Doc Class Validato

HTML

- Nell'html sul campo da validare settare l'attributo (data-validate-type="nome"), il valore di ('nome') lo utilizzeremo nel js per andare a settare i vari parametri di validazione
- In caso di campi da comparare come password o email settare l'attributo (data-validate-compare="nome") su entrambi i campi da comparare (inserire lo stesso nome)

Js

- Per andare a richiamare il validatore settare (new Validator(paramValidator, 'submit', event)) dove paramValidator è un oggetto con i parametri da valitare submit è la stringa del tipo di evento che stiamo per passare, event è l'intero evento che sta chiamando il validatore.

ParamValidator

- E' un oggetto js composto da selectorForm= selettore jQuery della form
- **localization** = localization (ES, IT ...)
- **fieldValidation** = array di oggetti composti da :
  - **dataValidateType** = nome settato nell'omonimo parametro nell'input html
  - **validationRules** = una stringa con i parametrio che andremo a settare su quell'input es:

```
"required|specialCharacters|minLength-4|maxLength-12|postalCode"
```

  - **functionOtherValidation** = funzioni extra da settare che si azioneranno una volta passate le se validazioni precedentemente settate.

ValidationRules Paramiters

- Di seguito tutti i parametri da poter settare nella validazione (da inserire in validationRules):
  - **compare**: Compariamo il valore dei due campi ***(field, value)***
  - **forbiddenCharacters**: Controlliamo se sono stati inseriti caratteri uguali a quelli che noi reputiamo non ammissibili e ritorniamo un errore ***(field, value)***
  - **replaceString**: Sostituiamo i valori del value andando a sostituire i caratteri speciali in base all'oggetto objReplaceCharacter ***(field)***
  - **checkTaxIdEs**: Controlliamo se inserto un valore che corrispende alle regole di DNI CIF NIE senza fare differenza ***(field)***
  - **checkTaxId**: Controlliamo che il taxId abbia un path valido ***(field)***
  - **required**: Se impostato il valore del campo non può essere vuoto o null ***(field)***
  - **postalCode**: Controlliamo che il postal code inserito sia valido tramite le regex per località ***(field)***

- **phoneLv1**: Controlliamo che sia stato inserito un numero , accettiamo che inizi con 00 oppure con il + ***(field)***
- **phoneLv2**: Passiamo prima per il lv1 eseguimao il controllo sul prefisso internazionale +39/+34/0039/ecc.. in base alla localizzazione ***(field)***
- **phoneLv3**: Passiamo prima per il lv2 -> lv1 controlliamo il prefisso di cellulare o fisso 331/0773/ecc... in base alla localizzazione ***(field)***
- **email**: Controlliamo tramite 2 regex diverse se quella inserita è una email 'valida' controlliamo solo il phat ***(field)***
- **minLength**: Controlliamo il min length ***(field, value)***
- **maxLength**: Controlliamo il max Length ***(field, value)***
- **exactLength**: Controlliamo il length esatto ***(field, value)***
- **alpha**: Controlliamo che sono stati inseriti solo lettere ***(field)***
- **alphaNumeric**:Controlliamo sono stati inseriti solo numeri e lettere ***(field)***
- **alphaDash**: Controlliamo sono stati inseriti lettere accettando solo il trattino centrale '-' ***(field)***
- **numeric**: Controlliamo sono stati inseriti solo numeri ***(field)***
- **integer**: Controlliamo se è stato inserito un numero intero accettiamo il trattino per i numeri negativi es: -42 ***(field)***
- **decimal**: Controlliamo se è stato inserito un numero decimale accettiamo il trattino per i numeri negativi es: -42.90 ***(field)***
- **isNatural**: Controlliamo se è stato inserito un numero naturale positivo es: 98 ***(field)***
- **isNaturalNoZero**: Controlliamo se è stato inserito un numero naturale che non inizia con 0 positivo es: 98 ok 098 no ***(field)***
- **validIp**: Controlliamo se è stato un indirizzo ip valido , controlliamo il phat ***(field)***
- **validUrl**: Controlliamo se è stato inserito un url valido , controlliamo il phat(validiamo l'inserimento dell' http/https) es :https://regex101.com/ ok www.regex101.com/ no ***(field)***
- **validCreditCard**: Controlliamo se è stato inserito numero valido di carta di credito ***(field)***
- **isFileType**: Controlliamo l'estensione di un file caricato ***(field, value)***

INFO

I campi che richiedono il ***value*** vanno settati con ***nomeproprietà-value***

- es: **minLength-12**
- per quanto riguarda il compare va settato con value il valore del ***data-validate-compare*** impostato nel'html

---

RESULT

La classe Validator setta un paramento interno ***isValidForm*** a true o false se viene trovato un valore che non passa la validazione.

- Perciò dove necessario possiamo settare :
  - let validation = new Validator(paramValidator, 'submit', event);
- Prendere il:
  - validation.isValidForm che avrà il valore bolleano .

INFO

Il validatore setta in automatico :

- La classe **is-valid** sull'input per avere un feedback user front-end se viene usato bootstrap o se vengono uste le stesse classi.
- Setta in automatico i **messaggi di validazione**, prendendoli dall'oggetto interno (messageVal) che ha come chiave di riferimento il nome della validazione che non verrà passata.

Essendo questi automatismi si raccomanda di inserire sotto l'input da validare un div con class invalidfeedback anche vuoto, la classe setterà in automatico il tutto e bootstrap si occuperà di mostrare o meno il div.

Es input html con bootstrap style :

```
<div class="user-box">
 <input type="password" pattern=".{8,50}"
id="input_account_password"name="input_account_password" value="" data-validate-
type="password" data-validate-compare="passwordConfirm" required>
<label for="input_account_password"class="label-form">{password} *</label>
 <div class="invalid-feedback"{message}</div>
</div>
```

Es Obj di parametri da validare :

```
const paramValidator = {
   selectorForm: "#formRubricaEdit",
   localization: localStorage.getItem("LOCALE").toUpperCase(),
   fieldValidation: [{
       dataValidateType: 'postalCode',
       validationRules: 'required|specialCharacters|minLength-4|maxLength-
12|postalCode',
       functionOtherValidation: '',
   }, {
       dataValidateType: 'phone',
       validationRules: 'required|specialCharacters|minLength-8|maxLength-
12|phoneGeneric|phoneLv3',
       functionOtherValidation: '',
   }, {
       dataValidateType: 'email',
       validationRules: 'required|specialCharacters|minLength-4|email',
       functionOtherValidation: '',
   }, {
       dataValidateType: 'alias',
```

```
        validationRules: 'required|minLength-4|maxLength-
12|replaceString|forbiddenCharacters-1',
        functionOtherValidation: '',
    }],
};
```

Es funzione che valida un input al focus out :

```
$('form .user-box input').on('focusout', (event) => {
    new Validator(paramValidator, 'focusout', event.target);
    })
```

Es funzione che valida una form al submit :

```
function checkvalidation () {
     "use strict";

  // Fetch all the forms we want to apply custom Bootstrap validation styles to
  var forms = document.querySelectorAll(".needs-validation");
  // Loop over them and prevent submission
  Array.prototype.slice.call(forms).forEach(function (form) {
     form.addEventListener(
        "submit",
        function (event) {
          console.log(
              "####################submit################################"
          );
          let validation = new Validator(paramValidator, 'submit', event);

          console.log('client validation', validation.isValidForm);

          if (!validation.isValidForm || !form.checkValidity()) {
              event.preventDefault();
              event.stopPropagation();
              // status_validation = false;
              console.log("°°°°°°°° validazioni NON passate °°°°°°°°");
          } else {
              console.log(
                  "°°°°°°°° Validazioni passate °°°°°°°°"
              );
          }

          form.classList.add("was-validated");

      },
```

```
        false
              );
          });
      }
```