BaxEnergy

Software solutions for energy and beyond

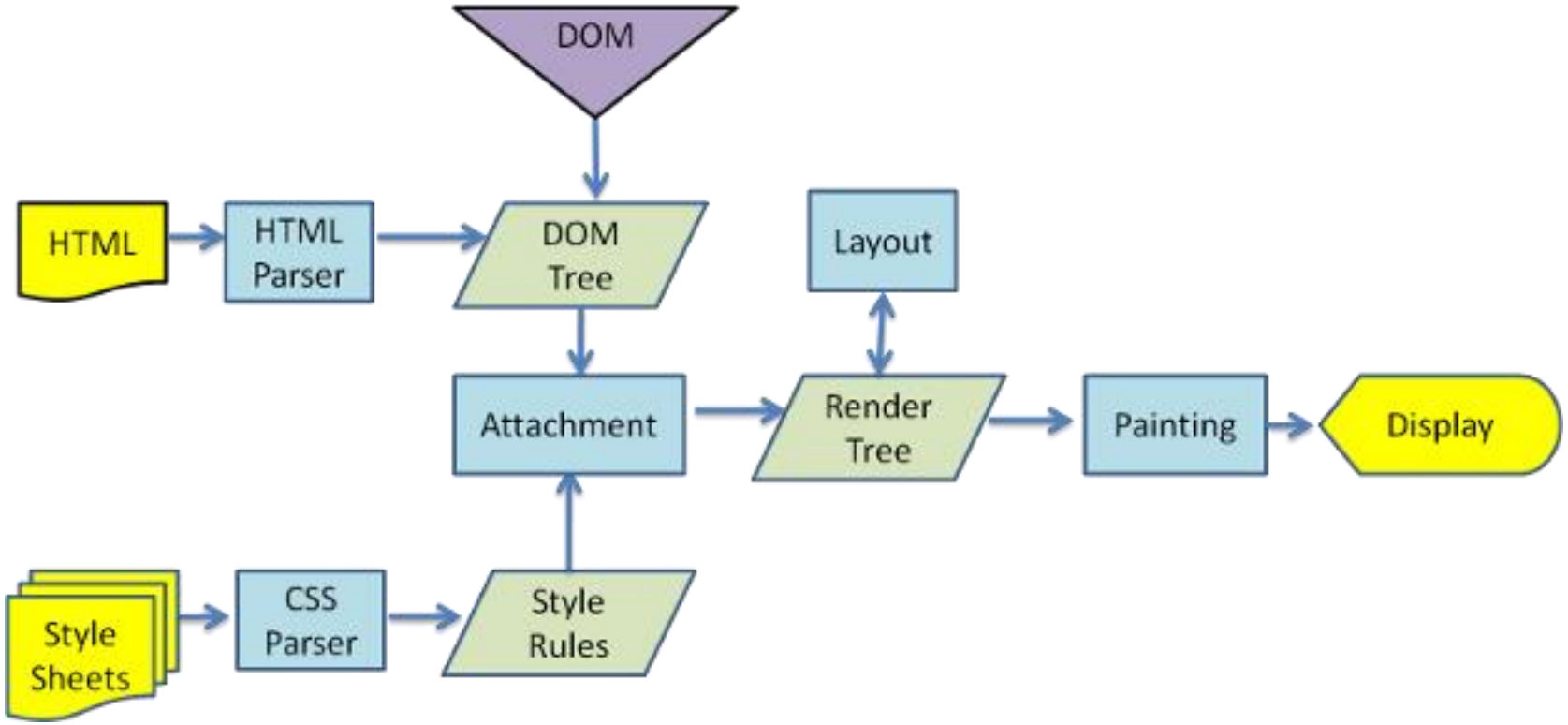
Eng. Vincenzo Di Martino

Course Overview…
Let's talk!

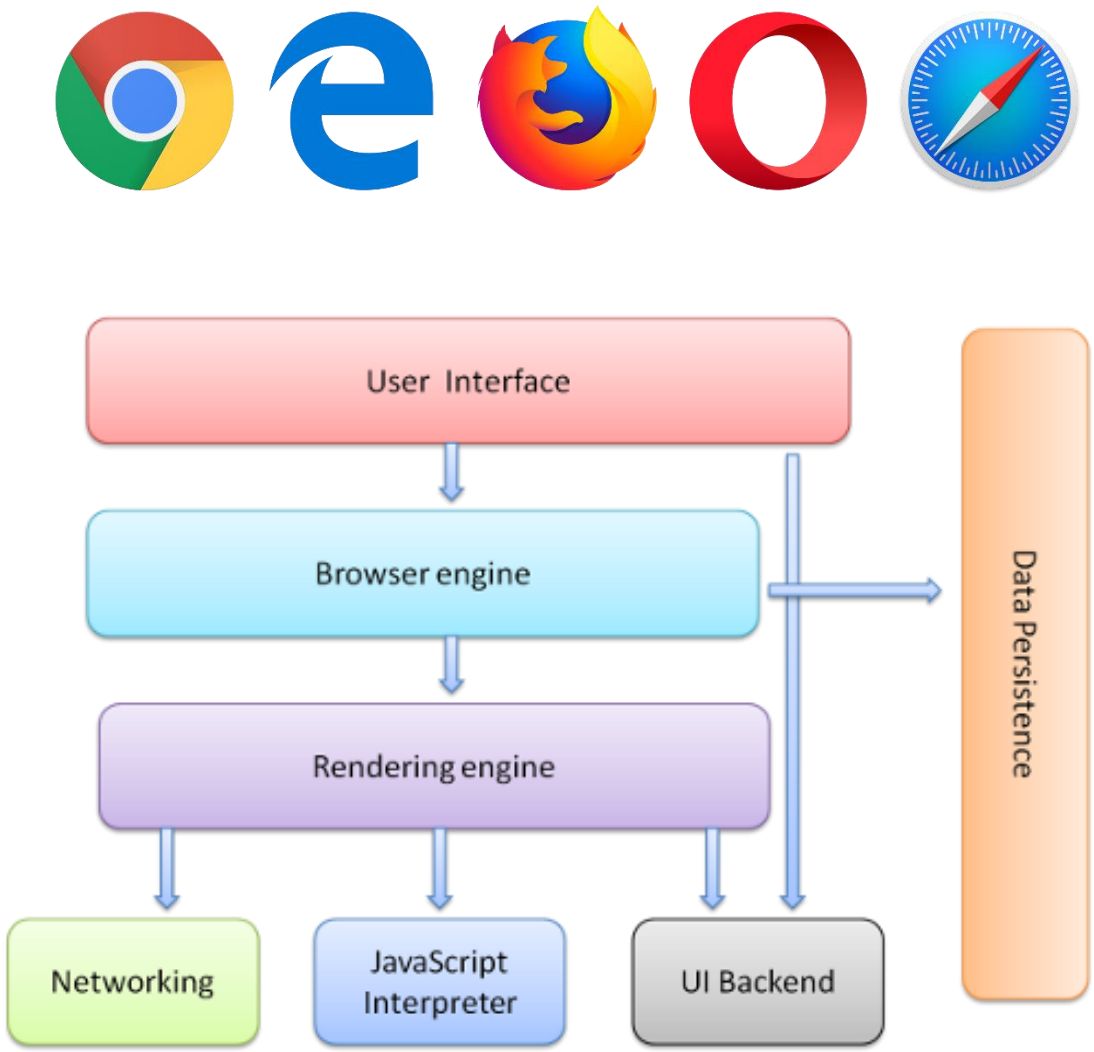
A bit of history

Web design Programming and Usability

Web design Programming and Usability



Web design Programming and Usability



Why?

(ovvero che ce frega a noi?)

Web design Programming and Usability

Web design Programming and Usability

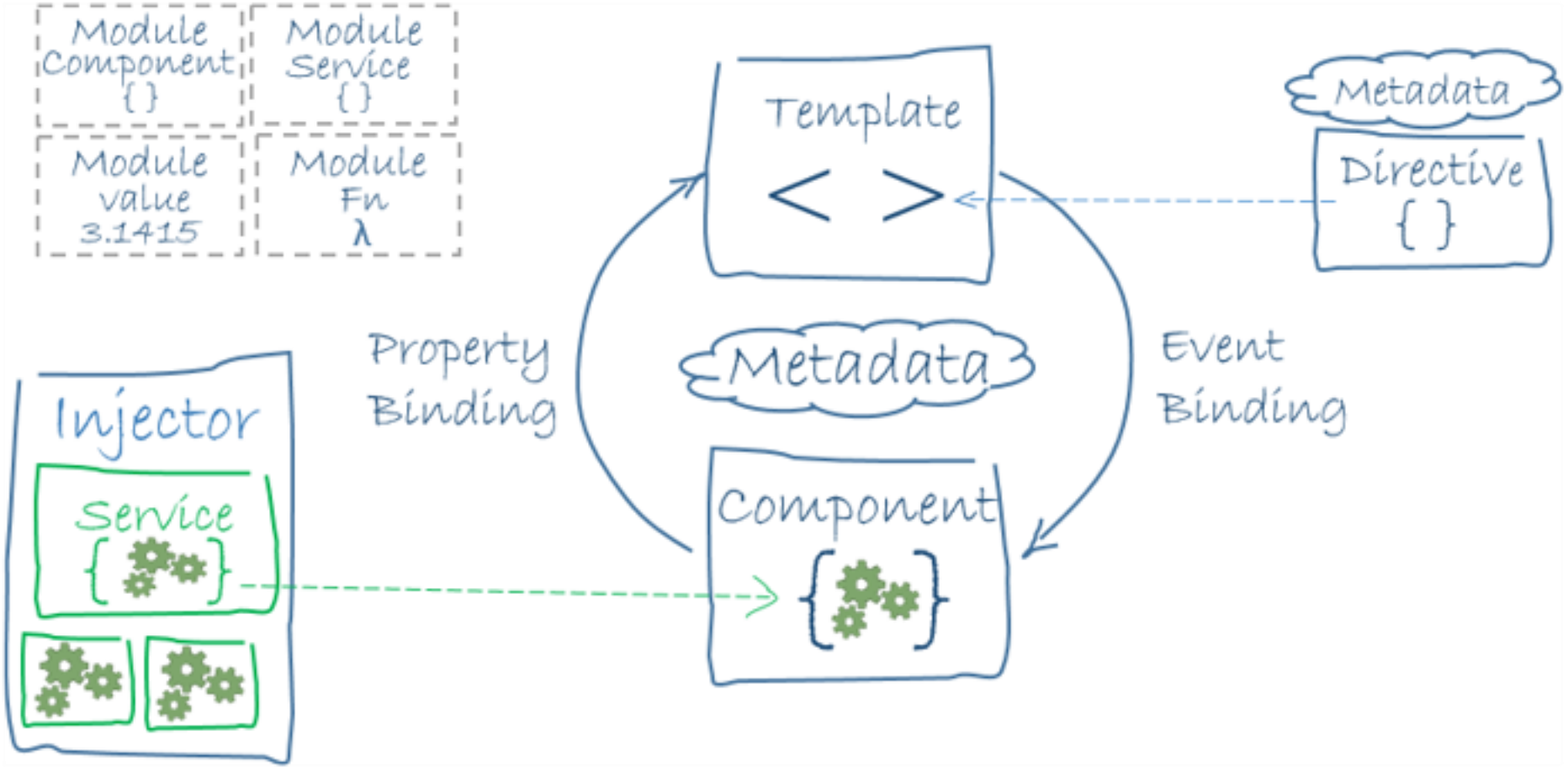
Angular CLI

```
Available Commands:
  add Adds support for an external library to your project.
  build (b) Compiles an Angular app into an output directory named dist/ at the given output path. Must be executed from within a workspace directory.
  config Retrieves or sets Angular configuration values in the angular.json file for the workspace.
  doc (d) Opens the official Angular documentation (angular.io) in a browser, and searches for a given keyword.
  e2e (e) Builds and serves an Angular app, then runs end-to-end tests using Protractor.
  generate (g) Generates and/or modifies files based on a schematic.
  help Lists available commands and their short descriptions.
  lint (l) Runs linting tools on Angular app code in a given project folder.
  new (n) Creates a new workspace and an initial Angular app.
  run Runs an Architect target with an optional custom builder configuration defined in your project.
  serve (s) Builds and serves your app, rebuilding on file changes.
  test (t) Runs unit tests in a project.
  update Updates your application and its dependencies. See https://update.angular.io/
  version (v) Outputs Angular CLI version.
  xi18n Extracts i18n messages from source code.
```

Web design Programming and Usability

```
npm install @angular/cli
ng new <project_name>
```

Web design Programming and Usability



Angular Modules

Web design Programming and Usability

NgModules

- Angular apps are modular with a modularity system called NgModule
- NgModules are containers for a cohesive block of code dedicated to an application domain
- Modules are a great way to **organize** an application and **extend** it with capabilities from external libraries
- They can contain components, service providers, and other code files whose scope is defined by the containing NgModule
- They can import functionality that is exported from other NgModules, and export selected functionality for use by other NgModules
- Every Angular app has **at least one** NgModule class, the **root** module, which is conventionally named AppModule and resides in a file named app.module.ts
- You launch your app by bootstrapping the root NgModule

Web design Programming and Usability

NgModules

An NgModule is defined by a class decorated with **@NgModule()**.

The **@NgModule() decorator is a function** that takes a **single metadata object**, whose properties describe the module.

In details @NgModule metadata does the following:

- **Declares** which components, directives, and pipes belong to the module
- Makes some of those components, directives, and pipes **public** so that **other module's component templates can use them**
- **Imports** other modules with the components, directives, and pipes that components in the current module need
- **Provides services** that the other application components can use

Web design Programming and Usability

NgModules

The metadata object input of NgModule function has the following properties:

Declarations

- The module's declarations array tells Angular which **components belong to** that module. As you create more components, add them to declarations.

- You must **declare every component in exactly one NgModule class**. If you use a component without declaring it, Angular returns an **error** message.

- The declarations array only takes declarables. Declarables are components, directives and pipes. Declarables must **belong to exactly one module**. The compiler emits an **error** if you try to declare the same class in more than one module.

- These declared classes are visible within the module but invisible to components in a different module unless they are **exported** from this module and the other module imports this one.

Web design Programming and Usability

NgModules

Imports

Other modules whose exported classes are needed by component templates declared in this NgModule.

The module's imports array appears exclusively in the @NgModule metadata object.

It tells Angular about **other NgModules** that this particular module **needs** to function properly.

Exports

The subset of declarations that should be visible and usable in the component templates of other NgModules.

Only component and directives that are **exported** from one module can be **used** in another module.

Providers

The providers array is where you **list the services the app needs**. When you list services here, they are **available app-wide**.

Web design Programming and Usability

NgModules

Bootstrap

The application launches by bootstrapping the root AppModule, which is also referred to as an entryComponent. Among other things, the bootstrapping process creates the component(s) listed in the bootstrap array and inserts each one into the browser DOM.

Each bootstrapped component is the base of its own tree of components.

Inserting a bootstrapped component usually triggers a cascade of component creations that fill out that tree.

Angular Components

Web design Programming and Usability

NgComponent

- A *component* controls a patch of screen called ***view***
- You define a component's application logic inside a class.
- Angular engine is in charge of synchronizing view and data back and forth
- Angular creates, updates, and destroys components through lifecycle hook methods

`ngOnChanges()`	Respond when Angular (re)sets data-bound input properties. The method receives a `SimpleChanges` object of current and previous property values. Called before `ngOnInit()` and whenever one or more data-bound input properties change.
`ngOnInit()`	Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties. Called *once*, after the *first* `ngOnChanges()`.
`ngDoCheck()`	Detect and act upon changes that Angular can't or won't detect on its own. Called during every change detection run, immediately after `ngOnChanges()` and `ngOnInit()`.
`ngAfterContentInit()`	Respond after Angular projects external content into the component's view / the view that a directive is in. Called *once* after the first `ngDoCheck()`.
`ngAfterContentChecked()`	Respond after Angular checks the content projected into the directive/component. Called after the `ngAfterContentInit()` and every subsequent `ngDoCheck()`.
`ngAfterViewInit()`	Respond after Angular initializes the component's views and child views / the view that a directive is in. Called *once* after the first `ngAfterContentChecked()`.
`ngAfterViewChecked()`	Respond after Angular checks the component's views and child views / the view that a directive is in. Called after the `ngAfterViewInit()` and every subsequent `ngAfterContentChecked()`.
`ngOnDestroy()`	Cleanup just before Angular destroys the directive/component. Unsubscribe Observables and detach event handlers to avoid memory leaks. Called *just before* Angular destroys the directive/component.

Web design Programming and Usability

NgComponent

- A *component* controls a patch of screen called ***view***
- You define a component's application logic inside a class.
- Angular engine is in charge of synchronizing view and data back and forth
- Angular creates, updates, and destroys components through lifecycle hook methods

```
export class HeroListComponent implements OnInit {
    heroes: Hero[];
    selectedHero: Hero;
    constructor(private service: HeroService) {
    }
    ngOnInit() {
        this.heroes = this.service.getHeroes();
    }
    ngOnDestroy () {
        /* . . .dispose stuff here */
    }
    selectHero(hero: Hero) {
        this.selectedHero = hero;
    }
}
```

Web design Programming and Usability

NgComponent

The **@Component** decorator identifies the class immediately below it as a component class, and specifies its metadata.

The metadata for a component tells Angular where to get the major building blocks that it needs to create and present the component and its view.

It associates a template with the component, either directly with inline code, or by reference.
Together, the component and its template describe a view.

```
@Component({
    selector: 'app-hero-list',
    templateUrl: './hero-list.component.html',
    styleUrls: ['./hero-list.component.css']
    providers: [ HeroService ]
})
export class HeroListComponent implements OnInit {
   /* . . . */
}
```

Web design Programming and Usability

NgComponent

Metadata:

- **selector**: A CSS selector that tells Angular to create and insert an instance of this component wherever it finds the corresponding tag in template HTML.
- **templateUrl**: The module-relative address of this component's HTML template. This template defines the component's host view.
- **styleUrls**: Array of css files that apply on component template
- **style**: A set of css rules written inline
- **providers**: An array of providers for services that the component requires.

```
@Component({
    selector: 'app-hero-list',
    templateUrl: './hero-list.component.html',
    styleUrls: ['./hero-list.component.css']
    providers: [ HeroService ]
})
export class HeroListComponent implements OnInit {
   /* . . . */
}
```

Web design Programming and Usability

NgComponent

A template is a form of **HTML** that tells Angular how to **render the component**.

Views are typically arranged **hierarchically**, allowing you to modify or show and hide entire UI sections or pages as a unit.

The template immediately associated with a component defines that component's *host view*. The component can also define a *view hierarchy*, which contains *embedded views*, hosted by other components.

```
<h2>Hero List</h2>
<p><i>Pick a hero from the list</i></p>
<ul>
  <li *ngFor="let hero of heroes"
              (click)="selectHero(hero)">
    {{hero.name}}
  </li>
</ul>
<app-hero-detail *ngIf="selectedHero"
                 [hero]="selectedHero">
  <!– More declarations here -->
</app-hero-detail>
```

Web design Programming and Usability

Templates and views

A template looks like regular HTML, except that it also contains Angular template syntax, which alters the HTML based on your app's logic and the state of app and DOM data.

Your template can use *data binding* to coordinate the app and DOM data, *pipes* to transform data before it is displayed, and *directives* to apply app logic to what gets displayed.

The template-syntax elements tell Angular how to render the HTML to the screen, using program logic and data.

```
<h2>Hero List</h2>
<p><i>Pick a hero from the list</i></p>
<ul>
  <li *ngFor="let hero of heroes"
              (click)="selectHero(hero)">
    {{hero.name}}
  </li>
</ul>
<app-hero-detail *ngIf="selectedHero"
                 [hero]="selectedHero">
  <!– More declarations here -->
</app-hero-detail>
```

Thank you

Eng. Vincenzo Di Martino