Animações.

O angular tem um modulo especifico para animações que é baseado em uma spec chamada web animations API. A partir do angular 4 as funções de animações pode ser importadas do módulo animations, antes era importada do core.

wildCard: utilizado para representar qualquer estado.

Void: usado para representar a transição de um elemento que não está na arvore de transição ainda, ou que vai sair dela.

Imporando o animations para o angular:

npm i --save @angular/animations@4.0.0

npm i --save web-animations-js

Em polyfills.ts:

import 'web-animations-js/web-animations.min.js';

Depois no app.module:

import { BrowserAnimationsModule} from '@angular/platform-browser/animations'

imports: [

    BrowserModule,

    BrowserAnimationsModule,

    HttpModule,

    SharedModule.forRoot(),

    RouterModule.forRoot(ROUTES, {preloadingStrategy: PreloadAllModules}),

  ]

Adicionando uma snackbar:

Criando o componente de snack bar:

Css:

.snackbar{

    min-width: 250px;

    margin-left: -125px;

    background-color: #333;

    color: #fff;

    text-align: center;

    border-radius: 2px;

    padding: 16px;

    position: fixed;

    z-index: 1;

    left: 80%;

    bottom: 30px;

}

Html:

<div class="snackbar">

    {{message}}

</div>

Ts:

import { Component, OnInit } from '@angular/core';

import {trigger, state, style, transition, animate} from "@angular/animations";

@Component({

  selector: 'mt-snackbar',

  templateUrl: './snackbar.component.html',

  styleUrls: ['./snackbar.component.css'],

  animations: [

    trigger('snack-visibility', [

      state('hidden', style({

        opacity: 0,

        bottom: 0,

      })),

      state('visible', style({

        opacity: 1,

        bottom: '30px'

      })),

//mostrando

      transition('hidden => visible', animate('500ms 0s ease-in')),

//escondendo

transition('visible => hidden', animate('500ms 0s ease-out'))

    ])

  ]

})

export class SnackbarComponent implements OnInit {

  message: string = 'Hello there!';

  constructor() { }

  ngOnInit() {

  }

}

Associando nosso trigger de animações com o elementos html:

<div class="snackbar" [@snack-visibility]="">

    {{message}}

</div>

Depois é necessário associar o estado da trigger com uma propriedade do componente e quando alterarmos o componente o estado da trigger vai mudar e a animação vai ocorrer naturalmente.

<div class="snackbar"  [@snack-visibility]="snackVisibility">

    {{message}}

</div>

Para isso foi criada a variável snackVisibility que irá manipular a visualização da animação:

toggleSnack(){

    this.snackVisibility = this.snackVisibility == 'hidden' ? 'visible' : 'hidden'

    console.log('Snack', this.snackVisibility)

  }

Método criado apenas para teste, quando clica ele seta o inverso da variável, exibindo e escondendo a snackbar.

Easy-in: entra acelerando

Usando Observable com SnackBar:

Para podemos utilizar o SnackBAr onde quisermos, primeiro criamos um serviço de notificação:

import { EventEmitter } from '@angular/core';

export class NotificationService {

    notifier = new EventEmitter<string>();

    notify(message: string){

        this.notifier.emit(message);

    }

}

Depois adicione esse provider a lista de providers do módulo compartilhado.

Adicionando snackbar no shopping cart:

Primeiro chamamos nosso serviço de notificação no serviço do cart:

 addItem(item: Menu){

        //verificando se o item já existe

        let foundItem = this.items.find(mItem => mItem.menuItem.id == item.id);

        //se sim, adicionamos mais um na sua quantidade

        if(foundItem){

           this.increaseQty(foundItem)

         //se não, adicionamos o item ao array;

        }else{

            this.items.push(new CartItem(item))

        }

//aqui

        this.notificationSrvc.notify(`Você adicionou o item ${item.name}`)

    }

    removeItem(item: CartItem){

        this.items.splice(this.items.indexOf(item), 1)

//aqui

        this.notificationSrvc.notify(`Você removeu o item ${item.menuItem.name}`)

    }

\*\* new EventEmmiter é um observable.

Utilizando o timer do observable para esconder a snackbar:

import { Observable } from 'rxjs/Observable';

import 'rxjs/add/observable/timer'

  this.notificationSrvc.notifier.subscribe(message=>{

      this.message = message;

      this.snackVisibility = 'visible';

      Observable.timer(3000).subscribe(timer=>{

         this.snackVisibility ='hidden'

      });

Método ‘do‘ da biblioteca rxjs: permite executar uma ação no instante que chega aquela mensagem, que é completamente diferente do subscribe que coloca um listener no observable e só a partir daquele ponto que o observable vai notificar você.

Switchmap: ai invés de trocar a mensagem igual ao map comum, ele troca o observale inteiro.

Resolvendo o problema de múltiplas mensagens no snackbar:

  this.notificationSrvc.notifier.do(message=>{

      this.message = message;

      this.snackVisibility = 'visible';

     })

     .switchMap(message => Observable.timer(3000)

     .subscribe(timer=> this.snackVisibility ='hidden' ))

  }

Com isso as mensagens não serão cosrtadas, apenas os textos substituídos.

Isso acontece pois o switchmap, quando chega um novo subscribe, ele faz um unsubscribe do antigo e o sobrescreve com o novo. Por isso não existem mais times concorrendo entre si, pois ele cancela o timer antigo e inicia um novo.