<input> 元素

<input> 元素主要用于表单组件，它继承了 HTMLInputElement 接口。

1. HTMLInputElement 的实例属性

1.1 特征属性

- name：字符串，表示 <input> 节点的名称。该属性可读写。
- type：字符串，表示 <input> 节点的类型。该属性可读写。
- disabled：布尔值，表示 <input> 节点是否禁止使用。一旦被禁止使用，表单提交时不会包含该 <input> 节点。该属性可读写。
- autofocus：布尔值，表示页面加载时，该元素是否会自动获得焦点。该属性可读写。
- required：布尔值，表示表单提交时，该 <input> 元素是否必填。该属性可读写。
- value：字符串，表示该 <input> 节点的值。该属性可读写。
- validity：返回一个 ValidityState 对象，表示 <input> 节点的校验状态。该属性只读。
- validationMessage：字符串，表示该 <input> 节点的校验失败时，用户看到的报错信息。如果该节点不需要校验，或者通过校验，该属性为空字符串。该属性只读。
- willValidate：布尔值，表示表单提交时，该 <input> 元素是否会被校验。该属性只读。

1.2. 表单相关属性

- form：返回 <input> 元素所在的表单（<form>）节点。该属性只读。
- formAction：字符串，表示表单提交时的服务器目标。该属性可读写，一旦设置了这个属性，会覆盖表单元素的 action 属性。
- formEncType：字符串，表示表单提交时数据的编码方式。该属性可读写，一旦设置了这个属性，会覆盖表单元素的 enctype 的属性。
- formMethod：字符串，表示表单提交时的 HTTP 方法。该属性可读写，一旦设置了这个属性，会覆盖表单元素的 method 属性。
- formNoValidate：布尔值，表示表单提交时，是否要跳过校验。该属性可读写，一旦设置了这个属性，会覆盖表单元素的 formNoValidate 属性。
- formTarget：字符串，表示表单提交后，服务器返回数据的打开位置。该属性可读写，一旦设置了这个属性，会覆盖表单元素的 target属性。

1.3. 文本输入框的特有属性

以下属性只有在 <input> 元素可以输入文本时才有效。

- autocomplete：字符串 on 或 off，表示该 <input> 节点的输入内容可以被浏览器自动补全。该属性可读写。
- maxLength：整数，表示可以输入的字符串最大长度。如果设为负整数，会报错。该属性可读写。
- size：整数，表示 <input> 节点的显示长度。如果类型是 text 或 password，该属性的单位是字符个数，否则单位是像素。该属性可读写。
- pattern：字符串，表示 <input> 节点的值应该满足的正则表达式。该属性可读写。
- placeholder：字符串，表示该 <input> 节点的占位符，作为对元素的提示。该字符串不能-包含回车或换行。该属性可读写。
- readOnly：布尔值，表示用户是否可以修改该节点的值。该属性可读写。

- `min`：字符串，表示该节点的最小数值或日期，且不能大于 `max` 属性。该属性可读写。
- `max`：字符串，表示该节点的最大数值或日期，且不能小于 `min` 属性。该属性可读写。
- `selectionStart`：整数，表示选中文本的起始位置。如果没有选中文本，返回光标在 `<input>`元素内部的位置。该属性可读写。
- `selectionEnd`：整数，表示选中文本的结束位置。如果没有选中文本，返回光标在 `<input>` 元素内部的位置。该属性可读写。
- `selectionDirection`：字符串，表示选中文本的方向。可能的值包括 `forward`（与文字书写方向一致）、`backward`（与文字书写方向相反）和 `none`（文字方向未知）。该属性可读写。

1.4. 复选框和单选框的特有属性

如果 `<input>` 元素的类型是复选框（`checkbox`）或单选框（`radio`），会有下面的特有属性。

- `checked`：布尔值，表示该 `<input>` 元素是否选中。该属性可读写。
- `defaultChecked`：布尔值，表示该 `<input>` 元素默认是否选中。该属性可读写。
- `indeterminate`：布尔值，表示该 `<input>` 元素是否还没有确定的状态。一旦用户点击过一次，该属性就会变成 `false`，表示用户已经给出确定的状态了。该属性可读写。

1.5. 图像按钮的特有属性

如果 `<input>` 元素的类型是 `image`，就会变成一个图像按钮，会有下面的特有属性。

- `alt`：字符串，图像无法显示时的替代文本。该属性可读写。
- `height`：字符串，表示该元素的高度（单位像素）。该属性可读写。
- `src`：字符串，表示该元素的图片来源。该属性可读写。
- `width`：字符串，表示该元素的宽度（单位像素）。该属性可读写。

1.6. 文件上传按钮的特有属性

如果 `<input>` 元素的类型是 `file`，就会变成一个文件上传按钮，会有下面的特有属性。

- `accept`：字符串，表示该元素可以接受的文件类型，类型之间使用逗号分隔。该属性可读写。
- `files`：返回一个FileList实例对象，包含了选中上传的一组File实例对象。

1.7. 其他属性

- `defaultValue`：字符串，表示该 `<input>` 节点的原始值。
- `dirName`：字符串，表示文字方向。
- `accessKey`：字符串，表示让该 `<input>` 节点获得焦点的某个字母键。
- `list`：返回一个 `<datalist>` 节点，该节点必须绑定 `<input>` 元素，且 `<input>` 元素的类型必须可以输入文本，否则无效。该属性只读。
- `multiple`：布尔值，表示是否可以选择多个值。
- `labels`：返回一个 `NodeList` 实例，代表绑定当前 `<input>` 节点的 `<label>` 元素。该属性只读。
- `step`：字符串，表示在 `min` 属性到 `max` 属性之间，每次递增或递减时的数值或时间。
- `valueAsDate`：Date 实例，一旦设置，该 `<input>` 元素的值会被解释为指定的日期。如果无法解析该属性的值，`<input>` 节点的值将是 `null`。
- `valueAsNumber`：浮点数，当前 `<input>` 元素的值会被解析为这个数值。

2. HTMLInputElement 的实例方法

- `focus()`：当前 `<input>` 元素获得焦点。
- `blur()`：移除 `<input>` 元素的焦点。
- `select()`：选中 `<input>` 元素内部的所有文本。该方法不能保证 `<input>` 获得焦点，最好先用 `focus()` 方法，再用这个方法。
- `click()`：模拟鼠标点击当前的 `<input>` 元素。
- `setSelectionRange()`：选中 `<input>` 元素内部的一段文本，但不会将焦点转移到选中的文本。该方法接受三个参数，第一个参数是开始的位置（从0开始），第二个参数是结束的位置（不包括该位置），第三个参数是可选的，表示选择的方向，有三个可能的值（`forward`、`backward` 和默认值 `none`）。
- `setRangeText()`：新文本替换选中的文本。该方法接受四个参数，第一个参数是新文本，第二个参数是替换的开始位置(从 `0` 开始计算)，第三个参数是结束位置（该位置不包括在内），第四个参数表示替换后的行为（可选），有四个可能的值：`select`（选中新插入的文本）、`start`（光标位置移到插入的文本之前）、`end`（光标位置移到插入的文本之后）、`preserve`（默认值，如果原先就有文本被选中且本次替换位置与原先选中位置有交集，则替换后同时选中新插入的文本与原先选中的文本，否则保持原先选中的文本）。
- `setCustomValidity()`：该方法用于自定义校验失败时的报错信息。它的参数就是报错的提示信息。注意，一旦设置了自定义报错信息，该字段就不会校验通过了，因此用户重新输入时，必须将自定义报错信息设为空字符串，请看下面的例子。
- `checkValidity()`：返回一个布尔值，表示当前节点的校验结果。如果返回 `false`，表示不满足校验要求，否则就是校验成功或不必校验。
- `stepDown()`：将当前 `<input>` 节点的值减少一个步长。该方法可以接受一个整数 `n` 作为参数，表示一次性减少 `n` 个步长，默认是 `1`。有几种情况会抛错：当前 `<input>` 节点不适合递减或递增、当前节点没有 `step` 属性、`<input>` 节点的值不能转为数字、递减之后的值小于`min` 属性或大于 `max` 属性。
- `stepUp()`：将当前 `<input>` 节点的值增加一个步长。其他与 `stepDown()` 方法相同。

`setSelectionRange()` 方法例子：

```
/*
  <p><input type="text" id="mytextbox" size="20" value="HelloWorld"/></p>
  <p><button onclick="SelectText()">选择文本</button></p>
*/
function SelectText() {
  let input = document.getElementById('mytextbox');
  input.focus();
  input.setSelectionRange(2, 5);
}
```

上面代码中，点击按钮以后，会选中llo三个字符。

`setCustomValidity()` 的例子：

```
/*
  <form id="form">
    <input id="field" type="text" pattern="[a-f,0-9]{4}" autocomplete=off />
  </form>
*/
const form   = document.querySelector('#form');
const field  = document.querySelector('#field');
```

```
form.addEventListener('submit', (e) => {
  e.preventDefault(); // 防止这个例子发出 POST 请求
});
field.oninvalid = (event) => {
  event.target.setCustomValidity('必须是一个 4 位十六进制数');
}
field.oninput = (event) => {
  event.target.setCustomValidity('');
}
```

上面代码中，输入框必须输入一个 4 位的十六进制数。如果不满足条件（比如输入xxx），按下回车键以后，就会提示自定义的报错信息。一旦自定义了报错信息，输入框就会一直处于校验失败状态，因此重新输入时，必须把自定义报错信息设为空字符串。另外，为了避免自动补全提示框遮住报错信息，必须将输入框的 `autocomplete` 属性关闭。