Form

Form consiste en `input`, `radio`, `select`, `checkbox`, etcétera. Con el formulario, usted puede recopilar, verificar y enviar datos.

Form básico

Incluye todo tipo de entradas, tales como `input`, `select`, `radio` y `checkbox`.

:::demo En cada componente `form`, necesita un campo `form-item` para que sea el contenedor del ítem.

```
<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="Pick a date" v-model="form.date1" style="widt
    </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">Create</el-button>
    <el-button>Cancel</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 reglamenta que

> Cuando sólo hay un campo de entrada de texto de una sola línea en un formulario, el agente usuario debe aceptar Enter en ese campo como una solicitud para enviar el formulario.

Para prevenir este comportamiento, puede agregar `@submit.native.prevent` on `<el-form>`.

Formulario inline

Cuando el espacio vertical es limitado y la forma es relativamente simple, puede ponerlo en una unica línea.

:::demo Establezca el atributo `inline` como `true` y el formulario sera inline.

```
<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>
```

:::

Alineamiento

Dependiendo de su diseño, hay varias maneras diferentes de alinear el elemento de la etiqueta.

:::demo El atributo `label-position` decide cómo se alinean las etiquetas, puede estar `top` o `left`. Cuando se establece en `top`, las etiquetas se colocarán en la parte superior del campo de formulario.

```
<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>
```

:::

Validación

El componente `form` le permite verificar sus datos, ayudándole a encontrar y corregir errores.

:::demo Sólo tiene que añadir el atributo `rules` en el componente `Form`, pasar las reglas de validación y establecer el atributo `prop` para `Form-Item` como una clave específica que necesita ser validada. Ver más información en 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="Pick a date" v-model="ruleForm.date1" style
      </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: 100
      </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')">Create</el-button>
    <el-button @click="resetForm('ruleForm')">Reset</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>
```

:::

Reglas personalizadas de validación

Este ejemplo muestra cómo personalizar sus propias reglas de validación para finalizar una verificación de contraseña de dos pasos.

:::demo Aquí utilizamos el `status-icon` para reflejar el resultado de la validación como un icono.

```
<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')">Reset</el-button>
  </el-form-item>
</el-form>
<script>
  export default {
    data() {
      var checkAge = (rule, value, callback) => {
        if (!value) {
          return callback(new Error('Please input the age'));
        }
        setTimeout(() => {
          if (!Number.isInteger(value)) {
            callback(new Error('Please input digits'));
          } else {
            if (value < 18) {
              callback(new Error('Age must be greater than 18'));
            } else {
              callback();
```

```
        }
      }
    }, 1000);
  };
  var validatePass = (rule, value, callback) => {
    if (value === '') {
      callback(new Error('Please input the password'));
    } else {
      if (this.ruleForm.checkPass !== '') {
        this.$refs.ruleForm.validateField('checkPass');
      }
      callback();
    }
  };
  var validatePass2 = (rule, value, callback) => {
    if (value === '') {
      callback(new Error('Please input the password again'));
    } else if (value !== this.ruleForm.pass) {
      callback(new Error('Two inputs don\'t match!'));
    } else {
      callback();
    }
  };
  return {
    ruleForm: {
      pass: '',
      checkPass: '',
      age: ''
    },
    rules: {
      pass: [
        { validator: validatePass, trigger: 'blur' }
      ],
      checkPass: [
        { validator: validatePass2, trigger: 'blur' }
      ],
      age: [
        { validator: checkAge, trigger: 'blur' }
      ]
    }
  };
},
methods: {
  submitForm(formName) {
    this.$refs[formName].validate((valid) => {
      if (valid) {
```

```
            alert('submit!');
          } else {
            console.log('error submit!!');
            return false;
          }
        });
      },
      resetForm(formName) {
        this.$refs[formName].resetFields();
      }
    }
  }
</script>
```

:::

Se debe llamar a la función de validación de llamada de retorno personalizada. Ver uso más avanzado en async-validator.

Eliminar o agregar validaciones dinámicamente

:::demo Además de pasar todas las reglas de validación al mismo tiempo en el componente `form`, también puede pasar las reglas de validación o borrar reglas en un único campo de formulario de forma dinámica.

```
<el-form :model="dynamicValidateForm" ref="dynamicValidateForm" label-width="120px" class="d
  <el-form-item
    prop="email"
    label="Email"
    :rules="[
      { required: true, message: 'Please input email address', trigger: 'blur' },
      { type: 'email', message: 'Please input correct email address', trigger: ['blur', 'cha
    ]"
  >
    <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 can not be 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')">Submit</el-button>
    <el-button @click="addDomain">New domain</el-button>
    <el-button @click="resetForm('dynamicValidateForm')">Reset</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>
```

:::

Validación numérica

:::demo La validación numérica necesita un modificador `.number` añadido en el enlace `v-model` de entrada, sirve para transformar el valor de la cadena al número proporcionado por Vuejs.

```
<el-form :model="numberValidateForm" ref="numberValidateForm" label-width="100px" class="dem
  <el-form-item
    label="age"
    prop="age"
    :rules="[
      { required: true, message: 'age is required'},
      { type: 'number', message: 'age must be a number'}
    ]"
  >
    <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')">Submit</el-button>
    <el-button @click="resetForm('numberValidateForm')">Reset</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>
```

:::

Cuando un `el-form-item` está anidado en otro `el-form-item`, su ancho de etiqueta será 0. Si es necesario, puede establecer el ancho de etiqueta en ese `el-form-item`.

Tamaño del control

Todos los componentes de un formulario heredan su atributo `size`. De manera similar, FormItem también tiene un atributo `size`.

:::demo Aún así, puede ajustar el `size` de cada componente si no desea que herede su tamaño de From o FormItem.

```
<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="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="Pick a date" v-model="sizeForm.date1" style="
    </el-col>
    <el-col class="line" :span="2">-</el-col>
    <el-col :span="11">
      <el-time-picker placeholder="Pick a time" v-model="sizeForm.date2" style="width: 100%;
    </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">Create</el-button>
    <el-button>Cancel</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>
```

:::

Form Atributos

Atributo	Descripción	Tipo	Valores aceptados	Por defecto
model	Datos del componente	object	—	—
rules	Reglas de validación	object	—	—
inline	Si el form es inline	boolean	—	false
label-position	Posición de la etiqueta	string	left / right / top	right

Atributo	Descripción	Tipo	Valores aceptados	Por defecto
label-width	anchura de la etiqueta, por ejemplo, “50px”. Todos sus elementos de formulario hijo directo heredarán este valor. El valor `auto` está soportado.	string	—	—
label-suffix	sufijo de la etiqueta	string	—	—
hide-required-asterisk	si los campos obligatorios deben tener un asterisco rojo (estrella) al lado de sus etiquetas	boolean	—	false
show-message	si mostrar o no el mensaje de error	boolean	—	true
inline-message	si desea visualizar el mensaje de error inline con la posición del form item	boolean	—	false
status-icon	si desea visualizar un icono que indique el resultado de la validación	boolean	—	false
validate-on-rule-change	si se dispara la validación cuando el prop `rules` cambia	boolean	—	true
size	el tamaño de los componentes en este form	string	medium / small / mini	—
disabled	si se desactivan todos los componentes del formulario. Si esta en `true` no puede ser cambiado por el prop `disabled` individual de los componentes.	boolean	—	false

Form Métodos

Método	Descripción	Parámetros
validate	el método para validar todo el formulario. Recibe una llamada como parámetro. Después de la validación, la llamada de retorno se ejecutará con dos parámetros: un booleano que indica si la validación ha pasado, y un objeto que contiene todos los campos que fallaron en la validación. Devuelve una promesa si se omite el return	Function(callback: Function(boolean, object))
validateFields	validar uno o varios elementos de formulario	Function(props: string | array, callback: Function(errorMessage: string))
resetFields	restablece todos los campos y elimina el resultado de validación	—
clearValidate	borra el mensaje de validación para determinados campos. El parámetro es un prop name o un array de props names de los items del formulario cuyos mensajes de validación se eliminarán. Si se omiten, se borrarán todos los mensajes de validación de los campos.	Function(props: string | array)

Eventos Form

Nombre	Descripción	Parámetros
validate	se dispara después de validar un ítem del formulario	la propiedad (`prop name`) nombre del ítem del form que se esta validando, si la validación paso o no, y el mensaje de error si existe.

Form-Item Atributos

Atributo	Descripción	Tipo	Valores aceptados	Por defecto
prop	un clave del modelo. En el uso del método `validate` y `resetFields`, el atributo es obligatorio.	string	Clave del modelo que se ha pasado a `form`	
label	etiqueta	string	—	—
label-width	ancho de la etiqueta, Ejemplo: '50px'. El valor `auto` esta soportado	string	—	—
required	si el campo es obligatorio o no, estará determinado por las reglas de validación si se omite.	boolean	—	false
rules	reglas de validación del form	object	—	—
error	mensaje de error de campo, establezca su valor y el campo validará el error y mostrará este mensaje inmediatamente.	string	—	—
show-message	si mostrar o no el mensaje de error	boolean	—	true
inline-message	mensaje de validación estilo inline	boolean	—	false
size	Tamaño de los componentes en este form item	string	medium / small / mini	-

Form-Item Slot

Nombre	Descripción
—	contenido del Form Item
label	contenido de la etiqueta

Form-Item Scoped Slot

Name	Description
error	Contenido personalizado para mostrar el mensaje de validación. El parámetro del scope es { error }

Form-Item Método

Método	Descripción	Parámetros
resetField	restablecer campo actual y eliminar resultado de validación	—
clearValidate	elimina el estado de la validación de un campo	-