VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY | KATEDRA INFORMATIKY

Basics of Information Technology

Ing. Michal Radecký, Ph.D. MBA

Front-end frameworky, etc.

Frond-end frameworks

They offer tools for effective implementation of web design

It is based on CSS frameworks (e.g. Blueprint – grid, typography, form elements)

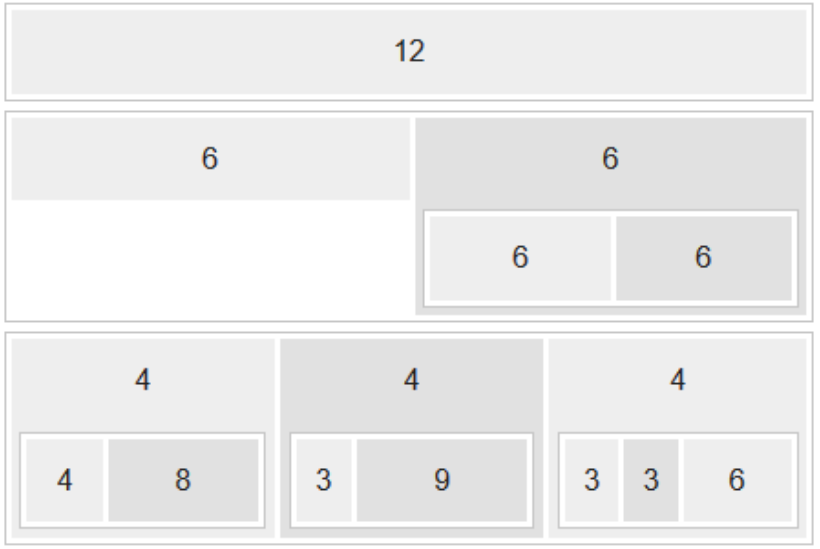
They mainly bring …

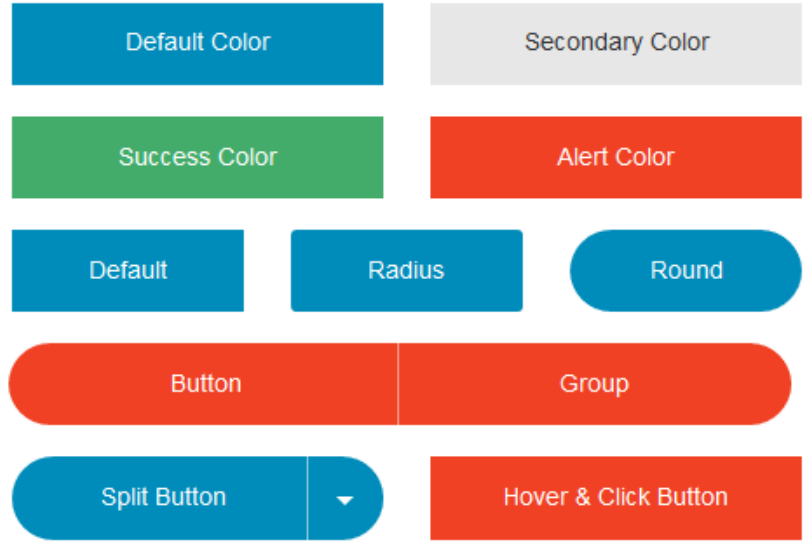
- Supporting responsibility and cross-platform
- Creating layouts
- Working with typography
- User Controls
- Extension libraries

Bootstrap, Tailwind, Foundation, Skeleton, etc.

Foundation

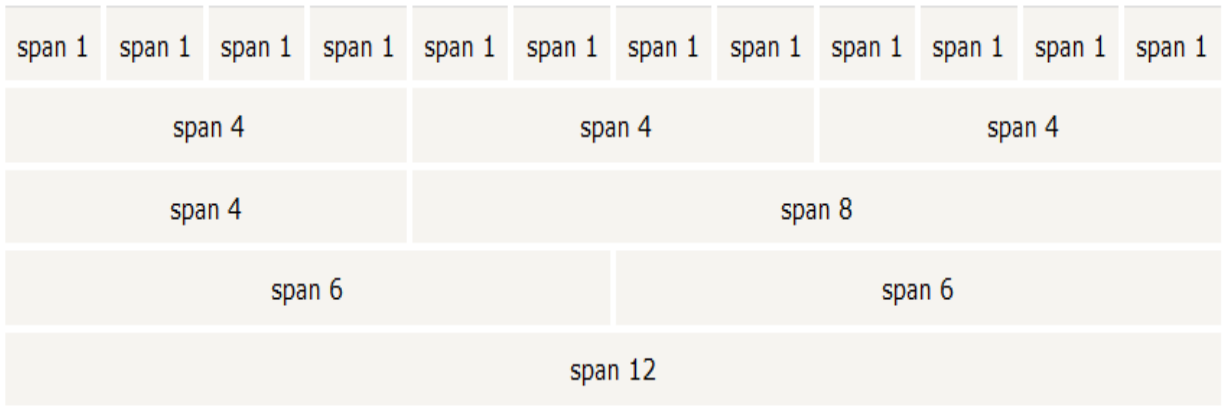
- http://foundation.zurb.com/
- Version 6, MIT licence, approx. 150 kB
- Preprocesor SASS, Vanilla JS (ES5)
- Modular architecture (custom build) vč. JS components
- Comprehensive and flexible





Bootstrap

- http://getbootstrap.com/
- Version 5, MIT licence, approx. 250 kB
- Preprocesor SASS from 4, Vanilla JS (ES6)
- Semi-modular (code-level)
- Large community, many components, templates, etc.
- Easy and flexible integration



Tailwind

- http://tailwindcss.com/
- Version 4, MIT licence, approx. 300 kB (dynamic)
- It does not work with components, but with "utilities" – a set of classes replacing direct CSS creation
- High flexibility and popularity
- Tailwind Plus (UI) – modular and component extension inc. JS

```html
<div class="flex flex-col items-center gap-6 p-7 rounded-2xl">
  <div>
    <img class="size-48 shadow-xl rounded-md" alt="" src="/img/cover.png" />
  </div>
  <div class="flex items-center">
    <span class="text-2xl font-medium">Class Warfare</span>
    <span class="font-medium text-sky-500">The Anti-Patterns</span>
    <span class="flex gap-2 font-medium text-gray-600 dark:text-gray-400">
      <span>No. 4</span>
      <span>·</span>
      <span>2025</span>
    </span>
  </div>
</div>
```

Why Use Styling Frameworks

Efficiency and scope of application for a particular project should always be considered

Advantages

- Rapid development within a "standardized" environment
- Native CSS preprocessor support
- Implementation of most commonly used elements (also with regard to cross-platform)
- Small entry barrier with spectacular results
- Optimization thanks to composite construction
- Support for customization and themes
- Support in development tools (extensions in VSCode) and web frameworks
- Wide deployment licensing rules (MIT)

Disadvantages

- From a certain point of view, too complex
- Link to a number of other tools and libraries with the need to interconnect, manage, etc. (node.js, Grunt, Rails, Compass, etc.)

Bootstrap - grid

https://www.w3schools.com/bootstrap5/index.php

- The basis of the page layout **is the grid system** (12 cells)
- Container (Responsive/Fluid) - Row – Cell/Columns
- Division by resolution – otherwise under each other, possibility of combination

`.col-` (extra small devices - screen width less than 576px)
`.col-sm-` (small devices - screen width equal to or greater than 576px)
`.col-md-` (medium devices - screen width equal to or greater than 768px)
`.col-lg-` (large devices - screen width equal to or greater than 992px)
`.col-xl-` (xlarge devices - screen width equal to or greater than 1200px)
`.col-xxl-` (xxlarge devices - screen width equal to or greater than 1400px)

Bootstrap – Styling the appearance

Native styling – direct styling of some elements (h1-6, attr, blockquote, code, ...)

Component styling – use existing components (.btn) first

Styling utilities – primarily use utilities (.text-center) for customization

Setting variables – setting CSS variables to customize the final appearance (--bs-link-color)

User styling - customize existing styling (custom class)

Never interfere with the original sources of Boostrap, but create your own in "rewrite" mode!

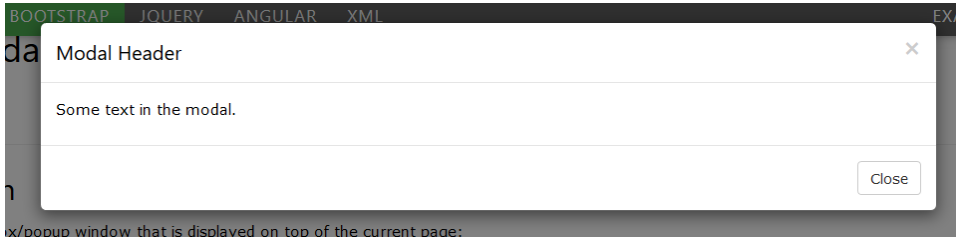
- The basic principle is to "not style directly", but everything using predefined styles. Basically, (for the main level) there is no need to create your own styling at all.
- The HTML structure (modal dialog) must also correspond to this
- For more complex interaction, an API built on JS is used
- Thanks to the grid, simple implementation of the "mobile-first" approach

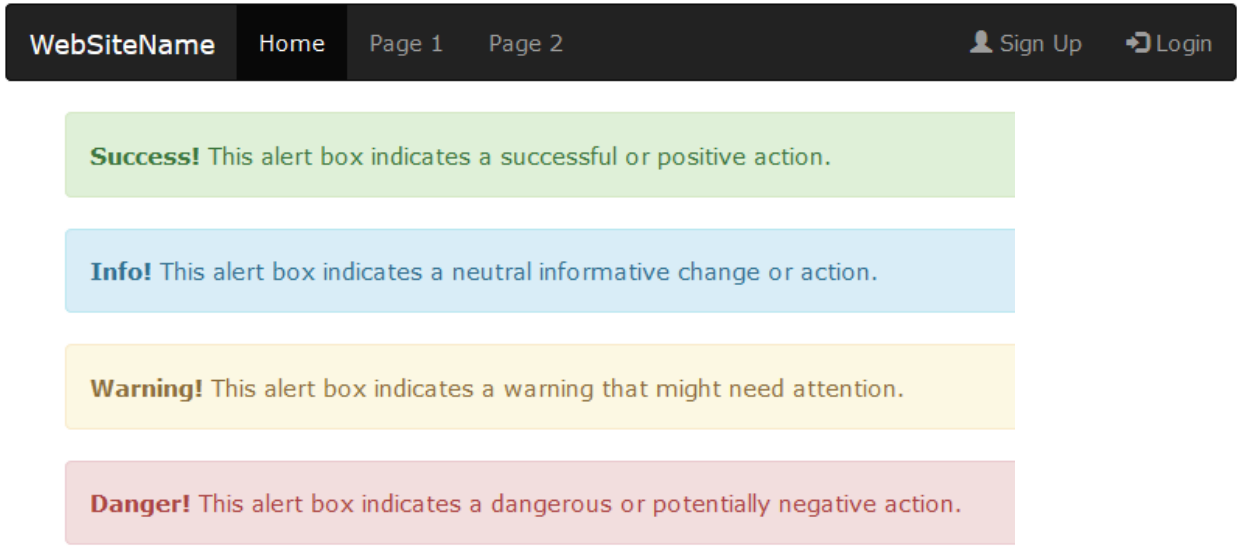
Bootstrap – Components

Selection of consistent visual elements (including functionality) only thanks to the combination of HTML and the set CLASS, příp. JS.

- **Structural elements** - cavbar, header, footer, grid, container, layout utilities
- **Content elements** - cards, list groups, tables, alerts, badges, text utilities
- **Form elements** - inputs, selects, toggles, sliders, floating labels, validace, stavy
- **Interactive Elements (JS)** – modal, tooltip, accordion, carousel

No need to write JS, just data attributes (data-bs-*)





Bootstrap – interaction

It uses a DOM-driven approach – everything is defined in HTML, using attribute data. These are searched for and activated by the corresponding JS when Bootstrap is loaded. It is still possible to add your own JS implementation (events, etc.) incl. Bootstrap API.

Data attributes allow

- activate components
- change status
- manage interactions

At the same time, Bootstrap uses **ARIA attributes** (aria-expanded, aria-controls, role=)
to ensure full accessibility and proper behaviour for readers and assistive technologies.

```
<button type="button" class="btn btn-primary"
        data-bs-toggle="modal"
        data-bs-target="#exampleModal">
  Open modal
</button>
```

```
<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">

      <div class="modal-header">
        <h5 class="modal-title">Modal title</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></butt
      </div>

      <div class="modal-body">
        Short modal text.
      </div>

      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
      </div>

    </div>
  </div>
</div>
```

Bootstrap – Plugins & Themes

In addition to native components, it is also possible to use third-party extensions or your own

- Material Design - https://mdbootstrap.com/
- Pixel UI Kit - https://github.com/themesberg/pixel-bootstrap-ui-kit
- Telerik - https://www.telerik.com/design-system/docs/themes/kendo-themes/bootstrap

Boostrap versions are not compatible, so it is always necessary to take versions into account!

In addition to component packages, it is also possible to implement complex templates that are focused on specific visual modifications and specific functionality

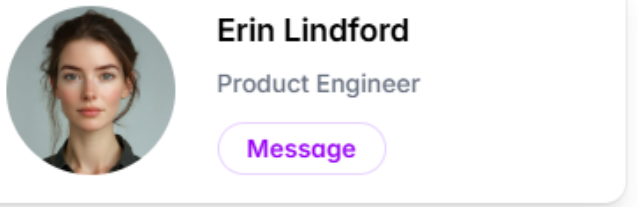
- AdminKit - https://adminkit.io
- AyroUI - https://ayroui.com
- CoreUI - https://coreui.io/product/free-bootstrap-admin-template

Expansion packages can be free (MIT, General Public License) or commercial.

Tailwind – Basic concept

Does not include components - **Utility-first CSS framework** - styling using small, reusable classes, where each class has one responsibility (eg. p-4, text-center, bg-blue-500)

- **Composition** - combining utilities creates more complex styling and appearance
- **Mobile-first** - classes can be targeted by breakpoints (sm:, md:, lg:…)
- **Status variants** - hover, focus, active, disabled (hover:bg-blue-700)
- **Dark mode** - dark: variants
- **JIT engine** - generates only utilities used – minimal resulting CSS



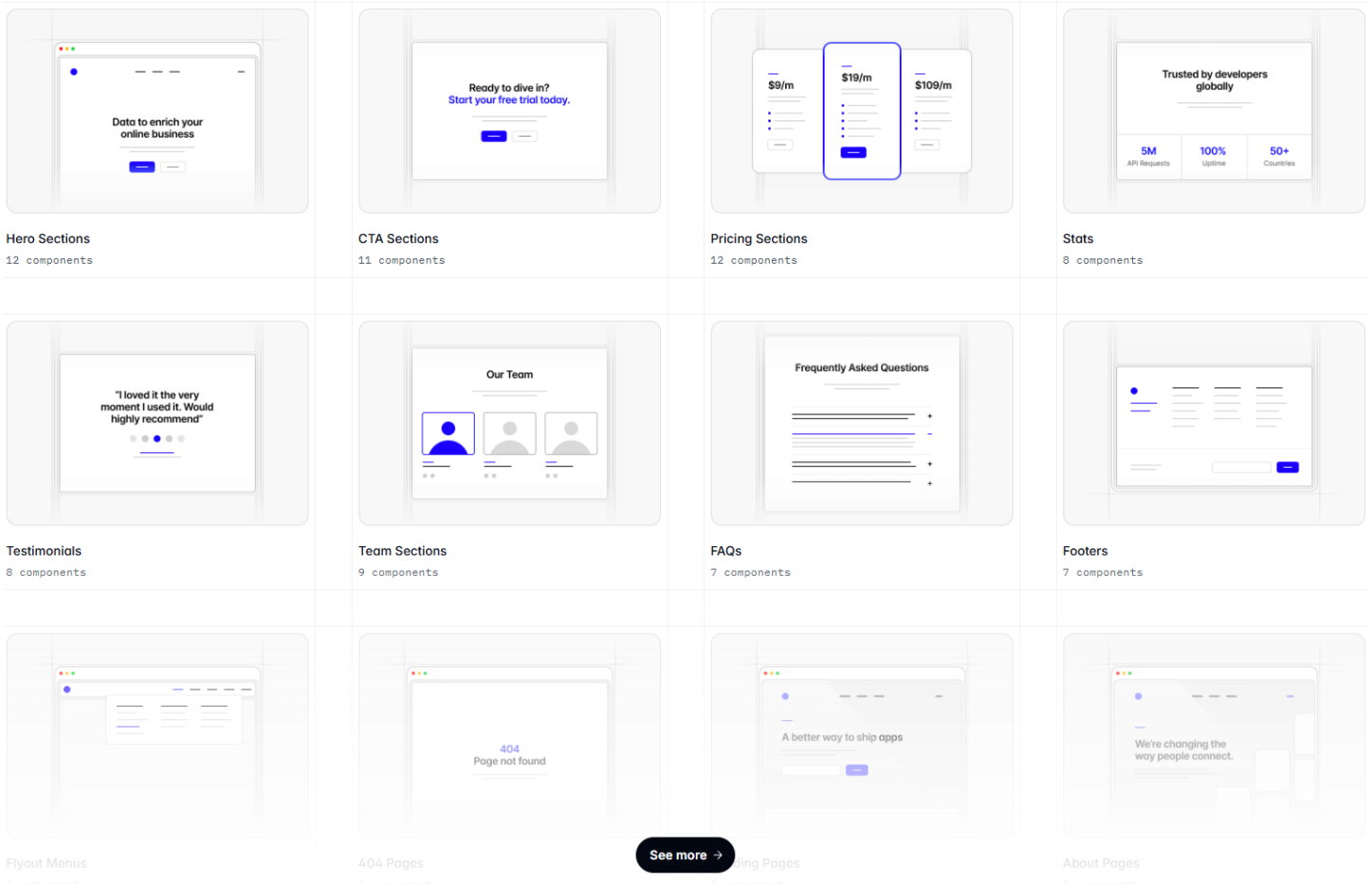

```
<div class="flex flex-col gap-2 p-8 sm:flex-row sm:items-center sm:gap-6 sm:py-4 ...">
  <img class="mx-auto block h-24 rounded-full sm:mx-0 sm:shrink-0" src="/img/erin-lindford.jpg" alt="" />
  <div class="space-y-2 text-center sm:text-left">
    <div class="space-y-0.5">
      <p class="text-lg font-semibold text-black">Erin Lindford</p>
      <p class="font-medium text-gray-500">Product Engineer</p>
    </div>
    <button class="border-purple-200 text-purple-600 hover:border-transparent hover:bg-purple-600 hover:t
      Message
    </button>
  </div>
</div>
```

Tailwind – Components

Tailwind is CSS only. You can extend components or interactive elements by using libraries.

- Tailwind Plus - https://tailwindcss.com/plus
- DaisyUI - https://daisyui.com
- Flowbite - https://flowbite.com

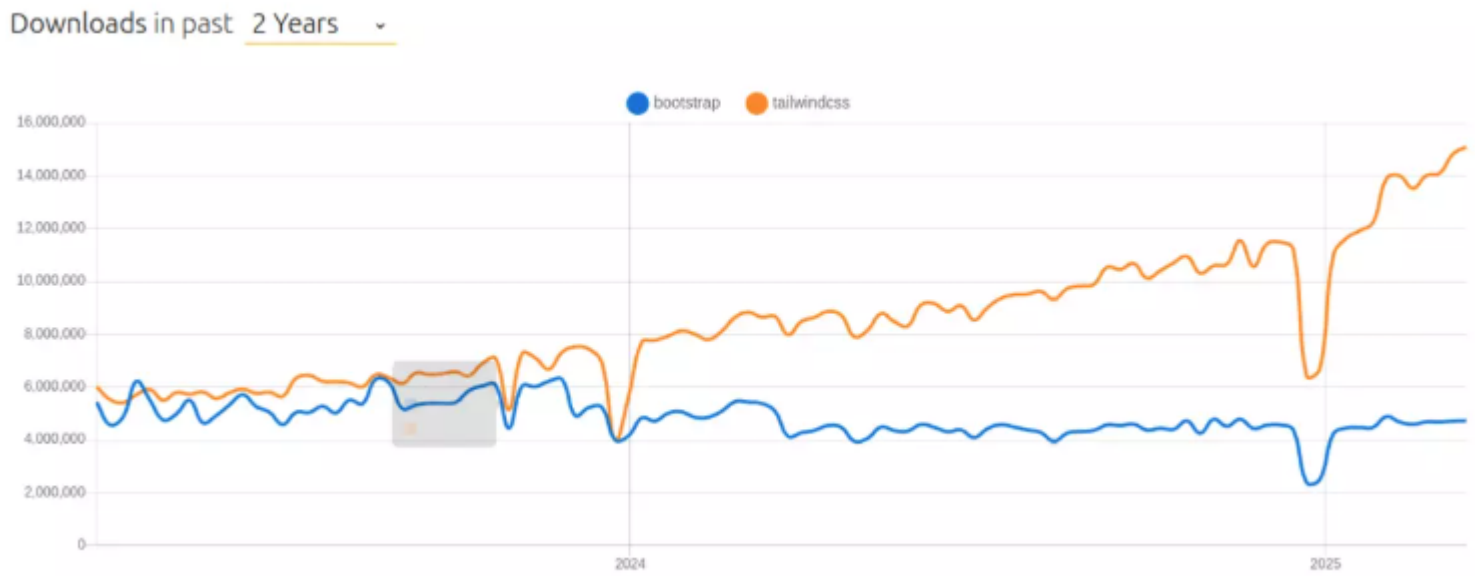
It is possible to customize
the appearance,
by setting CSS variables
or work with templates (@theme)



https://metadrop.net/en/articles/tailwind-css-vs-bootstrap

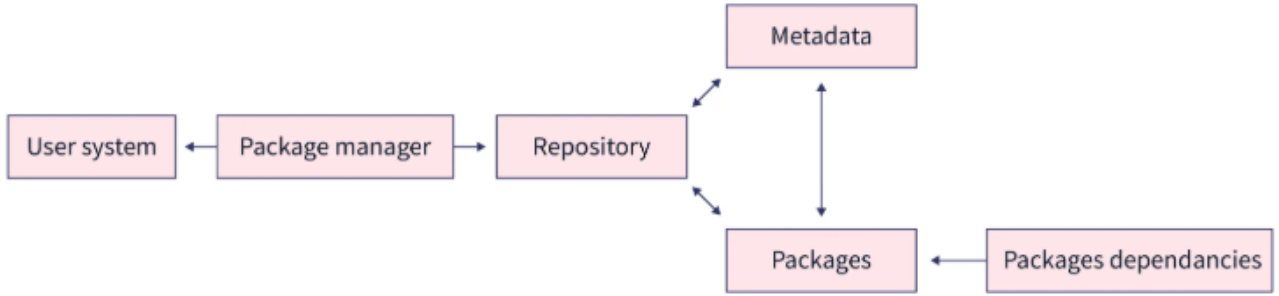
Summary

- Frontend CSS frameworks have their justification and can significantly increase the efficiency and final effect of web applications.
- Many things can be solved without knowledge of CSS or JS, but for complex applications, the link to other web technologies is a necessity.
- Using frameworks at a deeper level means using the CSS preprocessor and other integration tools.
- The key is the choice of a specific framework and related options.



Packaging tools

- Tools for installing, managing and updating libraries, frameworks and dependencies. Modern projects have hundreds of dependencies – manual management is practically impossible.
- They allow you to connect the project with external packages: CSS frameworks, JS libraries, utilities
- They ensure consistency of versions according to **package.json** - elimination of manual downloading of ZIP archives, management is done automatically
- Automate build processes (SCSS, TypeScript, minification, etc.)
  – CI/CD (Continuous Integration/Continuous Delivery/Deployment)



SCALER Topics

Packaging tools – How they work

Each package is placed in the registry (private/public, adding to the registry requires a publishing and approval process)

- **NPM registry – largest, web frameworks, tools (npm, yarn, pnpm - registry.npmjs.org)**
- PyPI – central registry for Python packages(pip)
- RubyGems – packages for Ruby(gems)
- NuGet registry - .NET platform (nuget)
- GitHub, GitLab, etc.

Package integration

1. The package is searched for by name (version) in the registry (npm install bootstrap)
2. The package metadata (versions, dependencies, configurations, ...) is obtained and written to the lock file (package-lock.json)
3. Download the package source files (node_modules)
4. Dependencies are resolved – the package needs additional packages, dependecy tree/graph – lock files
5. Integrity and signatures are checked for changes compared to the register
6. It is also possible to run "post-installation" scripts, e.g. for compilation or build

Packaging tools – package.json

- This is the project manifest (npm/yarn/pnpm).
- It defines information about the project, dependencies, scri
- Complex management and evaluation issues
  versions – lock file maintains state
- It may also contain
  - Configure the project environment
  - Configuring specific packages
  - Data for package publication
- Support for script definition and execution (automation)
  npm run release
  - lint, test, build, deploy

```
{
  "scripts": {
    "dev": "vite",
    "lint": "eslint src/",
    "test": "jest",

    "prebuild": "npm run lint && npm run test",
    "build": "vite build",

    "release": "npm run build && npm run deploy",
    "deploy": "node ./scripts/deploy.js"
  }
}
```

```
{
  "name": "version-demo",
  "version": "1.0.0",
  "private": true,

  "scripts": {
    "dev": "vite",
    "build": "vite build"
  },

  "dependencies": {
    "react": "^18.2.0",          // 1) caret: povoluje minor + patch
    "react-dom": "~18.2.0",      // 2) tilde: povoluje POUZE patch
    "axios": "1.6.2",            // 3) přesná verze, žádné aktualizace
    "dayjs": ">=1.11.0 <2.0.0",  // 4) rozsah verzí (range)
    "uuid": "*"                  // 5) libovolná verze (nedoporučuje se)
  },

  "devDependencies": {
    "vite": "^5.0.0",            // nejběžnější zadání (caret)
    "eslint": "~9.0.0",          // patch aktualizace
    "jest": "29.x",              // 6) povoluje libovolný minor/patch v major verzi
    "tailwindcss": "latest"      // 7) speciální npm tag (instaluje poslední verzi)
  }
}
```

https://metadrop.net/en/articles/tailwind-css-vs-bootstrap

Packaging tools

NPM, Yarn, Ppnpm

- Very similar principles
- Different principles of working with cache (node_modules)
- Speed differences, API
- They enable both local and global installation

Features	NPM	Yarn	PNPM
Package Installation	'npm install package-name'	'yarn add package-name'	'pnpm add package-name'
Dependency Tracking	package.json	yarn.lock or package.json	'package.json and pnpm-lock.yaml'
Scripts	'npm run script-name'	'yarn run script-name'	**'pnpm run script-name'**
Global Packages	Supported	Supported	Supported
Private Packages	Supported	Supported	Supported
Version Control	package-lock.json	yarn.lock	**pnpm-lock.yaml**
Security	Basic HTTPS download validation	Checksum validation	Checksum validation
Performance	Slower than Yarn (sequential downloads)	Faster than NPM (parallel downloads)	Faster than NPM (parallel downloads)
Offline Support	Yes	Yes	Yes

Packaging tools

Advantages

- Automatic updates
- Reduce the risk of version conflicts and dependency control
- Faster development thanks to scripts (npm run build)
- Links to other advanced development mechanisms
  - Build tools– webpack, vite, esbuild, etc.
  - Transpilers – TypeScript, SASS, etc.
  - Frameworks – Angular, React, etc.
- Security mechanisms for addictions
- Easy project sharing (no package content) and project reproducibility in different environments

Disadvantages

- Possible dependence on the tools used and their internal functioning
- More complex ecosystem and project management
- Great dynamics of package versions and their problematic tracking/control
- More complex configuration in relation to downstream tools and environments