Input

Ingresa datos usando el ratón o teclado.

:::warning Input es un componente controlado, **siempre muestra el valor de enlace Vue**.

Bajo circunstancias normales, el evento "input" debe ser manejado. Su handler debe actualizar el valor de enlace del componente (o usar `v-model`). De lo contrario, el valor del cuadro de entrada no cambiará.

No admite modificadores `v-model`. :::

Uso básico

:::demo

```
<el-input placeholder="Please input" v-model="input"></el-input>

<script>
export default {
  data() {
    return {
      input: ''
    }
  }
}
</script>
```

:::

Disabled

:::demo Deshabilite el Input con el atributo `disabled`.

```
<el-input
  placeholder="Please input"
  v-model="input"
  :disabled="true">
</el-input>

<script>
export default {
  data() {
    return {
      input: ''
    }
  }
}
</script>
```

:::

Limpiable

:::demo Marque que el input puede ser limpiable con el atributo `clearable` .

```
<el-input
  placeholder="Please input"
  v-model="input"
  clearable>
</el-input>

<script>
export default {
  data() {
    return {
      input: ''
    }
  }
}
</script>
```

:::

Password box

:::demo Haga un input de contraseña conmutable con el atributo `show-password` .

```
<el-input placeholder="Please input password" v-model="input" show-password></el-
input>

<script>
  export default {
    data() {
      return {
        input: ''
      }
    }
  }
</script>
```

:::

Input con icono

Añada un icono para indicar el tipo de Input.

:::demo Para añadir iconos en el Input, puede utilizar los atributos `prefix-icon` y `suffix-icon` . Además, los slots con nombre `prefix` y `suffix` también funcionan.

```
<div class="demo-input-suffix">
  <span class="demo-input-label">Using attributes</span>
  <el-input
    placeholder="Pick a date"
    suffix-icon="el-icon-date"
```

```
    v-model="input1">
  </el-input>
  <el-input
    placeholder="Type something"
    prefix-icon="el-icon-search"
    v-model="input2">
  </el-input>
</div>
<div class="demo-input-suffix">
  <span class="demo-input-label">Using slots</span>
  <el-input
    placeholder="Pick a date"
    v-model="input3">
    <i slot="suffix" class="el-input__icon el-icon-date"></i>
  </el-input>
  <el-input
    placeholder="Type something"
    v-model="input4">
    <i slot="prefix" class="el-input__icon el-icon-search"></i>
  </el-input>
</div>

<style>
  .demo-input-label {
    display: inline-block;
    width: 130px;
  }
</style>

<script>
export default {
  data() {
    return {
      input1: '',
      input2: '',
      input3: '',
      input4: ''
    }
  }
}
</script>
```

:::

Textarea

Re dimensiona para introducir varias líneas de información de texto. Agregue el atributo `type="textarea"` para cambiar el `input` al tipo nativo `textarea`.

:::demo Controle la altura ajustando el prop `rows`.

```
<el-input
  type="textarea"
  :rows="2"
  placeholder="Please input"
  v-model="textarea">
</el-input>

<script>
export default {
  data() {
    return {
      textarea: ''
    }
  }
}
</script>
```

:::

Textarea tamaño automático

El ajuste del prop `autosize` en el tipo de Input textarea hace que la altura se ajuste automáticamente en función del contenido. Se puede proporcionar opciones en un objeto para auto dimensionar y especificar el número mínimo y máximo de líneas que el textarea puede ajustar automáticamente.

:::demo

```
<el-input
  type="textarea"
  autosize
  placeholder="Please input"
  v-model="textarea1">
</el-input>
<div style="margin: 20px 0;"></div>
<el-input
  type="textarea"
  :autosize="{ minRows: 2, maxRows: 4}"
  placeholder="Please input"
  v-model="textarea2">
</el-input>

<script>
export default {
  data() {
    return {
      textarea1: '',
      textarea2: ''
    }
  }
}
</script>
```

:::

Mezclando elementos con input

Añade un elemento antes o después del input, generalmente una etiqueta o un botón.

:::demo Utilice el `slot` para seleccionar si el elemento se colocara antes (prepend) o después (append) del Input.

```
<div>
  <el-input placeholder="Please input" v-model="input1">
    <template slot="prepend">Http://</template>
  </el-input>
</div>
<div style="margin-top: 15px;">
  <el-input placeholder="Please input" v-model="input2">
    <template slot="append">.com</template>
  </el-input>
</div>
<div style="margin-top: 15px;">
  <el-input placeholder="Please input" v-model="input3" class="input-with-select">
    <el-select v-model="select" slot="prepend" placeholder="Select">
      <el-option label="Restaurant" value="1"></el-option>
      <el-option label="Order No." value="2"></el-option>
      <el-option label="Tel" value="3"></el-option>
    </el-select>
    <el-button slot="append" icon="el-icon-search"></el-button>
  </el-input>
</div>

<style>
  .el-select .el-input {
    width: 110px;
  }
  .input-with-select .el-input-group__prepend {
    background-color: #fff;
  }
</style>
<script>
export default {
  data() {
    return {
      input1: '',
      input2: '',
      input3: '',
      select: ''
    }
  }
}
</script>
```

:::

Tamaño

:::demo Añada el atributo `size` para cambiar el tamaño del Input. Además del tamaño predeterminado, hay otras tres opciones: `large`, `small` y `mini`.

```
<div class="demo-input-size">
  <el-input
    placeholder="Please Input"
    v-model="input1">
  </el-input>
  <el-input
    size="medium"
    placeholder="Please Input"
    v-model="input2">
  </el-input>
  <el-input
    size="small"
    placeholder="Please Input"
    v-model="input3">
  </el-input>
  <el-input
    size="mini"
    placeholder="Please Input"
    v-model="input4">
  </el-input>
</div>

<script>
export default {
  data() {
    return {
      input1: '',
      input2: '',
      input3: '',
      input4: ''
    }
  }
}
</script>
```

:::

Autocompletado

Puede obtener algunas sugerencias basadas en la entrada actual.

:::demo El componente Autocomplete proporciona sugerencias de entrada. El atributo `fetch-suggestions` es un método que devuelve la entrada sugerida. En este ejemplo, `querySearch(queryString, cb)` devuelve las sugerencias al componente mediante `cb(data)` cuando están listas.

```
<el-row class="demo-autocomplete">
  <el-col :span="12">
    <div class="sub-title">list suggestions when activated</div>
    <el-autocomplete
      class="inline-input"
      v-model="state1"
      :fetch-suggestions="querySearch"
      placeholder="Please Input"
      @select="handleSelect"
    ></el-autocomplete>
  </el-col>
  <el-col :span="12">
    <div class="sub-title">list suggestions on input</div>
    <el-autocomplete
      class="inline-input"
      v-model="state2"
      :fetch-suggestions="querySearch"
      placeholder="Please Input"
      :trigger-on-focus="false"
      @select="handleSelect"
    ></el-autocomplete>
  </el-col>
</el-row>
<script>
  export default {
    data() {
      return {
        links: [],
        state1: '',
        state2: ''
      };
    },
    methods: {
      querySearch(queryString, cb) {
        var links = this.links;
        var results = queryString ? links.filter(this.createFilter(queryString)) :
links;
        // call callback function to return suggestions
        cb(results);
      },
      createFilter(queryString) {
        return (link) => {
          return (link.value.toLowerCase().indexOf(queryString.toLowerCase()) ===
0);
        };
      },
      loadAll() {
        return [
          { "value": "vue", "link": "https://github.com/vuejs/vue" },
          { "value": "element", "link": "https://github.com/ElemeFE/element" },
          { "value": "cooking", "link": "https://github.com/ElemeFE/cooking" },
```

```
          { "value": "mint-ui", "link": "https://github.com/ElemeFE/mint-ui" },
          { "value": "vuex", "link": "https://github.com/vuejs/vuex" },
          { "value": "vue-router", "link": "https://github.com/vuejs/vue-router" },
          { "value": "babel", "link": "https://github.com/babel/babel" }
        ];
      },
      handleSelect(item) {
        console.log(item);
      }
    },
    mounted() {
      this.links = this.loadAll();
    }
  }
</script>
```

:::

Template personalizado

Personalice cómo se muestran las sugerencias.

:::demo Utilice `scoped slot` para personalizar los elementos de sugerencias. En el scope, puede acceder al objeto de sugerencia mediante la clave `item`.

```
<el-autocomplete
  popper-class="my-autocomplete"
  v-model="state"
  :fetch-suggestions="querySearch"
  placeholder="Please input"
  @select="handleSelect">
  <i
    class="el-icon-edit el-input__icon"
    slot="suffix"
    @click="handleIconClick">
  </i>
  <template slot-scope="{ item }">
    <div class="value">{{ item.value }}</div>
    <span class="link">{{ item.link }}</span>
  </template>
</el-autocomplete>

<style>
  .my-autocomplete {
    li {
      line-height: normal;
      padding: 7px;

      .value {
        text-overflow: ellipsis;
        overflow: hidden;
      }
```

```
      .link {
        font-size: 12px;
        color: #b4b4b4;
      }
    }
  }
</style>

<script>
  export default {
    data() {
      return {
        links: [],
        state: ''
      };
    },
    methods: {
      querySearch(queryString, cb) {
        var links = this.links;
        var results = queryString ? links.filter(this.createFilter(queryString)) :
links;
        // call callback function to return suggestion objects
        cb(results);
      },
      createFilter(queryString) {
        return (link) => {
          return (link.value.toLowerCase().indexOf(queryString.toLowerCase()) ===
0);
        };
      },
      loadAll() {
        return [
          { "value": "vue", "link": "https://github.com/vuejs/vue" },
          { "value": "element", "link": "https://github.com/ElemeFE/element" },
          { "value": "cooking", "link": "https://github.com/ElemeFE/cooking" },
          { "value": "mint-ui", "link": "https://github.com/ElemeFE/mint-ui" },
          { "value": "vuex", "link": "https://github.com/vuejs/vuex" },
          { "value": "vue-router", "link": "https://github.com/vuejs/vue-router" },
          { "value": "babel", "link": "https://github.com/babel/babel" }
         ];
      },
      handleSelect(item) {
        console.log(item);
      },
      handleIconClick(ev) {
        console.log(ev);
      }
    },
    mounted() {
      this.links = this.loadAll();
    }
```

```
  }
</script>
```

:::

Búsqueda remota

Búsqueda de datos desde el servidor.

:::demo

```
<el-autocomplete
  v-model="state"
  :fetch-suggestions="querySearchAsync"
  placeholder="Please input"
  @select="handleSelect"
></el-autocomplete>
<script>
  export default {
    data() {
      return {
        links: [],
        state: '',
        timeout:  null
      };
    },
    methods: {
      loadAll() {
        return [
          { "value": "vue", "link": "https://github.com/vuejs/vue" },
          { "value": "element", "link": "https://github.com/ElemeFE/element" },
          { "value": "cooking", "link": "https://github.com/ElemeFE/cooking" },
          { "value": "mint-ui", "link": "https://github.com/ElemeFE/mint-ui" },
          { "value": "vuex", "link": "https://github.com/vuejs/vuex" },
          { "value": "vue-router", "link": "https://github.com/vuejs/vue-router" },
          { "value": "babel", "link": "https://github.com/babel/babel" }
         ];
      },
      querySearchAsync(queryString, cb) {
        var links = this.links;
        var results = queryString ? links.filter(this.createFilter(queryString)) :
links;

        clearTimeout(this.timeout);
        this.timeout = setTimeout(() => {
          cb(results);
        }, 3000 * Math.random());
      },
      createFilter(queryString) {
        return (link) => {
          return (link.value.toLowerCase().indexOf(queryString.toLowerCase()) ===
0);
```

```
        };
      },
      handleSelect(item) {
        console.log(item);
      }
    },
    mounted() {
      this.links = this.loadAll();
    }
  };
</script>
```

:::

Limitar el tamaño

:::demo `maxlength` y `minlength` son atributos de la entrada nativa, declaran un límite en el número de caracteres que un usuario puede introducir. La configuración de la pro `maxlength` para un tipo de entrada de texto o de área de texto puede limitar la longitud del valor de entrada y le permite mostrar el recuento de palabras al establecer `show-word-limit` a `true` al mismo tiempo.

```
<el-input
  type="text"
  placeholder="Please input"
  v-model="text"
  maxlength="10"
  show-word-limit
>
</el-input>
<div style="margin: 20px 0;"></div>
<el-input
  type="textarea"
  placeholder="Please input"
  v-model="textarea"
  maxlength="30"
  show-word-limit
>
</el-input>

<script>
export default {
  data() {
    return {
      text: '',
      textarea: ''
    }
  }
}
</script>
```

:::

Input atributos

Atributo	Descripción	Tipo	Valores aceptados	Por defecto
type	tipo de input	string	text, textarea y otros tipos de entrada nativos	text
value / v-model	valor enlazado	boolean / string / number	—	—
maxlength	igual que `maxlength` en el input nativo	number	—	—
minlength	igual que `minlength` en el input nativo	number	—	—
show-word-limit	Si se muestra el contador de palabras, solamente funciona con los tipos `text` o `textarea`	boolean	—	false
placeholder	placeholder del Input	string	—	—
clearable	si debe mostrar el botón de limpieza	boolean	—	false
show-password	si debe mostrar la posibilidad de conmutación de password input	boolean	—	false
disabled	si esta deshabilitado	boolean	—	false
size	tamaño del input, esto no funciona cuando `type` no es textarea	string	medium / small / mini	—
prefix-icon	clase del icono de prefijo	string	—	—
suffix-icon	clase del icono de sufijo	string	—	—
rows	número de filas, sólo funciona cuando `type` es `textarea`.	number	—	2
autosize	si textarea tiene una altura adaptativa, sólo funciona cuando el`type` es `textarea`. Puede aceptar un objeto, p. ej. { minRows: 2, maxRows: 6 }	boolean / object	—	false
autocomplete	igual que `autocomplete` en el input nativo	string	on/off	off
auto-complete	@DEPRECATED en el próximo cambio mayor de versión	string	on/off	off
name	igual que `name` en el input nativo	string	—	—
readonly	igual que `readonly` en el input nativo	boolean	—	false
max	igual que `max` en el input nativo	—	—	—

min	igual que `min` en el input nativo	—	—	—
step	igual que `step` en el input nativo	—	—	—
resize	control para el dimensionamiento	string	none, both, horizontal, vertical	—
autofocus	igual que `autofocus` en el input nativo	boolean	—	false
form	igual que `form` en el input nativo	string	—	—
label	texto de la etiqueta	string	—	—
tabindex	orden de tabulación para el Input	string	-	-

Input slots

Nombre	Descripción
prefix	contenido como prefijo del input
suffix	contenido como sufijo del input
prepend	contenido antes del input
append	contenido a añadir después del input

Input eventos

Nombre	Descripción	Parámetros
blur	Se dispara cuando se pierde el foco	(event: Event)
focus	Se dispara cuando se obtiene el foco	(event: Event)
change	se activa cuando cambia el valor de entrada	(value: string | number)
change	se activa solo cuando el cuadro de entrada pierde el foco o el usuario presiona Enter	(value: string | number)
input	se activa cuando cambia el valor de entrada	(value: string | number)
clear	se dispara cuando la entrada es borrada por el botón generado por el atributo `clearable`.	—

Input Métodos

Método	Descripción	Parámetros
focus	coloca el foco en el elemento	—
blur	quita el foco del elemento	—
select	selecciona el texto del input	—

Autocomplete Atributos

Atributo	Descripción	Tipo	Opciones	Por defecto
placeholder	el placeholder del Autocomplete	string	—	—
disabled	si el Autocompete esta deshabilitado	boolean	—	false
value-key	nombre del campo del objeto de sugerencia del input para la visualización	string	—	value
icon	nombre del icono	string	—	—
value	valor enlazado	string	—	—
debounce	retardo al escribir, en milisegundos	number	—	300
placement	ubicación del menú emergente	string	top / top-start / top-end / bottom / bottom-start / bottom-end	bottom-start
fetch-suggestions	un método para obtener las sugerencias del input. Cuando las sugerencias estén listas, invocar `callback(data:[])` para devolverlas a Autocomplete	Function(queryString, callback)	—	—
popper-class	nombre personalizado de clase para el dropdown de autocomplete	string	—	—
trigger-on-focus	si se deben mostrar sugerencias cuando el input obtiene el foco	boolean	—	true
name	igual que `name` en el input nativo	string	—	—
select-when-unmatched	si se emite un evento `select` al pulsar enter cuando no hay coincidencia de Autocomplete	boolean	—	false
label	texto de la etiqueta	string	—	—
prefix-icon	prefix icon class	string	—	—
suffix-icon	suffix icon class	string	—	—
hide-loading	si se debe ocultar el icono de loading en la búsqueda remota	boolean	—	false
popper-append-to-body	si añadir el desplegable al cuerpo. Si la posición del menú desplegable es	boolean	-	true

	incorrecta, puede intentar establecer este prop a false			
validate-event	si se debe lanzar la validación de formulario	boolean	-	true
highlight-first-item	si se debe resaltar el primer elemento en las sugerencias de búsqueda remota de forma predeterminada	boolean	-	false

Autocomplete Slots

Nombre	Descripción
prefix	contenido como prefijo del input
suffix	contenido como sufijo del input
prepend	contenido antes del input
append	contenido a añadir después del input

Autocomplete Scoped Slot

Nombre	Descripción
—	Contenido personalizado para el input de sugerencias. El parámetro del scope es { ítem }

Autocomplete Eventos

Nombre	Descripción	Parámetros
select	se dispara cuando se hace clic a una sugerencia	sugerencia en la que se está haciendo clic
change	se activa cuando cambia el valor de entrada	(value: string | number)

Autocomplete Método

Método	Descripción	Parámetros
focus	coloca el foco en el elemento	—