**Bootstrapping:** Bootstraps the app using the root component from the specified NgModule

**Html Template Syntax:** Different syntax used within the HTML that allow for interactions such as binding, templating, variables and method calls.

**\*ngIf:** Removes or recreates a portion of the DOM tree

**\*ngFor:** Turns the html element into a template and instantiates a view for each element in a list

**\*ngSwitch:** Conditionally swaps the contents of the embedder templates basted on a condition

**[ngClass]:** Binds the presence of css classes on the element. {‘active’: isActive}

**[ngStyle]:**  Allows you to assign styles to an html element using css.

**@Modules:** Defines components, directives, pipes and providers

**@Component:** Declares a class is a component and allows you to define metadata about the component

**@Directive:** Declare that a class is a directive and provides metadata about the directive. Directives are used to extend the power of HTML attributes and allow you to reshape the DOM’s structure. Creating custom directives can allow for more flexibility then creating components.

1. Component Directives – Components are rendered as directives in html.
2. Structural Directive – Adds or removes DOM elements.(ngIf, [hidden] ect)
3. Attribute Directive – ngClass, ng Style – change styles/classes of html

**@Pipe:** Enables the transform data with the HTML. Pipes can be chained. Custom one’s must implement PipeTransform. Upper/Lower, date, Currency, Percent {{ 2 | customPipe: 10}}

**@Injectable:** Marks a class as available for Injections. Can be provided in modules or components. Usually referred to as a ‘service’. Can be used to share data across components.

**@Input():** Allows a parent component to communicate information to a child component. Not this info flow is one way. Changes in child are not reflected in parent. You can use a {setter} to intercept the input and act upon it.

**@Output():** A child component would use this to emit information and make that info available to a parent component.

**HttpInterceptor**: Gives the ability to intercept https calls and handle errors or add headers.

**ngOnChanges():** Used to observe all changes to properties on a component.

**Reactive Forms:** Model driven approach to handling form inputs. Built around observable streams where inputs and values are provided as streams accessed synchronously

**Template Forms:** No longer recommended. Uses [(ngModel)] binding for value controls.

**Async pipe:**

**Routing:** Requires a router-outlet. If hosting on IIS you need to configure redirect to index.html. Rout parameter are formatted like heros/:id and would be used like /heros/42. Order of routes matter. We can use route guards for authentication.

**Ng-template:**

**cli-budget:**

**Observable:** Data producer. Each subscription causes independent execution of the observable.

**Subject:** Data producer, data consumer. Can be subscribed to. Can subscribe to other observables. All subscribers to a subject share the data it produces.

**HttpUrlEncodingCodec:** Encodes the URL strings. I overwrote this for our dotnet core REST API once.

**HttpXsrfTokenExtractor:** Used to extract an xsrf token so it can be combined into the next request.