Moduler

Vad är moduler bra för?

- ✓ När program växer, och all kod finns i en enda fil, ökar risken för att de förlorar struktur och blir omöjliga att förändra och utveckla vidare.
- ✓ Genom att dela upp ett programs kod i flera filer, moduler, undviks många problem, och nya möjligheter skapas.
- ✓ En modul ett litet delprogram som kan ange vilka andra delprogram den beror av, och framför allt vilken funktionalitet den erbjuder andra moduler att använda.

ECMAScript-moduler

- JavaScript har sedan version ECMAScript 6 (2015) inbyggt stöd för moduler, så kallade ECMAScript-moduler (även ES-moduler eller ESM).
- En modul kan exportera värden genom att använda det reserverade ordet `export`.

```
// my-greeting.js
function bye () {
  console.log('God bye!')
}

export function hello () {
  console.log('Hello, World!')
}
```

Exporteras inte! Är endast tillgänglig i modulen.

Namngiven export (*"named export"*). Kan anropas utanför modulen.

- En modul kan ladda en annan modul den är beroende av med det reserverade ordet `import`.

```
// app.js
import path from 'path'
import { fileURLToPath } from 'url'

import * as myGreeting from './my-greeting.js'

import { hello } from './my-greeting.js'
```

Namnrymdsimport (*"namespace import"*). Importerar allt som exporterats.

Namngiven import (*"named import"*). Importerar endast funktionen 'hello'.

Exempel

```
// my-greeting.js
function bye () {
  console.log('Bye!')
}

export function hello () {
  console.log('Hello, World!')
}

// app.js
import * as myGreeting from './my-greeting.js'

myGreeting.hello() // OUTPUT: Hello, World!

myGreeting.bye() // ERROR: TypeError: myGreeting.bye is not a function
```

- ✓ Endast exporterade funktioner kan anropas!

CommonJS – ett ”utdaterat” sätt att hantera moduler

- ✓ Ett av de populära modulsystemen innan ESM är/var CommonJS.
  - Fortfarande vanligt framför allt inom Node.js-plattformen där implementationen av ESM tills nyligen var kategoriserad som experimentell.

```
// my-greeting.js
function bye () {
  console.log('Bye!')
}

function hello () {
  console.log('Hello, World!')
}

module.exports.hello = hello

// app.js
const myGreeting = require('./my-greeting.js')

myGreeting.hello() // OUTPUT: Hello, World!

myGreeting.bye() // ERROR: TypeError:  myGreeting.bye is not a function
```

Så aktiverar du ECMAScript-moduler i Node.js

- Node.js behandlar JavaScript-kod som CommonJS-moduler som standard.
- Du talar om för Node.js att du vill arbeta med ECMAScript-moduler genom att i `package.json` lägga till fältet `type` med värdet satt till `module`.
  - ~~Ge JavaScript-filer filändelsen `.mjs`.~~
- Enklast skapa du filen `package.json` med `npm init -y`, och lägger därefter till fältet `type`.

```
{
  "name": "my-awesome-app",
  "version": "1.0.0",
  "description": "",
  "type": "module",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "Ellen Nu <en999zz@student.lnu.se>",
  "license": "MIT"
}
```

Talar om för Node.js att använda ECMAScript-moduler.

npm – en samling för paketerad kod

- ✓ npm-registret innehåller paket. Ett paket kan innehålla moduler.
- ✓ Vilka paket en applikation använder beskrivs i filen `package.json`.
  - Under `dependencies` och `devDependencies` listas de paket, och dess versioner, applikationen använder och är beroende av.
  - Paket installeras med kommandot `npm install PAKETETS_NAMN`.
- ✓ Paket som installeras placeras i katalogen `node_modules`.
  - Saknas katalogen, och därmed paket applikationen är beroende av, kan paket installeras med `npm install`.