Modules in NodeJS:
IIFEs, CommonJS and ES Modules

Markus Hösel
@iamhaezl

flaws in javascript

All Images Videos News Maps More Settings Tools

About 21.500.000 results (0,43 seconds)

10 design flaws of JavaScript

- Not suitable for large projects. **JavaScript** doesn't have namespace, it's hard to be modular, there is no standard for putting codes in multiple source files. ...
- Small standard library. ...
- null and undefined. ...
- Global variable. ...
- The insertion of semi-colon at line end. ...
- + operator. ...
- NaN. ...
- The difference between array and object.

More items... • Nov 29, 2012

10 design flaws of JavaScript | Pixelstech.net

https://www.pixelstech.net/article/1354210754-10-design-flaws-of-JavaScript

About this result Feedback

Using the Global Namespace & IIFEs (Immediately Invoked Function Expressions)

<Examples>

`01_namespace_pollution`

`02_wrapper_function`

`03_iifes`

Disadvantages of using the Global Namespace & IIFEs

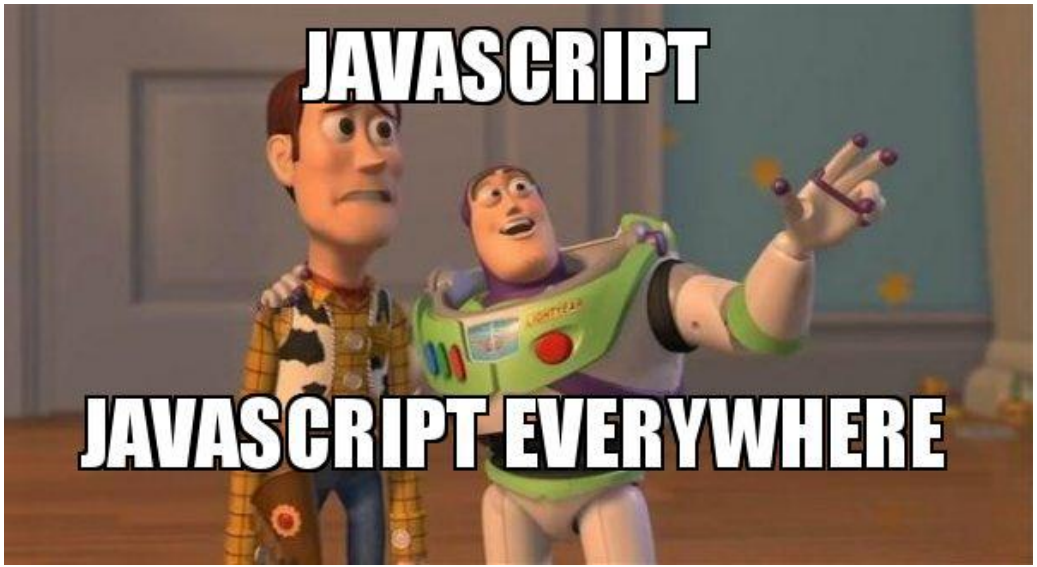
- Global Namespace Pollution
- Order of script tags matters

Advantages of a module system

- Reusability
- Composability
- Leverage
- Isolation
- Organisation

The rise of Server Side Javascript

- NodeJS
- CommonJS
- NPM



CommonJS

CommonJS is a (growing) collection of standards, including

- Modules
- Binary strings and buffers
- Charset encodings
- Binary, buffered, and textual input and output (io) streams
- System process arguments, environment, and streams
- File system interface
- Socket streams
- Unit test assertions, running, and reporting
- Web server gateway interface, JSGI
- Local and remote packages and package management

CommonJS is not only implemented by NodeJS, other examples are

- RingoJS, MongoDB, CouchDB, Wakanda, ...

“Forget CommonJS. It's Dead. We are ServerSide Javascript.”

https://github.com/nodejs/node-v0.x-archive/issues/5132#issuecomment-15432598

NodeJS & CommonJS Modules

<Example>

`04_commonjs_default_export`

CommonJS

Main disadvantages

- loads modules synchronous
- not supported by browsers

Bundlers

Bundlers, such as Browserify, Webpack or Rollup, bundle all modules into one file, which is then included in your index.html.

AMD, UMD, RequireJS

- The Asynchronous Module Definition (AMD) API specifies a mechanism for defining modules such that the module and its dependencies can be asynchronously loaded

- RequireJS is the most popular implementation of AMD

- Universal Module Definition (UMD): Standard which brings together CommonJS and AMD

ES Modules to the rescue

ES6 or ECMAScript 2015 defines a Module System named ES Modules

- supports synchronous and asynchronous loading
- supports tree shaking / dead code elimination
- does not support dynamic imports
  - proposal currently at Stage 3 of TC39 process
- supported by all modern browsers

ES Modules

<Examples>

05_es_modules_native_browser

06_es_modules_transpiled_to_commonjs

07_es_modules_transpiled_to_amd

08_es_modules_transpiled_to_umd

09_es_modules_native_nodejs_mjs

ES Modules in NodeJS

- Experimental support was added in NodeJS 8.5.0 in Sept 2017
  - via flag `--experimental-modules`
  - https://github.com/nodejs/node-eps/blob/master/002-es-modules.md
- Modules Team was formed by the NodeJS Committee in Feb 2018
  - https://github.com/nodejs/modules
- New implementation is added in Node 12 which was released in Apr 2019
  - https://github.com/nodejs/modules/blob/master/doc/announcement.md
- Experimental flag will be dropped when Node12 starts LTS in Oct 19

.mjs - the Python3 of NodeJS?

.js, .cjs and .mjs files

- .mjs files are always loaded as ES Modules
- .cjs files are always loaded as CommonJS Modules
- .js files are loaded as:
  - ES Modules when the nearest package.json contains a top-level field “type” with a value of “module”
  - CommonJS Modules when the nearest package.json contains a top-level field “type” with a value of “commonjs"
  - CommonJS Modules when no top-level field is present in the nearest package.json

Interoperability

- CJS Modules can be included in ES Modules through the “import” stmt
    - currently only the “default export” is supported for CJS Modules

- ES Modules can be included in CJS Modules through import() expression within an async function

Interoperability

<Examples>

`10_es_modules_native_nodejs_mjs_cjs`

Thank you!

Presentation and Source Code on Github :)

https://github.com/haezl/modules-in-nodejs