**Trabalhando com Páginas SPA com Angular**

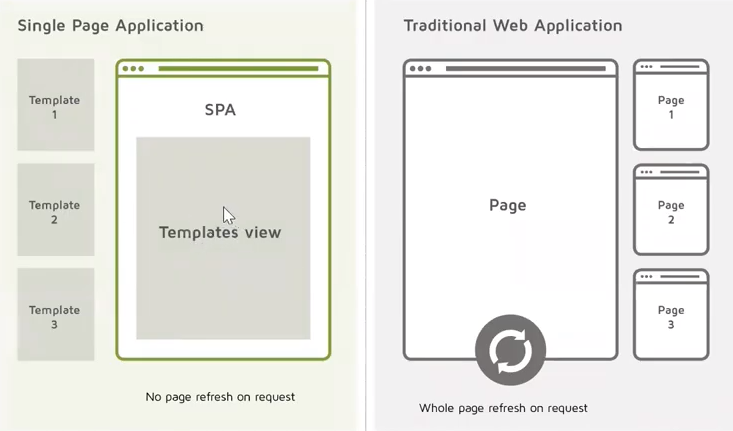
**sites:**

[www.material.angular.io/components/categories](http://www.material.angular.io/components/categories)

[www.angular.io/guide/testing](http://www.angular.io/guide/testing)

**Angular:**

* Framework que possibilita construir aplicações Web baseadas em HTML 5, CSS e JavaScript
* Focado na criação de aplicativos de página única
* Permite organizar essas tecnologias e entregar uma aplicação que executa no Browser, capaz de consumir um ou vários serviços disponibilizados por um servidor
* mais aderente a padrões
* trabalha com a ideia de classes
* qualquer propriedade do DOM pode receber um valor dinamicamente sem a necessidade de criar diretivas



**SPA** – Single Page Applications são aplicações Web que consistem em soluções com uma única página, onde o objetivo é fornecer uma experiência do usuário similar à de um aplicativo Desktop.

Comandos:

* ng build – compilar todo o conteúdo nosso feito com angular num outro arquivo Javascript numa pasta dist
* ng new spaangular – prefix=spa
  + would you lide to add angular routing? Ao responder “Yes”, ele cria um arquivo de rota em nossa aplicação
  + which stylesheet format would you like to use? Nesse caso, basta ir com a setinha e escolher CSS, que é a folha de estilo que iremos utilizar
* cd spaangular – comando para navegar até o seu projeto. Pode ser substituído abrindo o CMD diretamente na pasta da aplicação
* ng serve -o – comando para iniciar o servidor e ficar observando os arquivos. A cada mudança nos arquivos ele reconstrói “rebuild” a aplicação
  + o comando “-o” é uma abreviação para “—open”, e abre o seu browser padrão após finalizar a compilação
* vai criar a estrutura inicial do projeto

**Diretórios e arquivos:**

* **node\_modules** – aqui estão todos os pacotes e dependências que o Angular NPM baixou e instalou na nossa aplicação
* **src** – dentro de SRC existem algumas pastas que já traz pronta e que fazem parte da estrutura inicial
* **app** – dentro da pasta APP é onde ficam todos os nossos componentes e arquivos de módulo
* **assets** – é onde colocaremos nossos arquivos de estilo comuns em toda a aplicação, assim como imagens, etc
* **environments** – essa pasta nos ajuda a criar diferentes ambientes para que a aplicação tenha um comportamento único em cada um deles, facilitando no deploy do código
* **package.json** – nesse arquivo estão as dependências do nosso projeto. Aqui podemos ver as versões utilizadas em cada pacote
* **app.module.ts** – onde faremos a importação dos componentes

**Angular Material**

* para utilizar o angular material, temos que ter o angular cli instalado e usar o seguinte código: **ng add @angular/material**

**Criando nosso primeiro componente**

**app.module.ts**

* é o módulo raiz da nossa aplicação
* sempre que formos utilizar um novo componente, precisamos adicioná-lo neste arquivo para que nossa aplicação passe a enxerga-lo
* tantos os componentes criados por nós quanto os que importamos (ex: do Angular Material) deverão ser colocados lá
* no arquivo, colocamos o import e de onde veio e também adicionar o nome dele no “imports”
* import { HomeComponent } from './home/home.component';
* declarations: [
* AppComponent,
* HomeComponent
* ],
* imports: [
* BrowserModule,
* o arquivo **app.component.html** é aquele arquivo da página inicial do Angular, que foi criado quando iniciamos um novo projeto, contendo a página de Boas Vindas do Angular
* **app.component.spec** é para fazermos testes
* **app-routing.module.ts** é responsável por mapear as rotas, falando para nossa aplicação qual arquivo ele deve abrir
* no arquivo **app.component.css** podemos definir o estilo que será aplicado para este componente em específico

Para criar o componente, devemos digitar o comando **ng g**(generate) **c**(componente) **home**. Vai criar os arquivos CSS, HTML, TypeScript e Spec (pode ser excluído.

**Definindo nosso módulo como inicial**

No arquivo **app-routing.module.ts**, vamos editar sua rota, para que ele abra como default a página **HTML** do componente Home que acabamos de criar. Para isso, vamos inserir o código:

const routes: *Routes* = [{

    path: '',

    component, HomeComponent

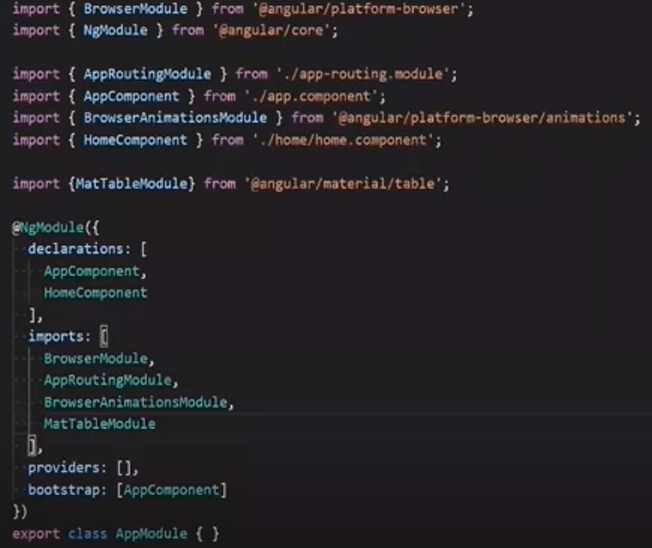
}];

Em seguida, iremos limpar quase todo o código padrão do nosso arquivo **app.component.html**, deixando apenas o código **<router-outlet></router-outlet>**. Esse código é responsável por abrir os demais componentes que formos criando dentro do Angular.

Feito isso, execute novamente o comando **ng serve -o** para visualizar nossa nova aplicação.

**Adicionando um componente do Angular Material**

No site Angular Material, escolhemos algum componente. Após escolhido, é preciso importar esse novo componente para dentro do nosso projeto **app.module.ts**, conforme a imagem abaixo





Após isso, se copia o código HTML, podendo ter um componente “filho” junto, sendo necessário importa-lo também. Exemplo desse código: <mat-icon>. É necessário achar o componente filho e importa-lo também. Nesse caso, estava na aba “Icon” do site material.angular.io.

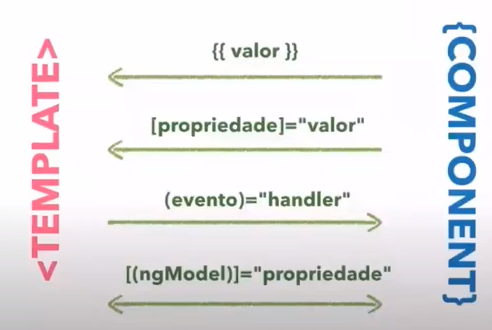


Após copiar, coloca o código no html desejado.



**Data Biding**

* é uma forma de associarmos informações que estão no componente para o template e vice-versa
* a **interpolação** nada mais é que um recurso de **“embedar”** expressões dentro de uma área delimitada por {{ .. }}. Dentro dessa área podemos trazer variáveis e métodos do nosso TS, realizar cálculos matemáticos e também realizar operações lógicas



Não há maneira correta de realizar Data Biding, pois cada situação pode exigir uma forma para aplicar. Existem 4 formas:

* **Interpolação** – {{ valor }}. associa informação do componente para o template (HTML)
* **Property Binding** – [propriedade]=“valor”. associa informação do componente para o template (HTML)
* **Event Binding** – (evento)=”handler”. associa informação do template (HTML) para o componente
* **Two-Way Data Binding** – [(ngMbodel)]=”propriedade”. associa informação entre ambos, ou seja, mantém ambos atualizados (componente e template (HTML)).

**Diretivas: ngIf /ngFor**

* são marcadores em um elemento DOM (como um atributo) que informam ao Angular para anexar um comportamento especificado a um elemento existente
* existem muitas diretivas prontas que podemos usar e também podemos criar nossas próprias diretivas

**exemplos de diretivas:**

* **ngIf** (ng if) – está é uma diretiva que você adiciona um elemento na marcação, geralmente para um elemento de contâiner como um div
* **ngFor** – está é uma diretiva para processar cada item de um objeto iterável, gerando uma marcação para cada um. Ela é conhecida como uma diretiva estrutural, porque pode alterar o layout do DOM adicionando e removendo elementos DOM de visualização
* **ngInit** – inicia a aplicação. quando a aplicação é iniciada, ela faz o que está dentro do ngInit

**Event Emitter**

* **Input Property - @Input()** – usamos quando queremos receber dados de um componente pai
* **Output Property - @Output()** – usamos quando queremos enviar dados de um componente filho para um componente pai

**Life Cycle hooks**

Cada componente no Angular possui um conjunto de **Eventos de Ciclo de Vida** e eles ocorrem quando o componente é criado, renderizado, tem o valor de suas propriedades alteradas ou quando ele é destruído.

**Quando um desses eventos é chamado, o Angular invoca uma série de métodos que são executados imediatamente**.

Um exemplo de seu uso pode ser aplicado em um site de compras, onde conseguiríamos agir de acordo com cada mudança feita pelo usuário, ao adicionar um item no carrinho.

Consome bastante memória.