**Now I am at: src/app/modals**

HTML

**Tags:** SVG, DEFS, PATH

**Google Maps:**

<reference types="@types/googlemaps" />

**Rect & Animate:**

<rect width="100%" x="0" y="0">

              <animate id="f1" attributeName="opacity" from="0" to="1" dur="0.6s" begin="0s; f5.end + 0.3s" fill="freeze"></animate>

              <animate attributeName="opacity" from="1" to="0" dur="0.5s" begin="f5.end" fill="freeze"></animate>

            </rect>

**JS**

**Propagination:**

event.stopPropagation();

event.stopImmediatePropagation();

**ReqAnimFrame:**

window.requestAnimationFrame(animate)

**Array.fill:**

Array(…).fill(0)

**RegEx:**

let regex = new RegExp(this.userInput.toString(), 'gi');

    this.generatedId =  this.listingItem.id.toString().replace(regex, `<b>${this.userInput}</b>`);

**newWindow:**

let dotRef = new window['Dotdotdot'](this.sellerCardName.nativeElement, {

          ellipsis: '\u2026 ',

          height: 70,

          truncate: 'letter',

          watch: 'window'

        });

**DefineProperty:**

Object.defineProperty(target, propertyKey, { value: {}, enumerable: true, configurable: true })

**Symbol:**

const timeoutKey = Symbol()

TS

**Angular**

**Device Detector:**

constructor(

    private deviceDetectorService: DeviceDetectorService,

    @Inject(PLATFORM\_ID) private platformId: string

  ) {

    this.isBrowser = isPlatformBrowser(this.platformId);

  }

**NgZone & CDR:**

constructor(

    private ngZone: NgZone,

    private cdr: ChangeDetectorRef

  ) {}

this.cdr.detectChanges();

this.ngZone.runOutsideAngular(() => {})

**file:**

<input appUploadFile (file)="uploadFile($event)" multiple />

**Detect Changes:**

constructor(private cdr: ChangeDetectorRef) {}

this.cdr.detectChanges();

**Sanitizer:**

import { DomSanitizer, SafeResourceUrl } from '@angular/platform-browser';

constructor(private sanitizer: DomSanitizer) {}

getIframeUrl(): SafeResourceUrl {

    return this.sanitizer.bypassSecurityTrustResourceUrl(this.iframeUrl);

  }

**Translate Service:**

import { TranslateService } from '@ngx-translate/core';

constructor(private translate: TranslateService) {}

this.translate.instant('listing.year')

**IUnsubscribeOnDestroy:**

export class PriceComponent implements OnInit, IUnsubscribeOnDestroy

@UnsubscribeOnDestroy()

**\*ngTemplateOutlet with ng-content**

<ng-content \*ngTemplateOutlet="totalPrice"></ng-content>

**Abstract control:**

import { AbstractControl } from '@angular/forms';

setSpace(control: AbstractControl, value: number) {

    control.setValue(String(value));

  }

**getBoundingCLientRect:**

this.tooltipElement.nativeElement.getBoundingClientRect()

**Renderer2.listen:**

\_initTouchOutside() {

    this.globalListenFunc = this.renderer.listen('window', 'touchstart', event => this.handleClickOutside(event));

  }

**newTypeOf Providers:**

@Component({

  selector: 'app-amount-input',

  templateUrl: './amount-input.component.html',

  styleUrls: ['./amount-input.component.scss'],

  providers: [

    {

      provide: NG\_VALUE\_ACCESSOR,

      useExisting: forwardRef(() => AmountInputComponent),

      multi: true,

    },

  ],

})

**Injector:**

constructor(private translate: TranslateService, private injector: Injector) {}

**Path & others:**

ngOnChanges(changes: SimpleChanges) {

    const hasChanges = pipe(

      path(['options', 'currentValue', 'length']),

      equals(path(['options', 'previousValue', 'length'])),

      not,

    )(changes);

  }

**View Providers:**

@Component({

  selector: 'app-field-error',

  templateUrl: './field-error.component.html',

  styleUrls: ['./field-error.component.scss'],

  viewProviders: [{ provide: ControlContainer, useExisting: FormGroupDirective }],

})

**Change detector ref:**

constructor(private cdr: ChangeDetectorRef) {}

this.cdr.detectChanges();

**Attribute modelChange:**

(ngModelChange)="updateChanges()"

*Unknown*

**Window.placement:**

if (window['ado'] && window['ado'].placement) {

        this.stopChecking$.next();

        this.stopChecking$.complete();

        window['ado'].placement({id: this.correctBanner.id, server: 'port80ge.adocean.pl' });

      }