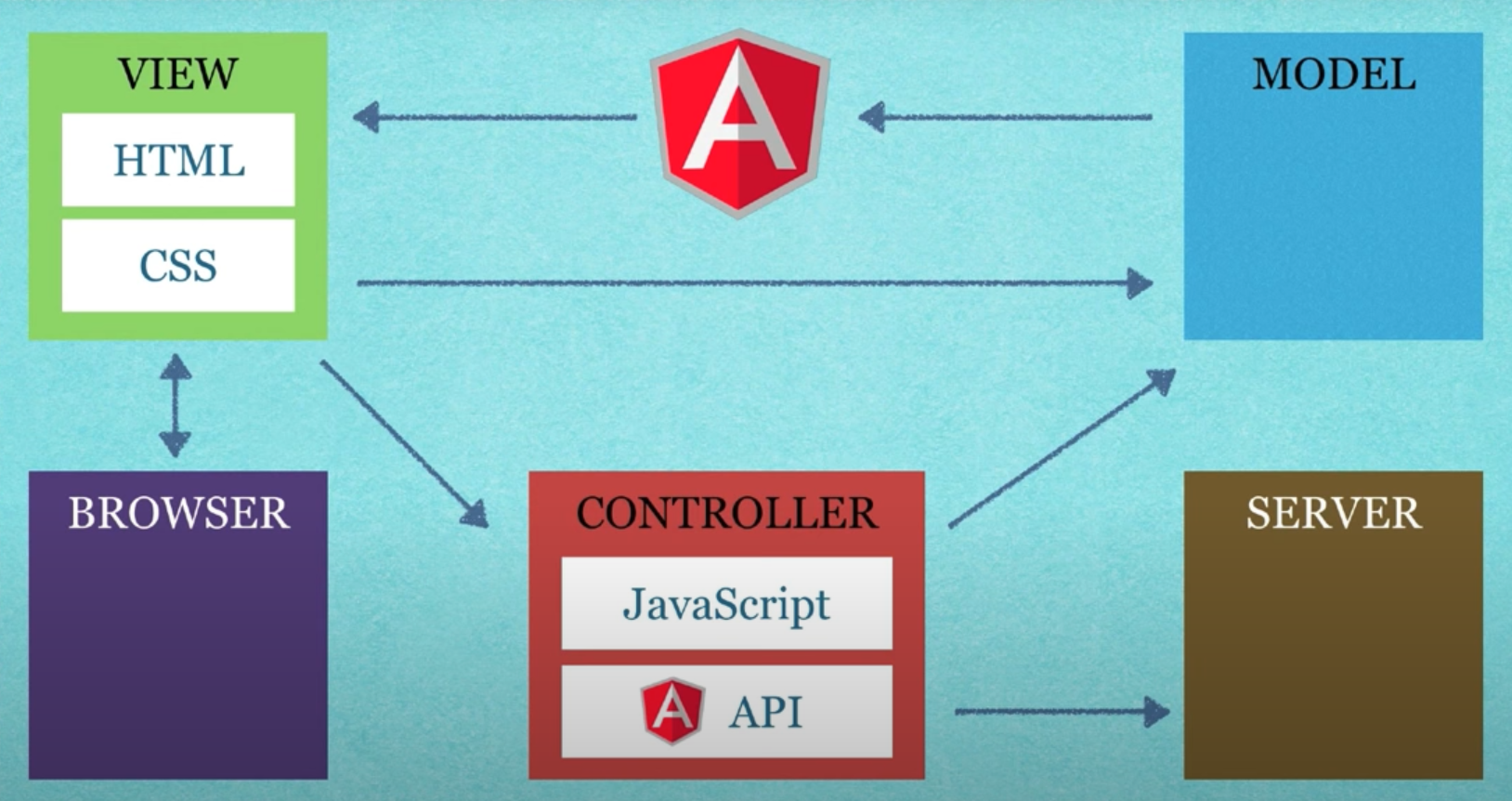
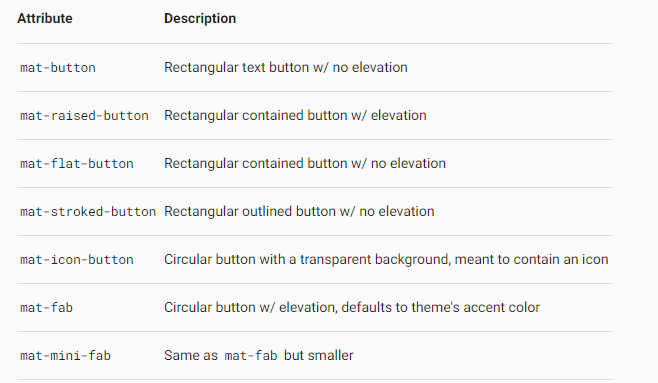
Angular training

* Angular is a development platform, built on TypeScript. Includes:
  + A component based framework.
  + A collection of libraries that cover a wide variety of features.
  + A suite of developer tools
* Components are the building blocks. They include:
  + TypeScript class with a @component() decorator, an HTML template, and styles.
* Add functionality through directives: ngIf and ngFor are the most popular.
* Angular CLI
  + Ng build- compiles an Angular app into an output directory
  + Ng serve – builds and serves the applications
  + Ng generate – generates or modifies files based on a schematic
  + Ng test – runs unit tests on a given project
  + Ng e2e – builds and serves an application, runs end to end tests.
* Directives bind behavior to HTML elements.

Angular Video

* 
* Classes:
* Decorators:
* Functions:
* Structures:
* Types:
* An NgModule always has a root component, but can include additional components.
* Angular components are organized into NgModules.
* An Angular app is defined by a set of NgModules
* Components define views, which are sets of screen elements that Angular can choose among and modify according to program logic and data.
* Components use services, which provide functionality not related to views. Services are injected into components as dependencies.
* Module, components and services are classes that use decorators. Decorators mark type and provide metadata that explains how to use them.
* A template combines HTML with angular directives and binding markup to prepare for display.
* Metadata is provided to components through dependency injection.
* Angular provided the Router service to help define navigation between components (views).
* Lazy loading – loading modules on demand, rather than at startup.
* Components connect with the Document Object Model (DOM). Each component defines a class that contains application data and logic, associated with an HTML template that defines a view to be displayed in the target environment.
* Event binding lets the app respond to user input.
* Property binding lets values that are computed from application data be used in the HTML.
* Angular supports two way object binding, which means changes in in the interface are reflected in the program data, and vice versa.
* A service class definition is preceded by the @Injectable() decorator.
* Decorators provide the metadata that allow other providers to be injected as dependencies in the class.
* 
* Modules contain:
  + Declarations: components, directives and pipes that belong to the NgModule
  + Exports: declarations that should be visible and usable in the component templates of other NgModules.
  + Imports: other modules who’s exported classes are needed by component templates declared in this NgModule
  + Providers: services that the module contributes to the global collection of services.s
* Components application logic is found inside a class.
* The class interacts with the view/template through an API of properties and methods.
* Actions are provided through lifecycle hooks, like ngOnInIt().
* Templates are mostly regular HTML, with some Angular template syntax.
* Pipes transform data before it is displayed.
* Directives apply app logic to what gets displayed.
* ngFor tells Angular to iterate over a list.
* Structural directives alter layout by adding, removing and replacing elements in the DOM.
* ngIf includes a component based on a condition.
* Attribute directives alter the appearance or behavior of an existing element.
* ngModel modifies an elements by setting display value property and response to change events.
* Service is a broad category, encompasses any value, function or feature that an app needs.
* A service is typically a class with a narrow, well-defined purpose.
* By putting services in a service class, it makes them more broadly available to all components.
* Services are injected into a class to allow access to them.
* Components shouldn’t fetch or save data directly, they should present and access data through a service.
* Angular binds data to properties, not attributes.
* To bind data and events across components, you have to use @Input and @Output decorators
* Decorators indicate to Angular what the class is
* Lifecycle Hooks
  + ngOnChanges: triggers following the modification of the @Input bound class.
  + ngOnInIt: triggers upon initialization of the component, and does not trigger again.
  + ngDoCheck: fires with every change detection cycle. Performing any action will cause it to trigger.
  + ngAfterContentInit: fires about the content DOM initializes.
  + ngAfterContentChecked: fires after change detection in the content DOM.
  + ngAfterViewInit: firers after the view DOM initializes
  + ngAfterViewChecked: fires after change detection in the view DOM.
  + ngOnDestroy: fires after the component is removed from the view and subsequent DOM.
* Using @Output() and @ Input() to pass information into and out of the component
* The [inputs] are used to pass inputs, (outputs) is used to handle outputs
* Two way data binding is completed using [(ngModel)]
* Tag options
  + <mat-autocomplete> autocompletes entries on a drop down.
    - Includes a <mat-option> to define the property.
  + <button mat-button> adds a clickable button
  + MatBottomSheet can be used to open panels on the bottom of the screen.
  + 
  + Button-toggle – accents a button when it is pushed.
  + <mat-card> is a container for text, photos and actions in the context of a single subject.
  + <mat-checkbox> creates a checkbox.
  + <mat-chip-list> displays a list of values as individual, keyboard accessible chips.
  + Datepicker allows a user to enter a date either through text input, or by choosing a date from the calendar.
  + MatDialog can be used to open modal dialogs.
  + <mat-divider> is used to create a horizontal or vertical divider between elements.
  + <mat-expansion-panel> creates an expandable details-summary view
  + <mat-form-field> is used to wrap several Angular Material components and apply common text field styles.
  + Mat-grid-list arranges cells into grid-based layout.
  + <mat-icon> uses vector based icons.
  + matInput uses <input> and <textarea> to work with <mat-form-field>
  + <mat-list> wraps and formats a series of line items. Contains <mat-list-item> elements.
    - Mat-nav-list is used for list items with anchor tags.
    - Mat-action-list used when an item in the list performs an action.
  + <mat-menu> is a floating panel containing list of options.
    - Uses matMenuTriggerFor directive to navigate.
  + <mat-paginator> provides navigation for paged information, typically used with a table.
  + <mat-progress-bar> is a horizontal progress bar for indicating progress.
  + <mat-progress-spinner> and <mat-spinner> are indicators of progress and activity.
  + <mat-radio-button> creates a radio button
  + matRipple attribute connects user input to screen reactions
  + <mat-select> is used to select a value from a set of options.
  + <mat-sidenav-content> represents main content, and <mat-sidenav> represents added side content.
  + <mat-slide-toggle> is used for an off/on control that can be toggled by clicking.
  + <mat-slider> is used to select the value from a range
  + matSnackBar is used to display snack-bar notifications
  + matSort and mat-sort-header are used to add sorting state and display to data.
  + Mat-horizontal-stepper and mat-vertical-stepper are used to divide content into logical steps.
  + Mat-table creates a table
  + Mat-tabs organize content into separate views where only one view is visible at a time.
  + <mat-toolbar> used for headers, titles or actions.
  + Mat-tooltip creates a text label that is displayed when the user hovers over or longpresses an element.
  + Mat-tree is used to display hierarchy data.
* Angular is a single page application. You don’t go to a different page, you just change the view

Typescript:

* Strong typing – defining variable types
* Object oriented features
* Compile time errors
* Great tooling
* Typescript transpiles into JavaScript
* Typescript will set a value for each variable.
* Code will compile correctly into JavaScript, but typescript will help catch runtime errors before they happen