Form

Form consists of `input` , `radio` , `select` , `checkbox` and so on. With form, you can collect, verify and submit data.

Basic form

It includes all kinds of input items, such as `input` , `select` , `radio` and `checkbox` .

:::demo In each `form` component, you need a `form-item` field to be the container of your input item.

```
<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="width: 100%;"></el-date-picker>
    </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-time-picker>
    </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>
```

:::

:::tip W3C regulates that

> *When there is only one single-line text input field in a form, the user agent should accept Enter in that field as a request to submit the form.*

To prevent this behavior, you can add `@submit.native.prevent` on `<el-form>`. :::

Inline form

When the vertical space is limited and the form is relatively simple, you can put it in one line.

:::demo Set the `inline` attribute to `true` and the form will be 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>
```

:::

Alignment

Depending on your design, there are several different ways to align your label element.

:::demo The `label-position` attribute decides how labels align, it can be `top` or `left`. When set to `top`, labels will be placed at the top of the form field.

```
<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

Form component allows you to verify your data, helping you find and correct errors.

:::demo Just add the `rules` attribute for `Form` component, pass validation rules, and set `prop` attribute for `Form-Item` as a specific key that needs to be validated. See more information at async-validator.

```
<el-form :model="ruleForm" :rules="rules" ref="ruleForm" label-width="120px"
class="demo-ruleForm">
  <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="width: 100%;"></el-date-picker>
      </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-time-picker>
      </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 type', trigger: 'change' }
          ],
          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>
```

:::

Custom validation rules

This example shows how to customize your own validation rules to finish a two-factor password verification.

:::demo Here we use `status-icon` to reflect validation result as an icon.

```
<el-form :model="ruleForm" status-icon :rules="rules" ref="ruleForm" label-
width="120px" class="demo-ruleForm">
  <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>
```

:::

:::tip Custom validate callback function must be called. See more advanced usage at async-validator. :::

Delete or add form items dynamically

:::demo In addition to passing all validation rules at once on the form component, you can also pass the validation rules or delete rules on a single form field dynamically.

```
<el-form :model="dynamicValidateForm" ref="dynamicValidateForm" label-width="120px"
class="demo-dynamic">
  <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', 'change'] }
    ]"
  >
    <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(domain)">Delete</el-button>
  </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>
```

:::

Number Validate

:::demo Number Validate need a `.number` modifier added on the input `v-model` binding，it's used to transform the string value to the number which is provided by Vuejs.

```
<el-form :model="numberValidateForm" ref="numberValidateForm" label-width="100px"
class="demo-ruleForm">
  <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-input>
  </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>
```

:::

:::tip When an `el-form-item` is nested in another `el-form-item`, its label width will be `0`. You can set `label-width` on that `el-form-item` if needed. :::

Size control

All components in a Form inherit their `size` attribute from that Form. Similarly, FormItem also has a `size` attribute.

:::demo Still you can fine tune each component's `size` if you don't want that component to inherit its size from From or FormIten.

```
<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="width: 100%;"></el-date-picker>
    </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-time-picker>
    </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 Attributes

Attribute	Description	Type	Accepted Values	Default
model	data of form component	object	—	—
rules	validation rules of form	object	—	—
inline	whether the form is inline	boolean	—	false
label-position	position of label. If set to 'left' or 'right', `label-width` prop is also required	string	left / right / top	right

label-width	width of label, e.g. '50px'. All its direct child form items will inherit this value. Width `auto` is supported.	string	—	—
label-suffix	suffix of the label	string	—	—
hide-required-asterisk	whether to hide a red asterisk (star) next to the required field label.	boolean	—	false
show-message	whether to show the error message	boolean	—	true
inline-message	whether to display the error message inline with the form item	boolean	—	false
status-icon	whether to display an icon indicating the validation result	boolean	—	false
validate-on-rule-change	whether to trigger validation when the `rules` prop is changed	boolean	—	true
size	control the size of components in this form	string	medium / small / mini	—
disabled	whether to disabled all components in this form. If set to true, it cannot be overridden by its inner components' `disabled` prop	boolean	—	false

Form Methods

Method	Description	Parameters
validate	validate the whole form. Takes a callback as a param. After validation, the callback will be executed with two params: a boolean indicating if the validation has passed, and an object containing all fields that fail the validation. Returns a promise if callback is omitted	Function(callback: Function(boolean, object))
validateField	validate one or several form items	Function(props: string | array, callback: Function(errorMessage: string))
resetFields	reset all the fields and remove validation result	—
clearValidate	clear validation message for certain fields. The parameter is prop name or an array of prop names of the form items whose validation messages will be removed. When omitted, all fields' validation messages will be cleared	Function(props: string | array)

Form Events

Event Name	Description	Parameters
validate	triggers after a form item is validated	prop name of the form item being validated, whether validation is passed and the error message if not

Form-Item Attributes

Attribute	Description	Type	Accepted Values	Default
prop	a key of `model`. In the use of validate and resetFields method, the attribute is required	string	keys of model that passed to `form`	
label	label	string	—	—
label-width	width of label, e.g. '50px'. Width `auto` is supported.	string	—	—
required	whether the field is required or not, will be determined by validation rules if omitted	boolean	—	false
rules	validation rules of form	object	—	—
error	field error message, set its value and the field will validate error and show this message immediately	string	—	—
show-message	whether to show the error message	boolean	—	true
inline-message	inline style validate message	boolean	—	false
size	control the size of components in this form-item	string	medium / small / mini	-

Form-Item Slot

Name	Description
—	content of Form Item
label	content of label

Form-Item Scoped Slot

Name	Description
error	Custom content to display validation message. The scope parameter is { error }

Form-Item Methods

Method	Description	Parameters

resetField	reset current field and remove validation result	—
clearValidate	remove validation status of the field	-