# Manual: How to deploy currency symbols and how to change format number to locale culture in Angular

## Problem

There are fields in the projects that requires a currency symbol. For example:

Price: 1449999,99 $

Besides, numbers are showed using dots/commas/other symbols to make them clearer.

(Culture = USA) Price: 1,449,999.99 $

(Culture = Switzerland) Price: 1’449’999.99 CHF

(Culture = France) Price: 1.449.999,99 €

By default numbers in Angular don’t display currency symbols and they are not formatted.

## How to implement it

Next languages are used in the Smurfit Kappa website:

* ar-MA: Arabic - Morocco (ar-MA)
* bg-BG: Bulgarian - Bulgaria (bg-BG)
* de-DE: German - Germany (de-DE)
* en-US: English - USA (en-US)
* es-CO: Spanish - Colombia (es-CO)
* es-ES: Spanish - Spain (es-ES)
* fr-FR: French - France (fr-FR)
* lt-LT: Lithuanian - Lithuania(lt-LT)
* nb-NO: Norwegian - Norway (nb-NO)
* pt-PT: Portuguese - Portugal (pt-PT)
* sv-SE: Swedish - Sweden (sv-SE)

1.- In app.module.ts, next lines must be added

|  |
| --- |
| **import** { NgModule } **from '@angular/core'**; **import** { BrowserModule } **from '@angular/platform-browser'**;  **import** { AppRoutingModule } **from './app-routing.module'**; **import** { AppComponent } **from './app.component'**; **import** { NavComponent } **from './nav/nav.component'**; **import** { FormsModule } **from '@angular/forms'**; **import** { *registerLocaleData* } **from '@angular/common'**; *// Activate only the one that is going to be used // import localeAr from '@angular/common/locales/ar'; // import localeBg from '@angular/common/locales/bg'; // import localeDe from '@angular/common/locales/de';* **import** localeEn **from '@angular/common/locales/en'**; *// import localeEs from '@angular/common/locales/es'; // import localeFr from '@angular/common/locales/fr'; // import localeLt from '@angular/common/locales/lt'; // import localeNb from '@angular/common/locales/nb'; // import localePt from '@angular/common/locales/pt'; // import localeSv from '@angular/common/locales/sv';* ***registerLocaleData*(localeEn);**  @NgModule({  **declarations**: [  AppComponent,  NavComponent  ],  **imports**: [  BrowserModule,  AppRoutingModule,  FormsModule  ],  **providers**: [],  **bootstrap**: [AppComponent] }) **export class** AppModule { } |

“RegisterLocaleData” is a function that will set the lOCALE\_ID setting, the language that will be used in the website.

For example, registerLocaleData(localeEn) set LOCALE\_ID = ‘en-EN’.

Note: if English is not going to be the by default language, it is necessary to uncomment the correspondent line of import. For example, if Portuguese was required as by default language:

|  |
| --- |
| *// import localeEn from '@angular/common/locales/en';* *// import localeEs from '@angular/common/locales/es'; // import localeFr from '@angular/common/locales/fr'; // import localeLt from '@angular/common/locales/lt'; // import localeNb from '@angular/common/locales/nb';* **import localePt from '@angular/common/locales/pt';**  *…* |
| ***registerLocaleData*(localePt);** |

2.- In the component (called Nav in this example) there are two pipes CurrencyPipe and DecimalPipe that are used to format correctly the number:

|  |
| --- |
| Nav.component.html |
| <**p**>Locale: {{ **localeSetting** }}</**p**> <**p**>Currency code: {{ **currencyCodeSetting** }}</**p**> <**br**> <**select class="custom-select"  (change)="changeLang(*$event*.target.value)"**>  <**option value="ar-MA" [selected] = "localeSetting == 'ar-MA'"**> Arabic - Morocco (ar-MA)</**option**>  <**option value="bg-BG" [selected] = "localeSetting == 'bg-BG'"**> Bulgarian - Bulgaria (bg-BG)</**option**>  <**option value="de-DE" [selected] = "localeSetting == 'de-DE'"**> German - Germany (de-DE)</**option**>  <**option value="en-US" [selected] = "localeSetting == 'en-US'"**> English - USA (en-US)</**option**>  <**option value="es-CO" [selected] = "localeSetting == 'es-CO'"**> Spanish - Colombia (es-CO)</**option**>  <**option value="es-ES" [selected] = "localeSetting == 'es-ES'"**> Spanish - Spain (es-ES)</**option**>  <**option value="fr-FR" [selected] = "localeSetting == 'fr-FR'"**> French - France (fr-FR)</**option**>  <**option value="lt-LT" [selected] = "localeSetting == 'lt-LT'"**> Lithuanian - Lithuania (lt-LT)</**option**>  <**option value="nb-NO" [selected] = "localeSetting == 'nb-NO'"**> Norwegian - Norway (nb-NO)</**option**>  <**option value="pt-PT" [selected] = "localeSetting == 'pt-PT'"**> Portuguese - Portugal (pt-PT)</**option**>  <**option value="sv-SE" [selected] = "localeSetting == 'sv-SE'"**> Swedish - Sweden (sv-SE)</**option**> </**select**> <**br**><**br**><**br**><**br**> <**p**>Price: {{ **price** }} (No format)</**p**> <**p**>Price: **{{ price | *currency*:currencyCodeSetting:'symbol':'1.2-2':localeSetting }}**</**p**> *<!-- :1.2-2 al menos un digito de la parte entera, y como mínimo dos decimales y como máximo dos decimales -->* <**input [ngModel]="**price | *currency*:currencyCodeSetting:'symbol':'1.2-2':localeSetting **"  (ngModelChange)="price=currencyInputChanged(*$event*)"**/> <**br**><**br**><**br**><**br**> <**p**>Number: {{ **numberVal** }} (No format)</**p**> <**p**>Number: {{ **numberVal** | number:'1.2-2':**localeSetting** }}</**p**> *<!-- :1.2-2 al menos un digito de la parte entera, y como mínimo dos decimales y como máximo dos decimales -->* <**input [ngModel]="numberVal | number:'1.2-2':localeSetting "  (ngModelChange)="numberVal=currencyInputChanged(*$event*)"**/> |

currenclyCodeSetting is a variable that contains the code of the currency (for example USD for dollar, EUR for euro, etc).

localeSetting is a variable that contains the locale culture (for example, ‘en-EN’ for English, ‘es-CO’ for Spanish- Colombia, sv-SE forSwedish – Sweden…).

price | *currency*:currencyCodeSetting:'symbol':'1.2-2':localeSetting

First, the price (for example 9876543210.98765) is formatted with a currencyPipe *currency*:’USD’ to generate $9876543210.98765.

Second, 'symbol':'1.2-2' is a pipe that sets how many digits will be in the full part and how many digits will be in the decimal part. 2 decimals maximum. $9876543210.98765 => $9876543210.99.

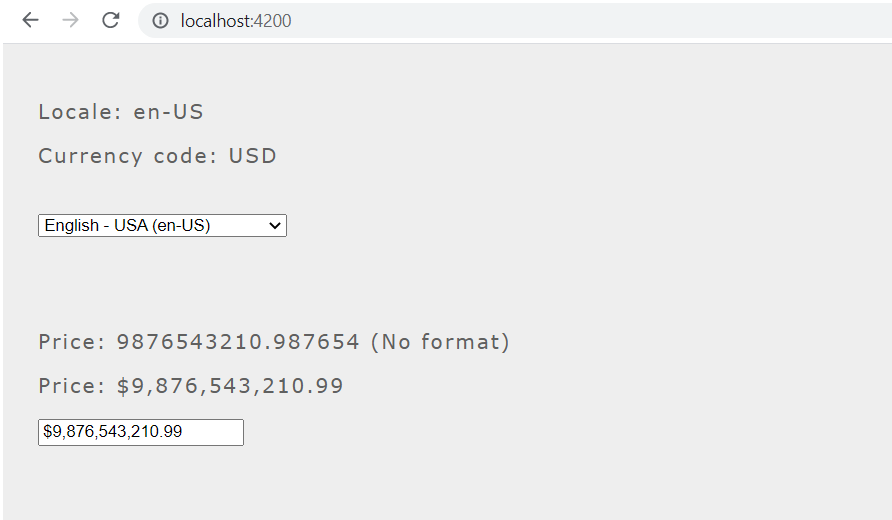
The last pipe (:localeSetting) will replace it for locale culture format ( ‘en-EN’). For example, $9876543210.99 => $9,876,543,210.99.

The problem appears in the textbox and in the double data binding.

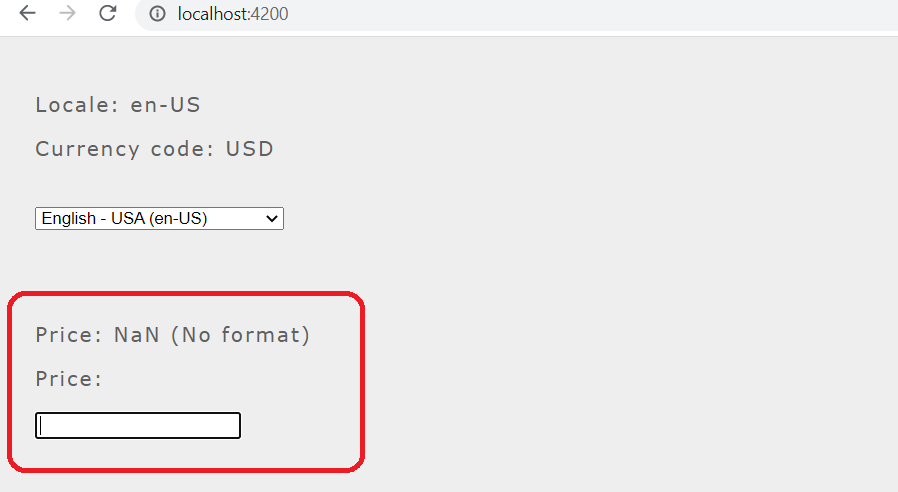
9876543210.99

When the pipes are applied {{ price | currency:'USD':'symbol':'1.2-2':'en-EN' }}, it is converted to:

9.876.543.210,99 €



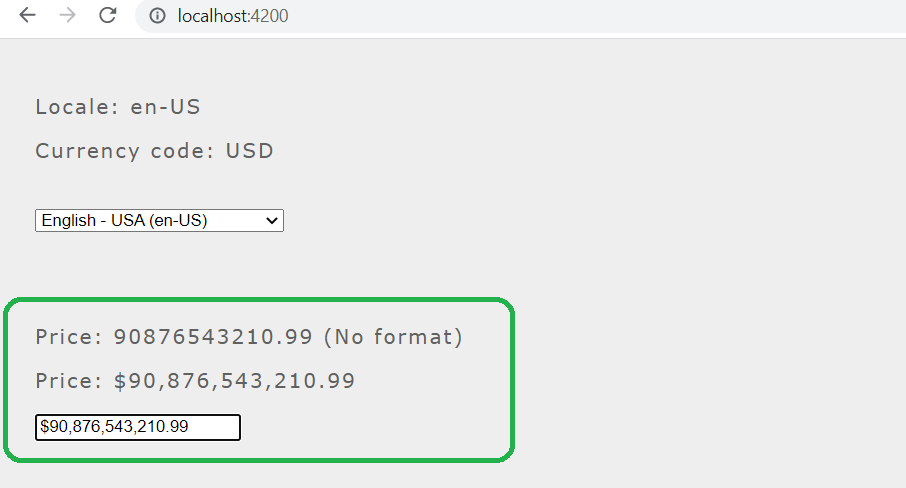
But if customer clicks on the textbox and write a digit, it crashes and a NaN is returned.



When the textbox was modified, "90.876.543.210,99 €" was parsed to number (instead of "90876543210.99").

The solution is defined onChange event and it will convert "90.876.543.210,99 €" to "90876543210.99" when the textbox was modified.

Each culture is different. White spaces, commas, dots, etc must be removed. And the decimal comma must be replaced for decimal dot.



To solve this, next function was added to the nav.component.ts. It replaces wrong characters for the correct number that can be assigned to the number variable:

|  |
| --- |
| Nav.component.ts |
| */\*\*  \* Remove currency symbols from value to avoid to get NaN.  \** ***@param*** *value textfield value ($34,44)  \** ***@return*** *textfield value without currency symbol (34,44)  \*/* currencyInputChanged(value: **any**): **any** {  **let** num;  **switch** (**this**.**localeSetting**) {  **case 'ar-MA'**:  *// 9,876,543,210.99 د.م => 9876543210.99* num = value.replace(/[€\s,]/g, **''**); *// \s => replace white spaces* num = num.replace(/[د.م]/g, **''**); *// currency symbol* **return** Number(num);  **case 'bg-BG'**:  *// 9876543210,99 € => 9876543210.99* num = value.replace(/[€\s]/g, **''**); *// \s => replace white spaces* num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'de-DE'**:  *// 9.876.543.210,99 € => 9876543210.99* num = value.replace(/[€\s.]/g, **''**); *// \s => replace white spaces* num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'en-US'**:  *// $9,876,543,210.99 => 9876543210.99* num = value.replace(/[$€,]/g, **''**);  **return** Number(num);  **case 'es-CO'**:  *// 9.876.543.210,99 COP => 9876543210.99* num = value.replace(/[A-Z\s.]/g, **''**); *// \s => replace white spaces* num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'es-ES'**:  *// 9.876.543.210,99 € => 9876543210.99* num = value.replace(/[€\s.]/g, **''**); *// \s => replace white spaces* num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'fr-FR'**:  *// 9 876 543 210,99 € => 9876543210.99* num = value.replace(/[€\s]/g, **''**);  num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'lt-LT'**:  *// 9 876 543 210,99 € => 9876543210.99* num = value.replace(/[€\s]/g, **''**);  num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'nb-NO'**:  *// kr 9 876 543 210,99 => 9876543210.99* num = value.replace(/[a-zA-Z\s]/g, **''**);  num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'pt-PT'**:  *// € 9.876.543.210,99 => 9876543210.99* num = value.replace(/[€\s.]/g, **''**);  num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'sv-SE'**:  *// 9 876 543 210,99 kr => 9876543210.99* num = value.replace(/[a-zA-Z$€,]/g, **''**);  num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **default**:  *// Display error message because in other case, database will receive a wrong value  alert*(**'ERROR: this currency format is not supported'**);  } } |

The rest of changes that were done are:

* Import locales for all the supported languages:

|  |
| --- |
| Nav.component.ts |
| **import** { Component, OnInit } **from '@angular/core'**;  **import** { *registerLocaleData* } **from '@angular/common'**; **import** localeAr **from '@angular/common/locales/ar'**; **import** localeBg **from '@angular/common/locales/bg'**; **import** localeDe **from '@angular/common/locales/de'**; **import** localeEn **from '@angular/common/locales/en'**; **import** localeEs **from '@angular/common/locales/es'**; **import** localeFr **from '@angular/common/locales/fr'**; **import** localeLt **from '@angular/common/locales/lt'**; **import** localeNb **from '@angular/common/locales/nb'**; **import** localePt **from '@angular/common/locales/pt'**; **import** localeSv **from '@angular/common/locales/sv'**;  @Component({  **selector**: **'app-nav'**,  **templateUrl**: **'./nav.component.html'**,  **styleUrls**: [**'./nav.component.scss'**] }) |

* Next attributes will contain information about locale, currency code…

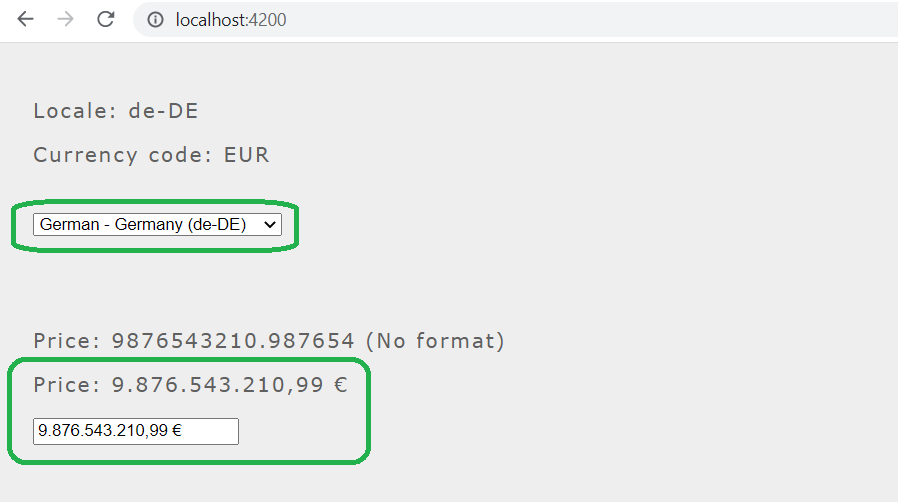
|  |
| --- |
| Nav.component.ts |
| **export class** NavComponent **implements** OnInit {   **localeSetting = 'en-EN';  currencyCodeSetting = 'USD';  lang: string;**  **numberVal** = 1234567890.123456;  **price** = 9876543210.987654; |

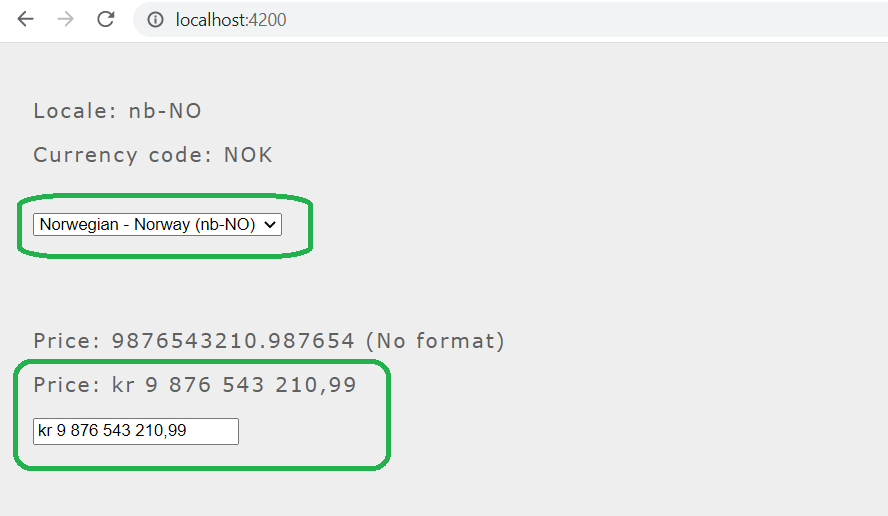
* The selected language will be stored in the localStorage. If it wasn’t, ‘en-US’ will be set by default.
  + After that, in setRegisterLocaleData function, the LOCALE\_ID of the application is set.
  + Then, the setCurrencyCode function defines the code of the currency (USD for dollar, EUR for EUR…).

|  |
| --- |
| Nav.component.ts |
| */\*\*  \* Constructor.  \*/* **constructor**() {  **this**.**localeSetting** = ***localStorage***.getItem(**'lang'**) || **'en-US'**;  **this**.setRegisterLocaleData();  **this**.setCurrencyCode(); }    */\*\*  \* Set locale culture  \*/* setRegisterLocaleData(): **void** {  **switch** (**this**.**localeSetting**) {  **case 'ar-MA'**:  *registerLocaleData*(localeAr);  **break**;  **case 'bg-BG'**:  *registerLocaleData*(localeBg);  **break**;  **case 'de-DE'**:  *registerLocaleData*(localeDe);  **break**;  **case 'en-US'**:  *registerLocaleData*(localeEn);  **break**;  **case 'es-CO'**:  *registerLocaleData*(localeEs);  **break**;  **case 'es-ES'**:  *registerLocaleData*(localeEs);  **break**;  **case 'fr-FR'**:  *registerLocaleData*(localeFr);  **break**;  **case 'lt-LT'**:  *registerLocaleData*(localeLt);  **break**;  **case 'nb-NO'**:  *registerLocaleData*(localeNb);  **break**;  **case 'pt-PT'**:  *registerLocaleData*(localePt);  **break**;  **case 'sv-SE'**:  *registerLocaleData*(localeSv);  **break**;  **default**:  *registerLocaleData*(localeEn);  **break**;  } }  */\*\*  \* Set currency code depending on language selected.  \*/* setCurrencyCode(): **void** {  **switch** (**this**.**localeSetting**) {  **case 'ar-MA'**:  **this**.**currencyCodeSetting** = **'MAD'**; *// Dirham marroqui* **break**;  **case 'bg-BG'**:  **this**.**currencyCodeSetting** = **'EUR'**; *// Euro* **break**;  **case 'de-DE'**:  **this**.**currencyCodeSetting** = **'EUR'**; *// Euro* **break**;  **case 'en-US'**:  **this**.**currencyCodeSetting** = **'USD'**; *// Dollar* **break**;  **case 'es-CO'**:  **this**.**currencyCodeSetting** = **'COP'**; *// Peso colombiano* **break**;  **case 'es-ES'**:  **this**.**currencyCodeSetting** = **'EUR'**; *// Euro* **break**;  **case 'fr-FR'**:  **this**.**currencyCodeSetting** = **'EUR'**; *// Euro* **break**;  **case 'lt-LT'**:  **this**.**currencyCodeSetting** = **'EUR'**; *// Euro* **break**;  **case 'nb-NO'**:  **this**.**currencyCodeSetting** = **'NOK'**; *// Corona noruega* **break**;  **case 'pt-PT'**:  **this**.**currencyCodeSetting** = **'EUR'**; *// Euro* **break**;  **case 'sv-SE'**:  **this**.**currencyCodeSetting** = **'SEK'**; *// Corona sueca* **break**;  **default**:  **this**.**currencyCodeSetting** = **'USD'**;  **break**;  } } |

* When another language is selected in combobox control, current locale is stored in localStorage and the page is reloaded:

|  |
| --- |
| Nav.component.ts |
| */\*\*  \* Change language.  \** ***@param*** *lang language selected (es-ES).  \*/* changeLang(lang): **void**{  ***localStorage***.setItem(**'lang'**, lang);  **this**.**localeSetting** = lang;  **this**.setRegisterLocaleData();  **this**.setCurrencyCode();  ***window***.**location**.reload(); } |





The complete code:

|  |
| --- |
| Nav.component.ts |
| **import** { Component, OnInit } **from '@angular/core'**;  **import** { *registerLocaleData* } **from '@angular/common'**; **import** localeAr **from '@angular/common/locales/ar'**; **import** localeBg **from '@angular/common/locales/bg'**; **import** localeDe **from '@angular/common/locales/de'**; **import** localeEn **from '@angular/common/locales/en'**; **import** localeEs **from '@angular/common/locales/es'**; **import** localeFr **from '@angular/common/locales/fr'**; **import** localeLt **from '@angular/common/locales/lt'**; **import** localeNb **from '@angular/common/locales/nb'**; **import** localePt **from '@angular/common/locales/pt'**; **import** localeSv **from '@angular/common/locales/sv'**;  @Component({  **selector**: **'app-nav'**,  **templateUrl**: **'./nav.component.html'**,  **styleUrls**: [**'./nav.component.scss'**] }) **export class** NavComponent **implements** OnInit {   **localeSetting** = **'en-EN'**;  **currencyCodeSetting** = **'USD'**;  **lang**: **string**;  **numberVal** = 1234567890.123456;  **price** = 9876543210.987654;   */\*\*  \* Constructor.  \*/* **constructor**() {  **this**.**localeSetting** = ***localStorage***.getItem(**'lang'**) || **'en-US'**;  **this**.setRegisterLocaleData();  **this**.setCurrencyCode();  }   */\*\*  \* Remove currency symbols from value to avoid to get NaN.  \** ***@param*** *value textfield value ($34,44)  \** ***@return*** *textfield value without currency symbol (34,44)  \*/* currencyInputChanged(value: **any**): **any** {  **let** num;  **switch** (**this**.**localeSetting**) {  **case 'ar-MA'**:  *// 9,876,543,210.99 د.م => 9876543210.99* num = value.replace(/[€\s,]/g, **''**); *// \s => replace white spaces* num = num.replace(/[د.م]/g, **''**); *// currency symbol* **return** Number(num);  **case 'bg-BG'**:  *// 9876543210,99 € => 9876543210.99* num = value.replace(/[€\s]/g, **''**); *// \s => replace white spaces* num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'de-DE'**:  *// 9.876.543.210,99 € => 9876543210.99* num = value.replace(/[€\s.]/g, **''**); *// \s => replace white spaces* num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'en-US'**:  *// $9,876,543,210.99 => 9876543210.99* num = value.replace(/[$€,]/g, **''**);  **return** Number(num);  **case 'es-CO'**:  *// 9.876.543.210,99 COP => 9876543210.99* num = value.replace(/[A-Z\s.]/g, **''**); *// \s => replace white spaces* num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'es-ES'**:  *// 9.876.543.210,99 € => 9876543210.99* num = value.replace(/[€\s.]/g, **''**); *// \s => replace white spaces* num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'fr-FR'**:  *// 9 876 543 210,99 € => 9876543210.99* num = value.replace(/[€\s]/g, **''**);  num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'lt-LT'**:  *// 9 876 543 210,99 € => 9876543210.99* num = value.replace(/[€\s]/g, **''**);  num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'nb-NO'**:  *// kr 9 876 543 210,99 => 9876543210.99* num = value.replace(/[a-zA-Z\s]/g, **''**);  num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'pt-PT'**:  *// € 9.876.543.210,99 => 9876543210.99* num = value.replace(/[€\s.]/g, **''**);  num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **case 'sv-SE'**:  *// 9 876 543 210,99 kr => 9876543210.99* num = value.replace(/[a-zA-Z$€,]/g, **''**);  num = num.replace(/,/g, **'.'**); *// replace , for .* **return** Number(num);  **default**:  *// Display error message because in other case, database will receive a wrong value  alert*(**'ERROR: this currency format is not supported'**);  }  }   */\*\*  \* Set locale culture  \*/* setRegisterLocaleData(): **void** {  **switch** (**this**.**localeSetting**) {  **case 'ar-MA'**:  *registerLocaleData*(localeAr);  **break**;  **case 'bg-BG'**:  *registerLocaleData*(localeBg);  **break**;  **case 'de-DE'**:  *registerLocaleData*(localeDe);  **break**;  **case 'en-US'**:  *registerLocaleData*(localeEn);  **break**;  **case 'es-CO'**:  *registerLocaleData*(localeEs);  **break**;  **case 'es-ES'**:  *registerLocaleData*(localeEs);  **break**;  **case 'fr-FR'**:  *registerLocaleData*(localeFr);  **break**;  **case 'lt-LT'**:  *registerLocaleData*(localeLt);  **break**;  **case 'nb-NO'**:  *registerLocaleData*(localeNb);  **break**;  **case 'pt-PT'**:  *registerLocaleData*(localePt);  **break**;  **case 'sv-SE'**:  *registerLocaleData*(localeSv);  **break**;  **default**:  *registerLocaleData*(localeEn);  **break**;  }  }   */\*\*  \* Set currency code depending on language selected.  \*/* setCurrencyCode(): **void** {  **switch** (**this**.**localeSetting**) {  **case 'ar-MA'**:  **this**.**currencyCodeSetting** = **'MAD'**; *// Dirham marroqui* **break**;  **case 'bg-BG'**:  **this**.**currencyCodeSetting** = **'EUR'**; *// Euro* **break**;  **case 'de-DE'**:  **this**.**currencyCodeSetting** = **'EUR'**; *// Euro* **break**;  **case 'en-US'**:  **this**.**currencyCodeSetting** = **'USD'**; *// Dollar* **break**;  **case 'es-CO'**:  **this**.**currencyCodeSetting** = **'COP'**; *// Peso colombiano* **break**;  **case 'es-ES'**:  **this**.**currencyCodeSetting** = **'EUR'**; *// Euro* **break**;  **case 'fr-FR'**:  **this**.**currencyCodeSetting** = **'EUR'**; *// Euro* **break**;  **case 'lt-LT'**:  **this**.**currencyCodeSetting** = **'EUR'**; *// Euro* **break**;  **case 'nb-NO'**:  **this**.**currencyCodeSetting** = **'NOK'**; *// Corona noruega* **break**;  **case 'pt-PT'**:  **this**.**currencyCodeSetting** = **'EUR'**; *// Euro* **break**;  **case 'sv-SE'**:  **this**.**currencyCodeSetting** = **'SEK'**; *// Corona sueca* **break**;  **default**:  **this**.**currencyCodeSetting** = **'USD'**;  **break**;  }  }   ngOnInit(): **void** {}   */\*\*  \* Change language.  \** ***@param*** *lang language selected (es-ES).  \*/* changeLang(lang): **void**{  ***localStorage***.setItem(**'lang'**, lang);  **this**.**localeSetting** = lang;  **this**.setRegisterLocaleData();  **this**.setCurrencyCode();  ***window***.**location**.reload();  }  } |

Additional problem: Currently after writing a digit in a textbox, the position of the cursor is moved at the end of the string, that should be controlled and how to do it is explained in next article:

<https://stackoverflow.com/questions/40579014/set-cursor-position-in-textbox-javascript>