Day-3

javascript development, without module standards,

problem(s)

- global naming collisions

> 📌 solution : using self executable function with single-global-object

- loading dependency modules in order is manual and difficult

---

Solutions:

> 📌 using javascript module standards and loaders

popular module standards

1. commonJS （ third-party community ）
2. ES modules （ ECMAScript community ）

📌 if you javascript code to Node.js runtime, we must use commonJS module standard

---

commonJS (from third-party community)

rules

- every .js file is module
- all variables and functions in a .js file are private by default
- to abstract/export any member(s) (var / function) to other module, use 'module.exports' object
- to require/import any module , use 'require()' function
- group related modules as package with description file - package.json

📌 imp-note : Node.js runtime already implemented this standard as default module standard,
but not not browser

---

how to use commonjs module standard javascript code in browser?

use module bundlers

📌 browserify or webpack

ES Modules **(from ECMAScript community) from ES6**

---

rules

- every .js file is module
- all variables and functions in a .js file are private by default
- to abstract/export any member(s) (var / function) to other module, use 'export' keyword
- to require/import any module , use 'import' keyword
- group related modules as package with description file - package.json

---

NPM (Node Package Manager)

- command line tool

why we need ?

- to publish or install javascript packages from NPM-repository (public | private)