

CMSC388B

Web Application Development with JavaScript



Modules

Department of Computer Science
University of MD, College Park

Slides material developed by Ilchul Yoon, Nelson Padua-Perez

<script> defer attribute

- **defer** attribute
- For external scripts
- **Does not block the page loading**
 - Tells the browser to continue with the page and loads the script “in background”, then runs the script when it loads
- `<script defer src=“...”></script>`

Modules

- Split features into multiple files (modules)
- A module usually contains a class or a library of functions
- A module is just a file. One script is one module
- Two types of JS module systems
 - **CommonJS**: implemented by Node.js
 - » “require()”
 - **ECMAScript Harmony (ES6)**: Used for both server/client
 - » “import”

CommonJS Modules

- Define functions, classes, and constants in a .js file, .cjs file
- Use **module.exports = { }** to export entities
 - **Example: module.exports = { DEFAULT, add, multiply };**
- Use **require** in the file you would like to use the module
 - **Example: const utils = require("./utils")**
 - » Assuming the file **./util.js** has DEFAULT constant and two functions: **add** and **multiply**
- **Example: CommonJSModules**
 - driver1.js, driver2.js, driver3.js

ES6 Modules

- Define functions, classes, and constants in a .js, .mjs file
- Use **exports = { }** to export entities
 - **Example:** **exports = { DEFAULT, add, multiply };**
- Use **import** in the file you would like to use the module
 - **Example:** **import * as utils from "./utils.mjs";**
 - » Assuming the file util.mjs has DEFAULT constant and two functions: **add** and **multiply**
- **Example:** ES6Modules
 - **NodeExamples** folder (to run these examples, package.json must have the entry "type":"module"). Try removing it
 - » driver1ES6.js, driver2ES6.js, driver3ES6.js
 - **BrowserExample** folder
 - » You need to run this example using a web server
 - Place files in htdocs or use VS Code Live Server
 - Try opening the file without a server; check the console

Extensions (.js, .mjs, .cjs)

- .mjs - extension for ES6 modules for use with a Node.js application
 - .mjs files are written in JavaScript and may use the .js extension outside of the Node.js context
- **How Node.js will treat files:**
 - .cjs files as CommonJS modules
 - .mjs files are ES6 modules
 - .js files based on the default module system. CommonJS is the default for Node.js unless package.json has the following directive:
 - » "type": "module"
 - » Try running examples after removing from package.json "type": "module"
 - If you rename the file with a .mjs extension it will work (without "type": "module")

Modules in Browsers

- If a file is used in a browser, you must tell the browser that a script should be treated as module, by using the attribute

<script type="module">

- Module scripts are always deferred
 - same effect as **defer** attribute