

Web Programming

Week 12

"Well-structured software is easy to write and to debug, and provides a collection of modules that can be reused to reduce future programming costs."

John Hughes



Retrospective

Quiz

Homework (excel)

Other

Agenda

ES 6 Module System

Special use cases for modules

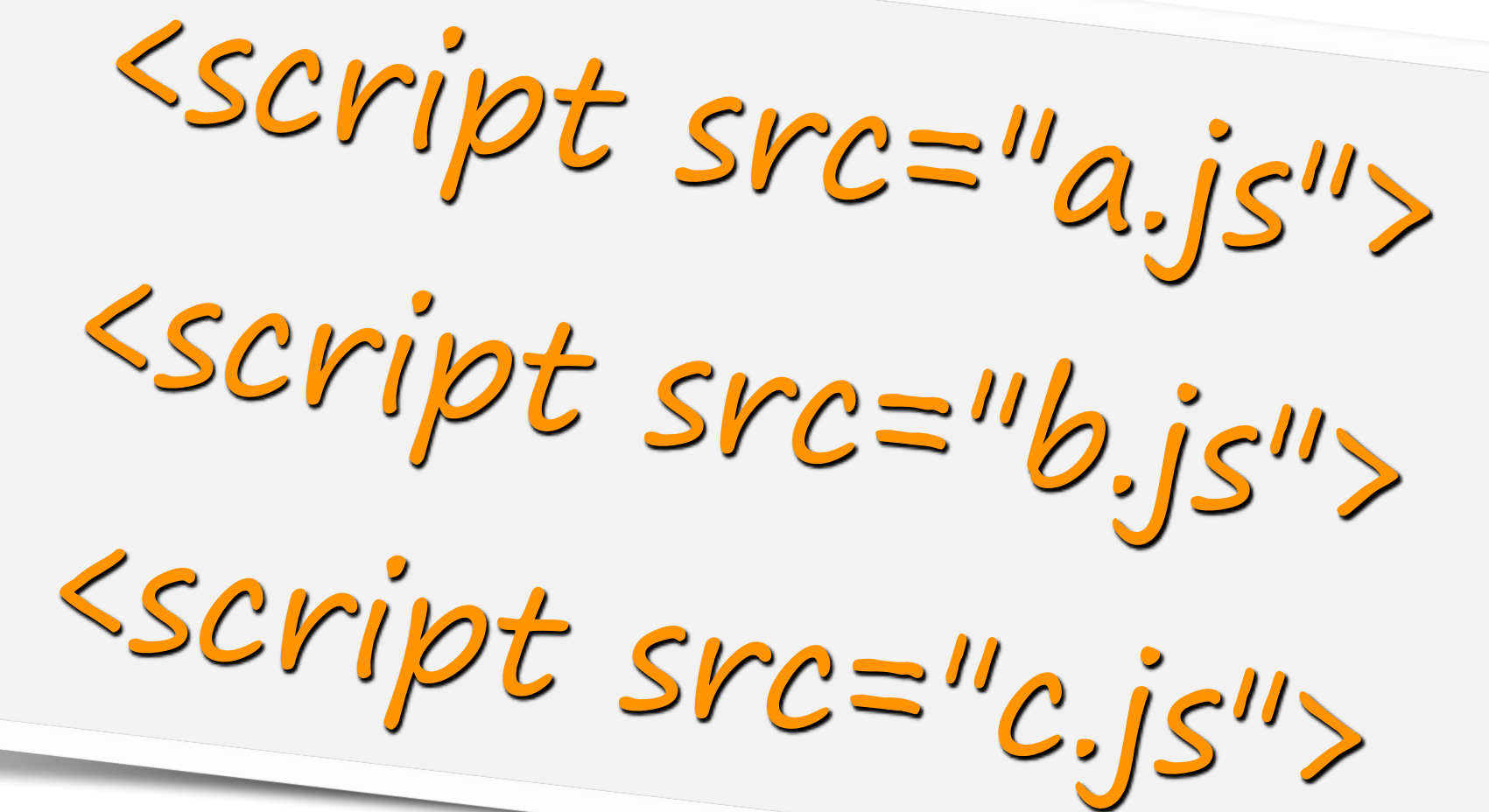
Why Modules?

Organize Code

Clear Dependencies

Avoid errors:

Globals, Scoping, Namespace



```
<script src="a.js">  
<script src="b.js">  
<script src="c.js">
```

Why Modules?

Distinguish:

How I want to edit the code

How I want to deliver the code

ES6 Modules are not

Packages (those have versions)

Dependencies, Libraries, Releases

Units of publication

Objects

Distinction

Package Manager

webpack, npm, bower, yarn, ...

Build Tools

webpack, npm, grunt, gulp, ...

Distinction

Legacy module systems

CommonJS, AMD, UMD, ...

Legacy Module Loader / Bundler

RequireJS, SystemJS, browserify, ...

A Module is code that

uses "import" oder "export"

==

imports or exports modules

Modules are async

best use URI format

implies "defer"

`<script src="./my.js" type="module">`

`import("./my.js").then(mod => ...)`

invasive, transitive impact

Import Variants

always explicit

```
import "module-name";  
import defaultExport from "module-name";  
import * as name from "module-name";  
import { export } from "module-name";  
import { export as alias } from "module-name";  
import { export1 , export2 } from "module-name";  
var promise = import("module-name");
```


Export Variants

always explicit

```
export { name1, name2, ..., nameN };  
export function FunctionName(){...}  
export const name1, name2, ..., nameN; //or let  
export class ClassName {...}  
export default expression;  
export { name1 as default, ... };  
export * from ...;  
export { name1, name2, ..., nameN } from ...;
```

Impact I

implicit "use strict"
exports are read-only !

no Global object, no Global "this",
no Global hoisting

implicit "defer" mode,
=> document.writeln is no longer useful

Impact II

Modules are Namespaces

Modules are Singletons

SOP

Single Origin Policy

Modules are subject to the SOP

Problem zur Entwicklungszeit:
das File System ist ein null origin

Useful Tools

don't forget to set back!

Developer Mode (suppress SOP)

Local Webserver

disable cache!

Bundler (Rollup, Parcel, Webpack, ...)

Start Browser in Debug mode

Let's code

Refactor in small slices

Once you start, it quickly ripples through the whole codebase

Resources

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import>

http://exploringjs.com/es6/ch_modules.html



Work at Home

Refactor your Toolkit Repository to use modules