Componentes

Custom Elements
Html templates
Html Imports
Shadow DOM

Custom Elements

- Essa API é a fundação da arquitetura, é ela que nos permite criar elementos HTML do tipo:

<login> </login>
<barra-navegacao> </barra-navegacao>
<contatos> </contatos>

Html Templates

- A API de elementos HTML template definem como declarar fragmentos de html que permanecem inertes no carregamento da página, mas podem ser carregados quando o componente a que pertencem for utilizado.
- Templates funcionam como uma planta de uma casa, a mesma planta pode construir inúmeras casa idênticas. Um mesmo template pode ser utilizados em vários componentes diferentes.

Html Imports

- Com essa API é possível incluir marcações HTML uma dentro das outras.

```
<link rel="import" href="myfile.html">
```

Shadow DOM

- Shadow DOM é provavelmente a API mais complexa das 4. Ela permite combinar várias árvores DOM em uma única hierarquia e como elas se comunicam dentro do documento. Com essa técnica nós conseguimos encapsular código HTML e CSS em componentes, isto é, escondê-lo do restante do documento principal.
- Não existe no AngularJs 1 porém existe no AngularJs 2

Event Emitter

- Esse padrão não é obrigatório no framework, mas é interessante, pois facilita a transição para o framework Angular 2.0 que faz uso dele. Repare que quando usamos o componente declaramos que a propriedade *on-change* será representada pela função *change* da controladora do componente pai e que ela receberá o parâmetro *event*. Na controladora a função respeita o mesmo padrão (vide exemplo acima), recebendo event (o objeto modificado) e atribuindo-o à *this.id e this.nome*, no entanto, quando o componente filho invoca *onChange,* ele passa um objeto que possui a propriedade *event*.

Event Emitter

- Portanto, podemos concluir que existe um comportamento implícito no uso da interface de saída: ele assume que estará "embrulhado" em um objeto e que os parâmetros declarados são propriedades desse objeto.

Stateless X Stateful

- Um componente *stateless* tem por objetivo **apresentar** os controles necessários para um dado cenário ou subcenário da aplicação. Eles precisam apenas de uma controladora e um template, como temos feito até agora. Seu uso segue a perfeita descrição de um componente no framework angularjs em seu formato mais simples.
- Já um componente *stateful* é responsável por controlar um cenário da aplicação. Ele normalmente está associado à um estado do framework ui-router. Nele podemos acessar entidades externas ao cenário, como os serviços da aplicação que gerenciam o acesso ao backend ou à uma base local de dados.

Project Seed

- https://github.com/PatrickJS/NG6-starter