<WA/>
2025

JavaScript packages: example with dates

“The” language of the Web

Enrico Masala
Antonio Servetti

Example: Day.js Package

DAY.JS https://day.js.org/

- Install (from command line)

```
# initialize the package manager files in the project
# if not already done (choose a name and default for the rest)
npm init

# download from registry, add to project package list
# make it available to the scripts in the project
npm install dayjs
```

package.json

```
{
  "name": "my-project",
  "version": "1.0.0",
  "main": "index.js",
  . . .
  "dependencies": {
    "dayjs": "^1.11.10"
  }
}
```

Folder Structure after running npm

```
my-project
├── node_modules
├── package.json
├── package-lock.json
└── index.js
```

- `my-project` is the project root
- `node_modules` is the folder where packages are installed. This is automatically managed/reconstructed by npm, do not touch!
- `package.json` contains (also) the list of packages needed by the project, with their minimum version
- `package-lock.json` contains the list of packages actually installed in the project, with more details (version, package hash)
- `index.js` is the code of the project
  - **Develop here!**
  - Insert the `require()` statement here to use the package

Example: Day.js Package usage in node

- In the Javascript file, after the package has been installed

index.js

```
// import (using name of my choice)
const dayjs = require ('dayjs');

// use (depends on the specific package)
let now = dayjs();
console.log(now.format());
```

Day.js main goals

- Compatible with moment.js (most used date library until a few years ago)
  - But very small (2kB) compared to moment.js
- Works in nodejs and in the browser
- All objects are *immutable*
  - All API functions that modify a date, will always return a new object instance
- Localization support
- Plugin system for extending functionality

Basic operations with Day.js

Creating date objects – dayjs() constructor

```
let now = dayjs()  // today
let date1 = dayjs('2019-12-27T16:00');
        // from ISO 8601 format
let date2 = dayjs('20191227');
        // from 8-digit format
let date3 = dayjs(new Date(2019, 11, 27));
        // from JS Date object
let date5 = dayjs.unix(1530471537);
    // from Unix timestamp

By default, Day.js parses in local time
```

https://day.js.org/docs/en/parse/parse

Displaying date objects – format()

```
console.log(now.format());
        2021-03-02T16:38:38+01:00

console.log(now.format('YYYY-MM [on the] DD'));
        2021-03 on the 02

console.log(now.toString());
        Tue, 02 Mar 2021 15:43:46 GMT

By default, Day.js displays in local time
```

Get/Set date/time components

```
# obj.unit() -> get
# obj.unit(new_val) -> set

let now2 = now.date(15);
let now2 = now.set('date', 15);
        2021-03-15T16:50:26+01:00

let now3 = now.minute(45);
let now3 = now.set('minute',45);
        2021-03-02T16:45:26+01:00

let today_day = now.day();
let today_day = now.get('day');
        2
```

Unit	Shorthand	Description
`date`	`D`	Date of Month
`day`	`d`	Day of Week (Sunday as 0, Saturday as 6)
`month`	`M`	Month (January as 0, December as 11)
`year`	`y`	Year
`hour`	`h`	Hour
`minute`	`m`	Minute
`second`	`s`	Second
`millisecond`	`ms`	Millisecond

https://day.js.org/docs/en/get-set/get-set

Date Manipulation and Comparison

```
let wow = dayjs('2019-01-25').add(1, 'day').subtract(1, 'year').year(2009).toString() ;
// "Sun, 25 Jan 2009 23:00:00 GMT"
```

- Methods to "modify" a date (and return a modified one)
- `.add` / `.subtract`
- `.startOf` / `.endOf`
- `d1.diff(d2, 'unit')`
- Specify the unit to be added/subtracted/rounded
- Can be easily *chained*

- Day.js objects can be compared
- `.isBefore` / `.isSame` / `.isAfter`
- `.isBetween`
- `.isLeapYear` / `.daysInMonth`

Day.js Plugins

- To keep install size minimal, several functions are only available in *plugins*
- Plugins must be
  - Loaded
  - Registered into the libraries
  - (in this case, they come with dayjs package, no need to install them)
- Then, functions may be freely used

```
const isLeapYear =
    require('dayjs/plugin/isLeapYear') ;
        // load plugin

dayjs.extend(isLeapYear) ;
        // register plugin

console.log(now.isLeapYear()) ;
        // use function
```

Advanced Day.js Topics

- Localization / Internationalization
  - Language-aware and locale-aware parsing and formatting
  - Various formatting patterns for different locales/languages
- Durations
  - Measuring time intervals (the difference between two time instants)
  - Interval arithmetic
- Time Zones
  - Conversion between time zones

License

- These slides are distributed under a Creative Commons license “**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**”
- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- https://creativecommons.org/licenses/by-nc-sa/4.0/