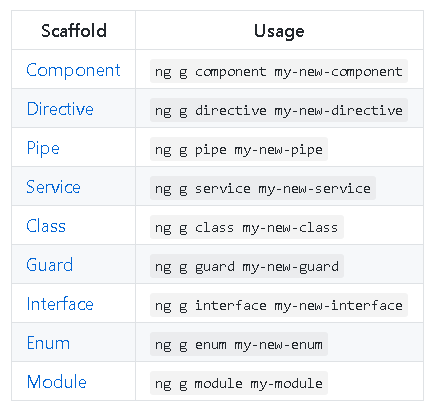
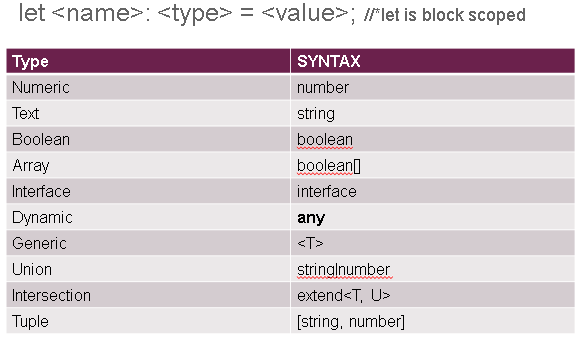
**Cheat Sheet**

**Angular**

|  |  |
| --- | --- |
| **NgModule** | |
| @NgModule({ declarations: ..., imports: ...,      exports: ..., providers: ..., bootstrap: ...}) class MyModule {} | Defines a module that contains components, directives, pipes, and providers |
| declarations: [MyRedComponent, MyBlueComponent] | List of components, directives, and pipes that belong to this module |
| imports: [BrowserModule, SomeOtherModule] | List of modules to import into this module. Everything from the imported modules available to declarations of this module. |
| providers: [MyService, { provide: ... }] | List of dependency injection providers visible both to the contents of this module and to importers of this module. |
| **Syntax** | |
| <p>Hello {{property}}</p> | Binds text content to an interpolated string, |
| <input [(ngModel)]="propety"/> | Sets up two-way data binding. Must  import { FormsModule } from '@angular/forms'; |
| <button (click)="readRainbow($event)"> | Calls method readRainbow when a click event is triggered on this button element (or its children) and passes in the event object. |
| <input [value]="expression"> | Binds property value to the result of expression |
| **Directives** | |
| <section \*ngIf="expression"> | Conditional based on expression |
| <li \*ngFor="let item of list"> | Loop through collection |
| **Decorators** | |
| @Component({...}) | Defines component |
| @Injectable() | Defines a class to be injectable |

**Ng Generate [Scaffold] [Name]**



**TypeScript**

Jasmine (hints)

|  |  |
| --- | --- |
| const nameField = fixture.debugElement.query(By.css("#name"));  let input = nameField.nativeElement;  input.value = 'new value';  input.dispatchEvent(new Event('input')); | Change value of input tag |
| const button = fixture.debugElement.query(By.css("#button"));  button.triggerEventHandler("click", null);  expect(component.CreateLunch).toHaveBeenCalled(); | Click event |