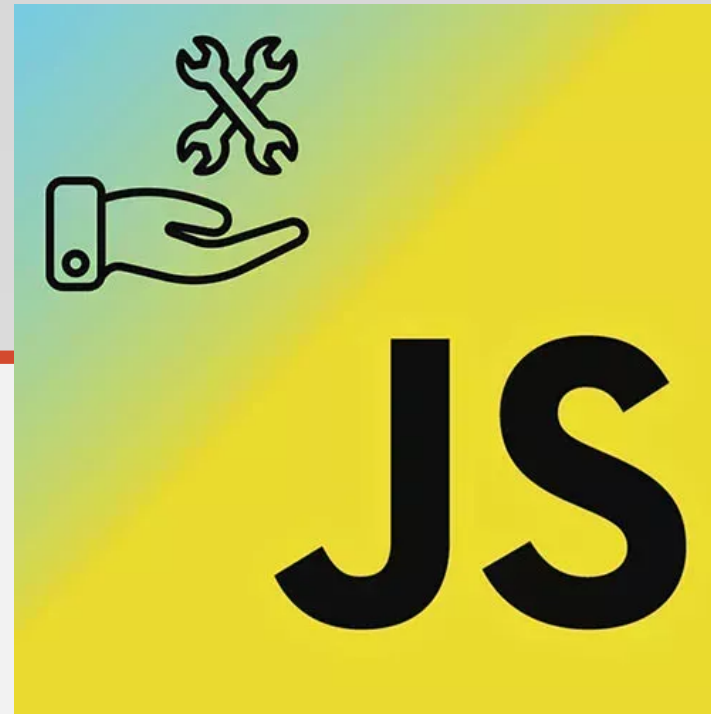
Array and Collection Enhancements

Array and Collection Enhancements

- Iterators and for..of
- Generators
- Array Methods
- Change Array by Copy
- Set and Maps
- Typed Arrays



Lab time

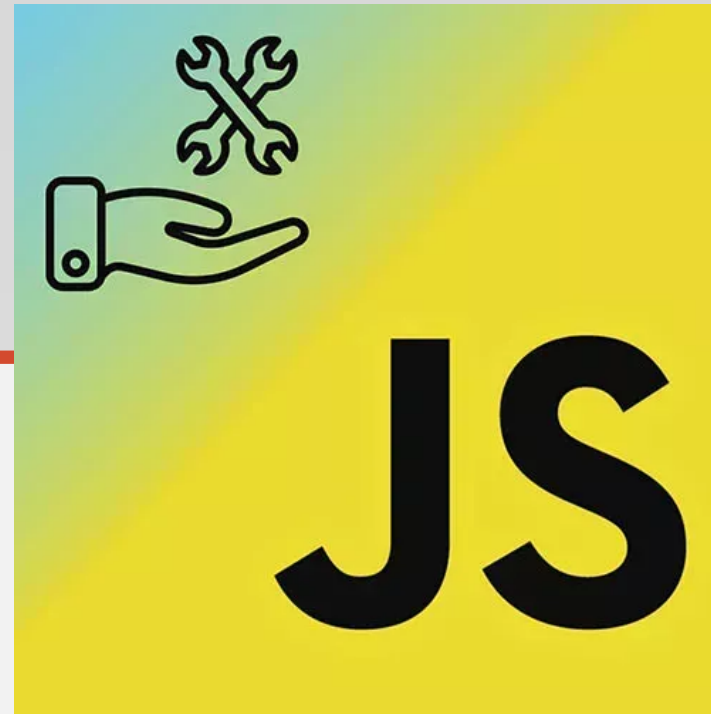


Asynchronous Programming

Asynchronous Programming

- Promises Recap
- Async/Await
- Promises Improvements
- Asynchronous Iteration
- Top-level await
- Advanced
 - Atomics
 - SharedArrayBuffer

Lab time

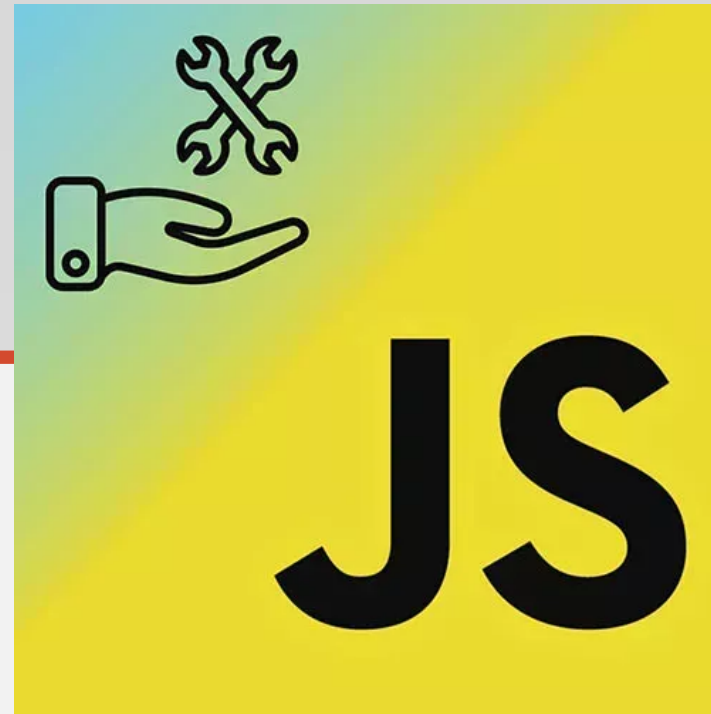


Advanced Techniques

Advanced Techniques

- Some small changes
- Dynamic Import
- Proper Tail Calls
- Proxies and Reflect API
- Tagged Templates
- WeakRefs and FinalizationRegistry
- Regular Expressions Enhancements

Lab time

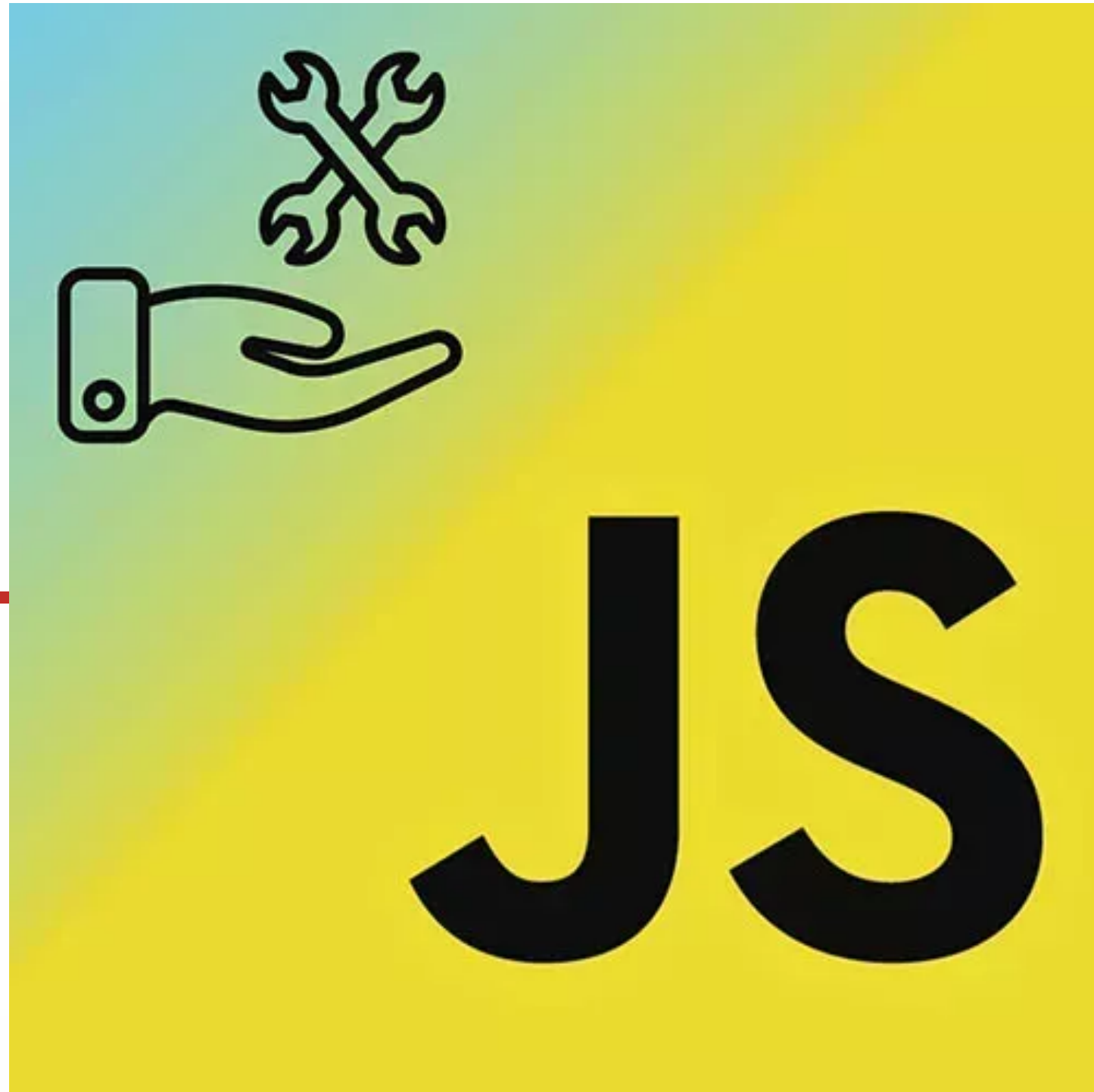


Recap

Recap

- JavaScript and ECMAScript
- We version ECMAScript every year
- Syntax sugar
- Transpilers
- ES6 Review
- Language enhancements
- Collections and Array
- Asynchronous Programming
- Advanced Topics

Thanks!



Maximiliano Firtman

firt.dev

 x.com/firt

 linkedin.com/in/firtman

 github.com/firtman