Önálló laboratórium 2

# Fejlesztői környezet

A projekt TypeScript és Less nyelvű, ezért a projektet fordítani kell. A fordítást a Gulp build rendszer végzi el. Ez úgy van beállítva, hogy magát a dátumválasztót egy fájlba, illetve minden integrációs fájlt külön-külön fájlba fordítson. Ez a config fájl a gulpfile.js.

A tesztek futtatását a Karma + Jasmine keretrendszerek végzik. A Karma configja a karma.conf.js fájlban található.

A projektet a VS Code IDE-ben fejlesztettem. A buildelés és a tesztelés megkönnyebbítése érdekében saját parancsokat definiáltam a .vscode/tasks.json fájlban. Így egy gombnyomással lehet buildelni vagy lefuttatni a teszteket.

# DateSlider

A DateSlider egy vagy több csúszkából álló dátumválasztó komponens. Egy DateSlider példányosításához szükség van egy HTMLElement-re, amibe bele fogja rakni a csúszkákat, illetve a komponens beállításaira.

## Belső modell

A komponens belső modellje a DateSliderModel. Ez külön-külön propertyben tárolja az év, hónap, nap, óra, perc, másodperc, időzóna adatait. A hónap és a nap 1-től kezdődik. Emiatt szükséges lehet a parszolás és a formatálás, mivel a komponenst használó fejlesztő nem biztos, hogy ilyen formátumú adatokkal akar dolgozni.

TODO összehasonlítás

## Parszolás és formatálás

A parszolás akkor történik, amikor a komponensnek be akarunk állítani egy új értéket. Az új érték beállításakor meghívódik a beállításoknál előre megadott parszoló függvény, ami átalakítja a megadott értéket az új belső modellé. Mivel a parszoláshoz nem biztos, hogy elegendő csak az érték, ezért a beállításoknál megadhatók plusz parszolási beállítások is. A komponens tartalmaz előre megírt parszoló függvényeket:

* stringből egy megadott formátum alapján

Plusz beállítások: { format: string }

* Date objektumból

Plusz beállítások: { type: "local" | "utc" }

* UNIX timestamp alapján

Plusz beállítások: { type: "milliseconds" | "seconds" }

Megadható saját parszoló függvény is, ekkor az alábbi szignatúrájú függvényt kell megvalósítania a fejlesztőnek:

* (input: any, options: any) => DateSliderModel

Ahol az input az új érték, az options a beállításoknál megadott parserOptions property és a visszatérési érték pedig a komponens egy belső modellje.

A formatálás akkor történik, amikor a komponensből valamilyen módon megkapjuk az éppen reprezentált dátumot. A formatálás nagyon hasonlóan működik a parszoláshoz és ugyanolyan típusú előre megírt formatáló függvényeket tartalmaz a komponens.

Saját formatáló függvény írásakor az alábbi szignatúrájú metódust kell megvalósítani:

* (input: DateSliderModel, options: any) => any

Ahol az input az komponens belső modellje, az options a beállításoknál megadott formatterOptions property és a visszatérési érték pedig a formatált érték.

## Validáció

A komponens aktuális értékének validálásához megadható minimális, maximális érték, illetve saját validáló függvény. A minimális és maximális értéket a fejlesztő által használt formátumban kell megadni, tehát le fog futni ezekre az értékekre a parszoló függvény, és annak az eredményét fogja a komponens összehasonlítani az épp aktuális értékkel.

Saját validáló függvény megadásakor az alábbi szignatúrájú függvényt kell implementálni:

* (value: any) => boolean

Ahol a value paraméter az éppen validálandó belső modell formatálva, tehát a fejlesztő által használt formátumban. A visszatérési érték pedig true, ha valid az érték, különben invalid.

A validáció után a komponens példányosításakor megadott HTMLElement-re rákerül a "date-slider-valid" vagy a "date-slider-invalid" class a validáció eredményétől függően.

## Callback

Fontos lehet, hogy a komponenst használó fejlesztő értesüljön arról, hogy megváltozott a komponens értéke. Erre használható a callback.onValueChanged beállítás. Megadható egy függvény, ami megkapja a változtatás kontextusát. A kontextus tartalmazza, hogy éppen valid-e a belső modell, illetve az előző értéket és az új értéket.

## Csúszkák

Megadható, hogy milyen csúszkákat tartalmazzon a példányosítandó komponens, ez részletesen taglalva lesz lentebb. Ritka, de szükséges lehet a csúszkák közti kommunikáció, mint például a nap csúszka maximuma az évtől és a hónaptól függően változhat. Ezt a komponens automatikusan kezeli.

## Beállítások

A fent taglalt működéshez szükséges beállításokat az alábbi formátumban adhatók meg.

{

value?: any;

sliders?: SliderOptions[];

parser?: "timestamp" | "string" | "date" | ((input: any, options: any) => DateSliderModel);

parserOptions?: any;

formatter?: "timestamp" | "string" | "date" | ((input: DateSliderModel, options: any) => any);

formatterOptions?: any;

callback?: {

onValueChanged?: (context: Context.ValueChangeContext) => void;

};

validation?: {

custom?: (value: any) => boolean;

min?: any;

max?: any;

};

}

Egy komponens példány egyes beállításait meg lehet változtatni az updateOptions és a replaceOptions függvényekkel. Az updateOptions módosítja az eddigi beállításokat az újakkal, tehát ezzel a függvénnyel lehet részleges beállításmódosításokat is végezni. A replaceOptions pedig teljesen lecseréli az eddigi beállításokat az újakra.

# Egy csúszka

## Felépítés

### Templatezés

## Belső modell

## Csúszka bütyök mozgatása és érték kiszámolás

### Csúszka szélén slide / expand

## Jelölők

## Callbackek

## Beállítások

# Tesztelés

## Unit tesztek

## Integrációs tesztek

# Integráció

## jQuery

## AngularJS