Input

Le champ d'input de base.

:::warning Input est un composant contrôlé, il **affiche toujours la valeur liée de Vue**.

En règle générale, l'évènement `input` devrait être géré. Son handler devrait mettre à jour la valeur du composant (ou utilisez `v-model`). Dans le cas contraire, la valeur du champ ne sera pas modifiée.

Les modificateurs de `v-model` ne sont pas supportés. :::

Usage

:::demo

```
<el-input placeholder="Entrez quelque chose" v-model="input"></el-input>

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

:::

Désactivé

:::demo Désactivez l'input avec l'attribut `disabled`.

```
<el-input
  placeholder="Entrez quelque chose"
  v-model="input"
  :disabled="true">
</el-input>

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

:::

Effaçable

:::demo Rendez l'input effaçable avec l'attribut `clearable` .

```
<el-input
  placeholder="Entrez quelque chose"
  v-model="input"
  clearable>
</el-input>

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

:::

Champ de mot de passe

:::demo Créez un champ de mot de passe avec icône de visualisation grâce à l'attribut `show-password` .

```
<el-input placeholder="Entrez votre mot de passe" v-model="input" show-password>
</el-input>

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

:::

Input avec icône

Ajoutez une icône pour indiquer le type d'input.

:::demo Pour ajouter une icône, vous pouvez utiliser les attributs `prefix-icon` et `suffix-icon` . De plus, les slots nommés `prefix` et `suffix` fonctionnent aussi.

```
<div class="demo-input-suffix">
  <span class="demo-input-label">Avec les attributs</span>
  <el-input
    placeholder="Entrez une date"
    suffix-icon="el-icon-date"
```

```
    v-model="input1">
  </el-input>
  <el-input
    placeholder="Entrez du texte"
    prefix-icon="el-icon-search"
    v-model="input2">
  </el-input>
</div>
<div class="demo-input-suffix">
  <span class="demo-input-label">Avec les slots</span>
  <el-input
    placeholder="Entrez une date"
    v-model="input3">
    <i slot="suffix" class="el-input__icon el-icon-date"></i>
  </el-input>
  <el-input
    placeholder="Entrez du texte"
    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>
```

:::

Zone de texte

Une zone de texte de taille réglable à la souris pour écrire plusieurs lignes. Ajoutez l'attribut `type="textarea"` pour changer `input` en un `textarea` natif.

:::demo Réglez la hauteur grâce à la propriété `rows`.

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

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

:::

Zone de texte avec taille automatique

Configurer la propriété `autosize` pour une zone de texte permet de rendre la hauteur automatique en fonction de la taille du texte. Un objet options pour être fournit à `autosize` les nombres minimal et maximal de lignes.

:::demo

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

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

:::

Input mixte

Ajouter un élément avant ou après l'input, généralement du texte ou un bouton.

:::demo Utilisez `slot` pour ajouter des éléments avant ou après l'input.

```
<div>
  <el-input placeholder="Entrez quelque chose" v-model="input1">
    <template slot="prepend">Http://</template>
  </el-input>
</div>
<div style="margin-top: 15px;">
  <el-input placeholder="Entrez quelque chose" v-model="input2">
    <template slot="append">.com</template>
  </el-input>
</div>
<div style="margin-top: 15px;">
  <el-input placeholder="Entrez quelque chose" v-model="input3" class="input-with-
select">
    <el-select v-model="select" slot="prepend" placeholder="Choisir">
      <el-option label="Restaurant" value="1"></el-option>
      <el-option label="Num. Commande" 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>
```

:::

Tailles

:::demo Ajoutez l'attribut `size` pour changer la taille de l'input. En plus de la taille par défaut, il y a trois autres options: `large`, `small` et `mini`.

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

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

:::

Autocomplétion

Vous pouvez obtenir de l'aide ou des suggestions basées sur ce que vous entrez.

:::demo Le composant d'autocomplétion fournit des suggestions d'entrées. L'attribut `fetch-suggestions` est une méthode qui retourne les suggestions. Dans cet exemple, `querySearch(queryString, cb)` renvoie des suggestions à l'autocomplétion via `cb(data)` quand elles sont prêtes.

```
<el-row class="demo-autocomplete">
  <el-col :span="12">
    <div class="sub-title">Liste des suggestions au focus</div>
    <el-autocomplete
      class="inline-input"
      v-model="state1"
      :fetch-suggestions="querySearch"
      placeholder="Entrez quelque chose"
      @select="handleSelect"
    ></el-autocomplete>
  </el-col>
  <el-col :span="12">
    <div class="sub-title">Liste des suggestions à l'écriture</div>
    <el-autocomplete
      class="inline-input"
      v-model="state2"
      :fetch-suggestions="querySearch"
      placeholder="Entrez quelque chose"
      :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 personnalisé

Vous pouvez personnaliser la manière dont les suggestions sont affichées.

:::demo Utilisez `scoped slot` pour personnaliser les différentes suggestions. Dans le scope, vous pouvez accéder à l'objet suggestion via la clé `item`.

```
<el-autocomplete
  popper-class="my-autocomplete"
  v-model="state"
  :fetch-suggestions="querySearch"
  placeholder="Entrez quelque chose"
  @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>
```

:::

Recherche distante

Vous pouvez aller chercher des infos de suggestions sur un serveur distant.

:::demo

```
<el-autocomplete
  v-model="state"
  :fetch-suggestions="querySearchAsync"
  placeholder="Entrez quelque chose"
  @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>
```

:::

Taille limite

:::demo `maxlength` et `minlength` sont des attributs natifs, indiquant la taille limite de l'input. Le nombre de caractères est mesuré par la taille de la chaine Javascript. Si vous utilisez `maxlength`, vous pourrez montrer le nombre de caractères en mettant `show-word-limit` à `true`.

```
<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>
```

:::

Attributs de l'Input

Attribut	Description	Type	Valeurs acceptées	Défaut
type	Type de l'input.	string	text, textarea et autres types d'input natifs	text
value / v-model	Variable liée.	string / number	—	—
maxlength	Identique à `maxlength` dans l'input natif.	number	—	—
minlength	Identique à `minlength` dans l'input natif.	number	—	—
show-word-limit	Affiche le nombre de caractères restant， ne marche que lorsque `type` est 'text' ou 'textarea'.	boolean	—	false
placeholder	Placeholder de l' Input.	string	—	—
clearable	Si le bouton de reset apparaît.	boolean	—	false
show-password	Si le champ doit un champ de mot de passe avec bouton de visualisation.	boolean	—	false
disabled	Si le champ est désactivé.	boolean	—	false
size	Taille du champ, marche quand `type` n'est pas 'textarea'.	string	medium / small / mini	—
prefix-icon	Classe de l'icône de préfixe.	string	—	—
suffix-icon	Classe de l'icône de suffixe.	string	—	—
rows	Nombre de ligne pour une zone de texte, ne marche que si `type` est 'textarea'.	number	—	2
autosize	Si la zone de texte à une hauteur adaptative, ne marche que si `type` est 'textarea'. Peut accepter un objet, e.g. { minRows: 2, maxRows: 6 }	boolean / object	—	false
autocomplete	Identique à `autocomplete` dans l'input natif.	string	on / off	off
auto-complete	@DEPRECATED dans la prochaine version majeure.	string	on/off	off
name	Identique à `name` dans l'input natif.	string	—	—
readonly	Identique à `readonly` dans l'input natif.	boolean	—	false
max	Identique à `max` dans l'input natif.	—	—	—
min	Identique à `min` dans l'input natif.	—	—	—
step	Identique à `step` dans l'input natif.	—	—	—

resize	Contrôle les changements de taille autorisés.	string	none, both, horizontal, vertical	—
autofocus	Identique à `autofocus` dans l'input natif.	boolean	—	false
form	Identique à `form` dans l'input natif.	string	—	—
label	Texte du label.	string	—	—
tabindex	tabindex de l'input.	string	-	-
validate-event	Si la validation doit avoir lieu.	boolean	-	true

Slots de l'Input

Nom	Description
prefix	Contenu du préfixe, ne marche que si `type` est 'text'.
suffix	Contenu du suffixe, ne marche que si `type` est 'text'.
prepend	Contenu à ajouter avant l'Input, ne marche que si `type` est 'text'.
append	Contenu à ajouter après l'Input, ne marche que si `type` est 'text'.

Évènements

Nom	Description	Paramètres
blur	Se déclenche quand Input perds le focus.	(event: Event)
focus	Se déclenche quand Input a le focus.	(event: Event)
change	Se déclenche quand la valeur change.	(value: string \ number)
change	Déclenché uniquement lorsque la zone de saisie perd le focus ou que l'utilisateur appuie sur Entrée.	(value: string | number)
input	Déclenché lorsque la valeur d'entrée change.	(value: string | number)
clear	Se déclenche quand le champ est effacé par le bouton de reset.	—

Méthodes de l'Input

Méthode	Description	Paramètres
focus	Met le focus sur le champ.	—
blur	Retire le focus de le champ.	—
select	Sélectionne le texte du champ.	—

Attributs de l'autocomplétion

Attribut	Description	Type	Options	Défaut
placeholder	Le placeholder de l'autocomplétion.	string	—	—
clearable	Si un bouton d'effacement doit apparaître.	boolean	—	false
disabled	Si l'autocomplétion est déactivée.	boolean	—	false
value-key	Nom de la clé de l'objet suggestion pour l'affichage.	string	—	value
icon	Nom de l'icône.	string	—	—
value	Variable liée.	string	—	—
debounce	Délai d'attente après écriture, en millisecondes.	number	—	300
placement	Emplacement du menu popup.	string	top / top-start / top-end / bottom / bottom-start / bottom-end	bottom-start
fetch-suggestions	La méthode pour rechercher les suggestions. Lorsqu'elles sont prêtes, appelle `callback(data:[])` pour les renvoyer à l'autocomplétion.	Function(queryString, callback)	—	—
popper-class	Nom de classe pour le menu de l'autocomplétion.	string	—	—
trigger-on-focus	Si les suggestions doivent apparaître quand l'input a le focus.	boolean	—	true
name	Identique à `name` dans l'input natif.	string	—	—
select-when-unmatched	Si un évènement `select` doit être émis après pression sur entrée quand il n'y a pas de résultats.	boolean	—	false
label	Texte du label.	string	—	—
prefix-icon	Classe de l'icône de préfixe.	string	—	—
suffix-icon	Classe de l'iône de suffixe.	string	—	—
hide-loading	Si l'icône de chargement doit être cachée dans le cas d'une recherche distante.	boolean	—	false

popper-append-to-body	Si le menu doit être ajouter au body. Si le positionnement du menu est incorrect, essayez de mettre cette propriété à `false`.	boolean	-	true
highlight-first-item	Si la première suggestion de la liste issue de la recherche distante doit être en surbrillance par défaut.	boolean	—	false

Slots de l'autocomplétion

Nom	Description
prefix	Contenu du préfixe.
suffix	Contenu du suffixe.
prepend	Contenu à ajouter avant le champ.
append	Contenu à ajouter après le champ.

Template personnalisé pour l'autocomplétion

Nom	Description
—	Contenu personnalisé pour les suggestions. Le paramètre de scope est { item }.

Évènements de l'autocomplétion

Nom	Description	Paramètres
select	Se déclenche quand une suggestion est cliquée.	La suggestion sélectionnée.
change	Se déclenche quand la valeur change.	(value: string \ number)

Méthodes de l'autocomplétion

Méthode	Description	Paramètres
focus	Met le focus sur l'élément.	—