Form

Un formulaire est constitué des éléments `input`, `radio`, `select`, `checkbox`, etc. Il sert principalement à collecter, vérifier et soumettre des données.

Formulaire de base

Il peut contenir toutes sortes de champs tels que `input`, `select`, `radio` et `checkbox`.

:::demo Dans chaque composant `form`, il vous faudra utiliser la balise `form-item` pour servir de conteneur à chaque champ.

```
<el-form ref="form" :model="form" label-width="120px">
  <el-form-item label="Activity name">
    <el-input v-model="form.name"></el-input>
  </el-form-item>
  <el-form-item label="Activity zone">
    <el-select v-model="form.region" placeholder="please select your zone">
      <el-option label="Zone one" value="shanghai"></el-option>
      <el-option label="Zone two" value="beijing"></el-option>
    </el-select>
  </el-form-item>
  <el-form-item label="Activity time">
    <el-col :span="11">
      <el-date-picker type="date" placeholder="Choisissez une date" v-model="form.date1" sty
    </el-col>
    <el-col class="line" :span="2">-</el-col>
    <el-col :span="11">
      <el-time-picker placeholder="Pick a time" v-model="form.date2" style="width: 100%;"></
    </el-col>
  </el-form-item>
  <el-form-item label="Instant delivery">
    <el-switch v-model="form.delivery"></el-switch>
  </el-form-item>
  <el-form-item label="Activity type">
    <el-checkbox-group v-model="form.type">
      <el-checkbox label="Online activities" name="type"></el-checkbox>
      <el-checkbox label="Promotion activities" name="type"></el-checkbox>
      <el-checkbox label="Offline activities" name="type"></el-checkbox>
      <el-checkbox label="Simple brand exposure" name="type"></el-checkbox>
    </el-checkbox-group>
  </el-form-item>
  <el-form-item label="Resources">
    <el-radio-group v-model="form.resource">
      <el-radio label="Sponsor"></el-radio>
      <el-radio label="Venue"></el-radio>
```

```
    </el-radio-group>
  </el-form-item>
  <el-form-item label="Activity form">
    <el-input type="textarea" v-model="form.desc"></el-input>
  </el-form-item>
  <el-form-item>
    <el-button type="primary" @click="onSubmit">Créer</el-button>
    <el-button>Annuler</el-button>
  </el-form-item>
</el-form>
<script>
  export default {
    data() {
      return {
        form: {
          name: '',
          region: '',
          date1: '',
          date2: '',
          delivery: false,
          type: [],
          resource: '',
          desc: ''
        }
      }
    },
    methods: {
      onSubmit() {
        console.log('submit!');
      }
    }
  }
</script>
```

:::

W3C stipule que > Lorsqu'il n'y a qu'un seul champ de type texte dans un formulaire, le navigateur devrait accepter la pression de la touche Entrée sur ce champ comme méthode de soumission du formulaire

Pour éviter ce comportement, vous pouvez ajouter `@submit.native.prevent` dans `<el-form>`. :::

Formulaire horizontal

Lorsque l'espace vertical est limité et que le formulaire est relativement simple, vous pouvez le placer sur une seule ligne.

:::demo Mettez l'attribut `inline` à `true` et le formulaire sera en une seul ligne.

```
<el-form :inline="true" :model="formInline" class="demo-form-inline">
  <el-form-item label="Approved by">
    <el-input v-model="formInline.user" placeholder="Approved by"></el-input>
  </el-form-item>
  <el-form-item label="Activity zone">
    <el-select v-model="formInline.region" placeholder="Activity zone">
      <el-option label="Zone one" value="shanghai"></el-option>
      <el-option label="Zone two" value="beijing"></el-option>
    </el-select>
  </el-form-item>
  <el-form-item>
    <el-button type="primary" @click="onSubmit">Query</el-button>
  </el-form-item>
</el-form>
<script>
  export default {
    data() {
      return {
        formInline: {
          user: '',
          region: ''
        }
      }
    },
    methods: {
      onSubmit() {
        console.log('submit!');
      }
    }
  }
</script>
```

Alignement

Suivant votre design, il y a différents moyens d'aligner vos labels.

:::demo L'attribut `label-position` permet de régler l'alignement, il peut être à `top` ou `left`. Quand il est à `top`, les labels sont placés au-dessus des champs.

```
<el-radio-group v-model="labelPosition" size="small">
  <el-radio-button label="left">Left</el-radio-button>
  <el-radio-button label="right">Right</el-radio-button>
  <el-radio-button label="top">Top</el-radio-button>
</el-radio-group>
<div style="margin: 20px;"></div>
```

```
<el-form :label-position="labelPosition" label-width="100px" :model="formLabelAlign">
  <el-form-item label="Name">
    <el-input v-model="formLabelAlign.name"></el-input>
  </el-form-item>
  <el-form-item label="Activity zone">
    <el-input v-model="formLabelAlign.region"></el-input>
  </el-form-item>
  <el-form-item label="Activity form">
    <el-input v-model="formLabelAlign.type"></el-input>
  </el-form-item>
</el-form>
<script>
  export default {
    data() {
      return {
        labelPosition: 'right',
        formLabelAlign: {
          name: '',
          region: '',
          type: ''
        }
      };
    }
  }
</script>
```

:::

Validation

Le composant Form vous permet d'effectuer des vérifications, afin de détecter et corriger les erreurs facilement.

:::demo Ajoutez l'attribut `rules` au composant `Form`, passez les règles de validation, et configurez l'attribut `prop` de `Form-Item` pour ajouter la clé de la règle correspondante au champ. Plus d'informations ici: async-validator.

```
<el-form :model="ruleForm" :rules="rules" ref="ruleForm" label-width="120px" class="demo-rul
  <el-form-item label="Activity name" prop="name">
    <el-input v-model="ruleForm.name"></el-input>
  </el-form-item>
  <el-form-item label="Activity zone" prop="region">
    <el-select v-model="ruleForm.region" placeholder="Activity zone">
      <el-option label="Zone one" value="shanghai"></el-option>
      <el-option label="Zone two" value="beijing"></el-option>
    </el-select>
  </el-form-item>
```

```
  <el-form-item label="Activity time" required>
    <el-col :span="11">
      <el-form-item prop="date1">
        <el-date-picker type="date" placeholder="Choisissez une date" v-model="ruleForm.date
      </el-form-item>
    </el-col>
    <el-col class="line" :span="2">-</el-col>
    <el-col :span="11">
      <el-form-item prop="date2">
        <el-time-picker placeholder="Pick a time" v-model="ruleForm.date2" style="width: 10
      </el-form-item>
    </el-col>
  </el-form-item>
  <el-form-item label="Instant delivery" prop="delivery">
    <el-switch v-model="ruleForm.delivery"></el-switch>
  </el-form-item>
  <el-form-item label="Activity type" prop="type">
    <el-checkbox-group v-model="ruleForm.type">
      <el-checkbox label="Online activities" name="type"></el-checkbox>
      <el-checkbox label="Promotion activities" name="type"></el-checkbox>
      <el-checkbox label="Offline activities" name="type"></el-checkbox>
      <el-checkbox label="Simple brand exposure" name="type"></el-checkbox>
    </el-checkbox-group>
  </el-form-item>
  <el-form-item label="Resources" prop="resource">
    <el-radio-group v-model="ruleForm.resource">
      <el-radio label="Sponsorship"></el-radio>
      <el-radio label="Venue"></el-radio>
    </el-radio-group>
  </el-form-item>
  <el-form-item label="Activity form" prop="desc">
    <el-input type="textarea" v-model="ruleForm.desc"></el-input>
  </el-form-item>
  <el-form-item>
    <el-button type="primary" @click="submitForm('ruleForm')">Créer</el-button>
    <el-button @click="resetForm('ruleForm')">Réinitialiser</el-button>
  </el-form-item>
</el-form>
<script>
  export default {
    data() {
      return {
        ruleForm: {
          name: '',
          region: '',
          date1: '',
```

```
      date2: '',
      delivery: false,
      type: [],
      resource: '',
      desc: ''
    },
    rules: {
      name: [
        { required: true, message: 'Please input Activity name', trigger: 'blur' },
        { min: 3, max: 5, message: 'Length should be 3 to 5', trigger: 'blur' }
      ],
      region: [
        { required: true, message: 'Please select Activity zone', trigger: 'change' }
      ],
      date1: [
        { type: 'date', required: true, message: 'Please pick a date', trigger: 'change'
      ],
      date2: [
        { type: 'date', required: true, message: 'Please pick a time', trigger: 'change'
      ],
      type: [
        { type: 'array', required: true, message: 'Please select at least one activity t
      ],
      resource: [
        { required: true, message: 'Please select activity resource', trigger: 'change'
      ],
      desc: [
        { required: true, message: 'Please input activity form', trigger: 'blur' }
      ]
    }
  };
},
methods: {
  submitForm(formName) {
    this.$refs[formName].validate((valid) => {
      if (valid) {
        alert('submit!');
      } else {
        console.log('error submit!!');
        return false;
      }
    });
  },
  resetForm(formName) {
    this.$refs[formName].resetFields();
  }
```

```
    }
  }
</script>
```

:::

Validations personnalisées

Cet exemple montre comment vous pouvez personnaliser vos règles de validation pour effectuer une identification à deux facteurs.

:::demo Ici, nous utilisons `status-icon` pour afficher le résultat de la validation sous forme d'icône.

```
<el-form :model="ruleForm" status-icon :rules="rules" ref="ruleForm" label-width="120px" cla
  <el-form-item label="Password" prop="pass">
    <el-input type="password" v-model="ruleForm.pass" autocomplete="off"></el-input>
  </el-form-item>
  <el-form-item label="Confirm" prop="checkPass">
    <el-input type="password" v-model="ruleForm.checkPass" autocomplete="off"></el-input>
  </el-form-item>
  <el-form-item label="Age" prop="age">
    <el-input v-model.number="ruleForm.age"></el-input>
  </el-form-item>
  <el-form-item>
    <el-button type="primary" @click="submitForm('ruleForm')">Submit</el-button>
    <el-button @click="resetForm('ruleForm')">Réinitialiser</el-button>
  </el-form-item>
</el-form>
<script>
  export default {
    data() {
      var checkAge = (rule, value, callback) => {
        if (!value) {
          return callback(new Error('Veuillez entrer l\'âge'));
        }
        setTimeout(() => {
          if (!Number.isInteger(value)) {
            callback(new Error('Veuillez entrer des chiffres'));
          } else {
            if (value < 18) {
              callback(new Error('L\'âge doit être supérieur à 18 ans'));
            } else {
              callback();
            }
          }
        }, 1000);
```

```
    };
    var validatePass = (rule, value, callback) => {
      if (value === '') {
        callback(new Error('Veuillez entrer le mot de passe'));
      } else {
        if (this.ruleForm.checkPass !== '') {
          this.$refs.ruleForm.validateField('checkPass');
        }
        callback();
      }
    };
    var validatePass2 = (rule, value, callback) => {
      if (value === '') {
        callback(new Error('Veuillez entrer à nouveau le mot de passe'));
      } else if (value !== this.ruleForm.pass) {
        callback(new Error('Les deux entrées ne correspondent pas!'));
      } else {
        callback();
      }
    };
    return {
      ruleForm: {
        pass: '',
        checkPass: '',
        age: ''
      },
      rules: {
        pass: [
          { validator: validatePass, trigger: 'blur' }
        ],
        checkPass: [
          { validator: validatePass2, trigger: 'blur' }
        ],
        age: [
          { validator: checkAge, trigger: 'blur' }
        ]
      }
    };
  },
  methods: {
    submitForm(formName) {
      this.$refs[formName].validate((valid) => {
        if (valid) {
          alert('submit!');
        } else {
          console.log('error submit!!');
```

```
            return false;
          }
        });
      },
      resetForm(formName) {
        this.$refs[formName].resetFields();
      }
    }
  }
</script>
```

:::

Les callback de validations personnalisées doivent être appelées. Un usage plus avancé se trouve ici: async-validator.

Ajouter ou supprimer des champs dynamiquement

:::demo En plus de pouvoir passer toutes les règles de validation en une seule fois au formulaire, vous pouvez aussi ajouter ou supprimer des règles sur un seul champ de manière dynamique.

```
<el-form :model="dynamicValidateForm" ref="dynamicValidateForm" label-width="120px" class="d
  <el-form-item
    prop="email"
    label="Email"
    :rules="[
      { required: true, message: 'Veuillez entrer l\'adresse e-mail', trigger: 'blur' },
      { type: 'email', message: 'Veuillez entrer une adresse e-mail valide', trigger: ['blur
    ]"
  >
    <el-input v-model="dynamicValidateForm.email"></el-input>
  </el-form-item>
  <el-form-item
    v-for="(domain, index) in dynamicValidateForm.domains"
    :label="'Domain' + index"
    :key="domain.key"
    :prop="'domains.' + index + '.value'"
    :rules="{
      required: true, message: 'domain ne peut pas être null', trigger: 'blur'
    }"
  >
    <el-input v-model="domain.value"></el-input><el-button @click.prevent="removeDomain(doma
  </el-form-item>
  <el-form-item>
    <el-button type="primary" @click="submitForm('dynamicValidateForm')">Soumettre</el-butto
    <el-button @click="addDomain">Nouveau domaine</el-button>
```

```
    <el-button @click="resetForm('dynamicValidateForm')">Réinitialiser</el-button>
  </el-form-item>
</el-form>
<script>
  export default {
    data() {
      return {
        dynamicValidateForm: {
          domains: [{
            key: 1,
            value: ''
          }],
          email: ''
        }
      };
    },
    methods: {
      submitForm(formName) {
        this.$refs[formName].validate((valid) => {
          if (valid) {
            alert('submit!');
          } else {
            console.log('error submit!!');
            return false;
          }
        });
      },
      resetForm(formName) {
        this.$refs[formName].resetFields();
      },
      removeDomain(item) {
        var index = this.dynamicValidateForm.domains.indexOf(item);
        if (index !== -1) {
          this.dynamicValidateForm.domains.splice(index, 1);
        }
      },
      addDomain() {
        this.dynamicValidateForm.domains.push({
          key: Date.now(),
          value: ''
        });
      }
    }
  }
</script>
```

:::

Validation des nombres

:::demo Pour valider les nombres correctement, il vous faudra ajouter le modificateur `.number` à l'attribut `v-model`. Il est utilisé par Vuejs pour transformer les valeurs en nombres .

```
<el-form :model="numberValidateForm" ref="numberValidateForm" label-width="100px" class="den
  <el-form-item
    label="age"
    prop="age"
    :rules="[
      { required: true, message: 'l\'âge est requis'},
      { type: 'number', message: 'l\'âge doit être un nombre'}
    ]"
  >
    <el-input type="age" v-model.number="numberValidateForm.age" autocomplete="off"></el-inp
  </el-form-item>
  <el-form-item>
    <el-button type="primary" @click="submitForm('numberValidateForm')">Soumettre</el-buttor
    <el-button @click="resetForm('numberValidateForm')">Réinitialiser</el-button>
  </el-form-item>
</el-form>
<script>
  export default {
    data() {
      return {
        numberValidateForm: {
          age: ''
        }
      };
    },
    methods: {
      submitForm(formName) {
        this.$refs[formName].validate((valid) => {
          if (valid) {
            alert('submit!');
          } else {
            console.log('error submit!!');
            return false;
          }
        });
      },
      resetForm(formName) {
        this.$refs[formName].resetFields();
```

```
      }
    }
  }
</script>
```

:::

Lorsqu'un `el-form-item` est imbriqué dans un autre `el-form-item`, la largeur de son label sera `0`. Utilisez `label-width` sur ce `el-form-item` si besoin.

Taille

Tout les composants d'un formulaire héritent leur attribut `size` de ce formulaire. Il est aussi possible de l'utiliser individuellement sur chaque FormItem.

:::demo Vous pouvez régler le `size` de chaque item si vous ne souhaitez pas qu'il hérite de son parent.

```
<el-form ref="form" :model="sizeForm" label-width="120px" size="mini">
  <el-form-item label="Activity name">
    <el-input v-model="sizeForm.name"></el-input>
  </el-form-item>
  <el-form-item label="Activity zone">
    <el-select v-model="sizeForm.region" placeholder="veuillez sélectionner votre zone">
      <el-option label="Zone one" value="shanghai"></el-option>
      <el-option label="Zone two" value="beijing"></el-option>
    </el-select>
  </el-form-item>
  <el-form-item label="Activity time">
    <el-col :span="11">
      <el-date-picker type="date" placeholder="Choisissez une date" v-model="sizeForm.date1"
    </el-col>
    <el-col class="line" :span="2">-</el-col>
    <el-col :span="11">
      <el-time-picker placeholder="Choisissez une heure" v-model="sizeForm.date2" style="wi
    </el-col>
  </el-form-item>
  <el-form-item label="Activity type">
    <el-checkbox-group v-model="sizeForm.type">
      <el-checkbox-button label="Online activities" name="type"></el-checkbox-button>
      <el-checkbox-button label="Promotion activities" name="type"></el-checkbox-button>
    </el-checkbox-group>
  </el-form-item>
  <el-form-item label="Resources">
    <el-radio-group v-model="sizeForm.resource" size="medium">
      <el-radio border label="Sponsor"></el-radio>
      <el-radio border label="Venue"></el-radio>
    </el-radio-group>
```

```
  </el-form-item>
  <el-form-item size="large">
    <el-button type="primary" @click="onSubmit">Créer</el-button>
    <el-button>Annuler</el-button>
  </el-form-item>
</el-form>

<script>
  export default {
    data() {
      return {
        sizeForm: {
          name: '',
          region: '',
          date1: '',
          date2: '',
          delivery: false,
          type: [],
          resource: '',
          desc: ''
        }
      };
    },
    methods: {
      onSubmit() {
        console.log('submit!');
      }
    }
  };
</script>
```

:::

Attributs de Form

Attribut	Description	Type	Valeurs acceptées	Défaut
model	Données du formulaire.	object	—	—
rules	Règles de validation du formulaire.	object	—	—
inline	Si le formulaire est horizontal.	boolean	—	false

Attribut	Description	Type	Valeurs acceptées	Défaut
label-position	Position des labels. Si 'left' ou 'right', `label-width` est aussi requis.	string	left / right / top	right
label-width	Largeur des labels, tout les enfants directs hériteront de cette valeur. La largeur `auto` est supportée.	string	—	—
label-suffix	Suffixe de labels.	string	—	—
hide-required-asterisk	Si les champs obligatoires doivent avoir une astérisque rouge (étoile) à coté de leurs labels.	boolean	—	false
show-message	Si le message d'erreur doit apparaître.	boolean	—	true
inline-message	Si le message d'erreur doit apparaître en ligne avec son champ.	boolean	—	false
status-icon	Si une icône indiquant le résultat de validation doit apparaître.	boolean	—	false

Attribut	Description	Type	Valeurs acceptées	Défaut
validate-on-rule-change	Si la validation doit se déclencher lorsque `rules` est modifié.	boolean	—	true
size	Contrôle la taille des champs du formulaire.	string	medium / small / mini	—
disabled	Si tout les champs du formulaire doivent être désactivés. Si `true`, il ne peut pas être modifié par l'attribut `disabled` des enfants.	boolean	—	false

Méthodes de Form

Méthode	Description	Paramètres
validate	Valide le formulaire. Prends une callback en paramètre. Après la validation, la callback est exécutée avec deux paramètres: un boolean indiquant si la validation est bonne, et un objet contenant tout les champs qui ont échoués. Retourne une promesse si aucune callback n'est passée.	Function(callback: Function(boolean, object))

Méthode	Description	Paramètres
validateField	Valide un ou plusieurs champs du formulaire.	Function(props: string | array, callback: Function(errorMessage: string))
resetFields	Efface tout les champs et les résultats de validation.	—
clearValidate	Efface les messages de validation de certains champs. Le paramètre est le nom du champ ou une liste des champs concernés. S'il est omis, tout les champs seront concernés.	Function(props: string | array)

Évènnements de Form

Nom	Description	Paramètres
validate	Se déclenche après la validation d'un champ.	Nom du champs qui a été validé, si la validation est bonne et le message d'erreur sinon.

Attributs de FormItem

Attribut	Description	Type	Valeurs acceptées	Défaut
prop	Une des clés de `model`. Utilisés par les méthodes validate et resetFields. Requis.	string	Clés du model passé à `form`.	
label	Le label.	string	—	—

Attribut	Description	Type	Valeurs acceptées	Défaut
label-width	Largeur du label, e.g. '50px'. La largeur `auto` est supportée.	string	—	—
required	Si le champ est requis ou non. Si omis, sera déterminé par les règles de validation.	boolean	—	false
rules	Règles de validation du formulaire.	object	—	—
error	Message d'erreur du champ. S'il est modifié, le champ l'affichera immédiatement.	string	—	—
show-message	Si le message d'erreur doit apparaître.	boolean	—	true
inline-message	Si le message d'erreur doit être en ligne avec le champ.	boolean	—	false
size	Contrôle la taille du FormItem.	string	medium / small / mini	-

Slot de Form-Item

Nom	Description
—	Contenu de Form Item.
label	Contenu du label.

Slot avec portée de Form-Item

Nom	Description
error	Contenu personnalisé pour les messages de validation. Le paramètre du scope est { error }.

Méthodes de Form-Item

Méthode	Description	Paramètres
resetField	Efface le champ et les résultats de validation.	—
clearValidate	Efface le status de validation du champ.	-