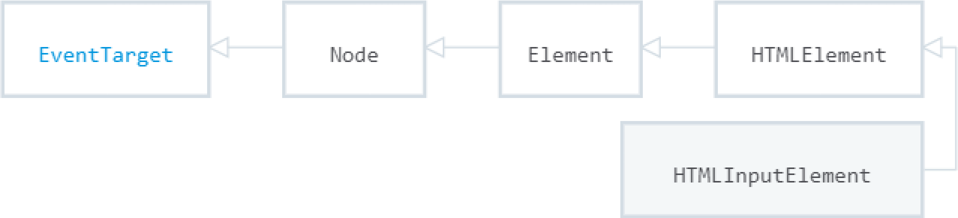
UT-6-B.O.M.-III
ESTUDIO DE LAS API´S
DE LOS CONTROLES INTERACTIVOS
DEPENDIENTES DEL FORMULARIO

1. Introducción

https://developer.mozilla.org/es/docs/Web/API/HTMLInputElement

La interface **HTMLInputElement** proporciona propiedades y métodos especiales para manipular las opciones, estructura y presentación de los elementos **<input>.**



2.- PROPIEDADES

2.1.- Propiedades relacionadas al formulario padre

form Read only	*HTMLFormElement object:* Retorna una referencia al elemento <form> padre
formAction	*string:* **Retorna / Establece** los atributos de los elementos formaction, conteniendo la URI de un programa que procesa la información enviada por el elemento. Esto invalida al atributo action del formulario padre.
formEncType	*string:* **Returns / Sets** the element's formenctype attribute, containing the type of content that is used to submit the form to the server. This overrides the enctype attribute of the parent form.
formMethod	*string:* **Returns / Sets** the element's formmethod attribute, containing the HTTP method that the browser uses to submit the form. This overrides the method attribute of the parent form.
formNoValidate	*boolean:* **Returns / Sets** the element's formnovalidate attribute, indicating that the form is not to be validated when it is submitted. This overrides the novalidate attribute of the parent form.
formTarget	*string:* **Returns / Sets** the element's formtarget attribute, containing a name or keyword indicating where to display the response that is received after submitting the form. This overrides the target attribute of the parent form.

2.2.- Propiedades que aplican a cualquier tipo de elemento input que no se encuentre oculto.

name	*string:* **Returns / Sets** the element's name attribute, containing a name that identifies the element when submitting the form.
type	*string:* **Returns / Sets** the element's type attribute, indicating the type of control to display. See type attribute of <input> for possible values.
disabled	*boolean:* **Returns / Sets** the element's disabled attribute, indicating that the control is not available for interaction. The input values will not be submitted with the form. See also readonly
autofocus	*boolean:* **Returns / Sets** the element's autofocus attribute, which specifies that a form control should have input focus when the page loads, unless the user overrides it, for example by typing in a different control. Only one form element in a document can have the autofocus attribute. It cannot be applied if the type attribute is set to hidden (that is, you cannot automatically set focus to a hidden control).
required	*boolean:* **Returns / Sets** the element's required attribute, indicating that the user must fill in a value before submitting a form.
value	*string:* **Returns / Sets** the current value of the control. **Nota:** Si el usuario introduce un valor diferente al esperado, esto puede retornar una cadena vacía.
Validity Read only	*ValidityState (en-US) object:* Returns the validity state that this element is in.
validationMessage Read only	*string:* **Returns** a localized message that describes the validation constraints that the control does not satisfy (if any). This is the empty string if the control is not a candidate for constraint validation (willvalidate is false), or it satisfies its constraints.
willValidate Read only	*Boolean:* **Indicates** whether the element is a candidate for constraint validation. It is false if any conditions bar it from constraint validation.

2.3.- Propiedades que aplican únicamente los elementos de tipo text o number.

autocomplete	*string:* **Returns / Sets** the element's autocomplete attribute, indicating whether the value of the control can be automatically completed by the browser. Ignored if the value of the type attribute is hidden, checkbox, radio, file, or a button type (button, submit, reset, image). Possible values are: "on": the browser can autocomplete the value using previously stored value "off": the user must explicity enter a value
maxLength	*long:* **Returns / Sets** the element's maxlength attribute, containing the **maximum length of characters** (in Unicode code points) that the value can have. (If you set this to a negative number, an exception will be thrown.)
size	*unsigned long:* **Returns / Sets** the element's size attribute, containing **size of the control**. This value is in pixels unless the value of type is text or password, in which case, it is an integer number of characters. Applies only when type is set to text, search, tel, url, email, or password; otherwise it is ignored.
pattern	*string:* **Returns / Sets** the element's pattern attribute, containing a **regular expression** that the control's value is checked against. Use the title attribute to describe the pattern to help the user. This attribute applies when the value of the type attribute is text, search, tel, url or email; otherwise it is ignored.
placeholder	*string:* **Returns / Sets** the element's placeholder attribute, containing a hint to the user of what can be entered in the control. The placeholder text must not contain carriage returns or line-feeds. This attribute applies when the value of the type attribute is text, search, tel, url or email; otherwise it is ignored.
readOnly	*boolean:* **Returns / Sets** the element's readonly attribute, indicating that the user cannot modify the value of the control. HTML5This is ignored if the value of the type attribute is hidden, range, color, checkbox, radio, file, or a button type.
min	*string:* **Returns / Sets** the element's min attribute, containing the minimum (numeric or date-time) value for this item, which must not be greater than its maximum (max attribute) value.
max	*string:* **Returns / Sets** the element's max attribute, containing the maximum (numeric or date-time) value for this item, which must not be less than its minimum (**min** attribute) value.
selectionStart	*unsigned long:* **Returns / Sets** the beginning index of the selected text. When nothing is selected, this returns the position of the text input cursor (caret) inside of the <input> element.

selectionEnd	*unsigned long:* **Returns / Sets** the end index of the selected text. When there's no selection, this returns the offset of the character immediately following the current text input cursor position.
selectionDirection	*string:* **Returns / Sets** the direction in which selection occurred. Possible values are: "forward" if selection was performed in the start-to-end direction of the current locale, "backward" for the opposite direction, "none" if the direction is unknown."

2.4.- Propiedades que aplican solo a los elementos de tipo checkbox o radio.

Las casillas de verificación o "***checkboxes***" nos permiten seleccionar varias opciones marcando el cuadrito que aparece a su izquierda. La marca de la casilla equivale a un valor "true" y si no está marcada equivale a un valor "false".

El objeto "***radio***" nos permitirá elegir una posibilidad de entre todas las que hay. (ej: Mensual, Trimestral, Anual). Todos los botones de un grupo ***van a compartir el mismo nombre***, de esta manera JavaScript conoce al grupo de botones de tal forma que al hacer clic en uno de ellos se desactive el resto de botones del grupo.

checked	*boolean:* **Returns / Sets** the current state of the element when type is checkbox or radio.
defaultChecked	*boolean:* **Returns / Sets** the default state of a radio button or checkbox as originally specified in HTML that created this object.
indeterminate	*boolean:* **indicates** that the checkbox is neither on nor off. Changes the appearance to resemble a third state. Does not affect the value of the **checked** attribute, and clicking the checkbox will set the value to false.

2.5.- Propiedades que sólo aplican a los elementos de tipo image.

alt	*string:* **Returns / Sets** the element's alt attribute, containing alternative text to use when type is image.
height	*string:* **Returns / Sets** the element's height attribute, which defines the height of the image displayed for the button, if the value of type is image.
src	*string:* **Returns / Sets** the element's src attribute, which specifies a URI for the location of an image to display on the graphical submit button, if the value of type is image; otherwise it is ignored.
width	*string:* **Returns / Sets** the document's width attribute, which defines the width of the image displayed for the button, if the value of type is image.

2.6.- Propiedades que sólo aplican a los elementos de tipo file

accept	*string:* **Returns / Sets** the element's accept attribute, containing comma-separated list of file types accepted by the server when type is file.
files	**Returns/accepts** a FileList (en-US) object, which contains a list of File objects representing the files selected for upload.

2.7.- Propiedades aún no categorizadas.

defaultValue	*string:* **Returns / Sets** the default value as originally specified in the HTML that created this object.
dirName	*string:* **Returns / Sets** the directionality of the element.
accessKey	*string:* **Returns** a string containing a single character that switches input focus to the control when pressed.
list Read only	*HTMLElement object:* **Returns** the element pointed by the list attribute. The property may be null if no HTML element found in the same tree.
multiple	*boolean:* **Returns / Sets** the element's multiple attribute, indicating whether more than one value is possible (e.g., multiple files).
files	*FileList (en-US) array:* **Returns** the list of selected files.
HTMLInputElement.labels (en-US) Read Only	*Node list* array: **Returns** a list of <label> elements that are labels for this element.
step	*string:* **Returns / Sets** the element's step attribute, which works with min and max to limit the increments at which a numeric or date-time value can be set. It can be the string any or a positive floating point number. If this is not set to any, the control accepts only values at multiples of the step value greater than the minimum.
valueAsDate	*Date object:* Returns / Sets the value of the element, interpreted as a date, or null if conversion is not possible.
valueAsNumber	double: **Returns** the value of the element, interpreted as one of the following, in order: • a time value • a number • NaN if conversion is impossible
autocapitalize	*string:* **defines** the capitalization behavior for user input. Valid values are none, off, characters, words, or sentences.

3.- Métodos.

focus()	Focus on the input element; keystrokes will subsequently go to this element.
blur()	Removes focus from input; keystrokes will subsequently go nowhere
select()	Selects the input text in the element, and focuses it so the user can subsequently replace the whole entry.
click()	Simulates a click on the element.
setSelectionRange()	Selects a range of text in the element (but does not focus it). The optional selectionDirection parameter may be "forward" or "backward" to establish the direction in which selection was set, or "none" if the direction is unknown or not relevant. The default is "none". Specifying a selectionDirection parameter sets the value of the selectionDirection property.
setRangeText()	Replaces a range of text with the new text. If the *start* and *end* arguments are not provided, the range is assumed to be the selection.The final argument determines how the selection should be set after the text has been replaced. The possible values are: **"select"** Selects the newly inserted text. **"start"** Moves the selection to just before the inserted text. **"end"** Moves the selection to just after the selected text. **"preserve"** Attempts to preserve the selection. This is the default.
setCustomValidity()	Sets a custom validity message for the element. If this message is not the empty string, then the element is suffering from a custom validity error, and does not validate.
checkValidity()	Returns a Boolean that is false if the element is a candidate for constraint validation, and it does not satisfy its constraints. In this case, it also fires an invalid (en-US) event at the element. It returns true if the element is not a candidate for constraint validation, or if it satisfies its constraints.

4.- Eventos.

Responde a este elemento usando ***addEventListener()*** o asignando una respuesta a la propiedad ***oneventname*** de esta interface:

input

Se activa cuando el ***value*** de un elemento <input>, <select>, o <textarea> ha sido modificado.

Nota: Esto es actualmente contemplado en la interface ***HTMLElement*** y también aplica a los elementos ***contenteditable***, pero se ha reflejado aquí porque es más utilizado con los elementos de entrada de formularios.

También disponible por la propiedad controladora de evento ***oninput***.

invalid

Fired when an element does not satisfy its constraints during constraint validation.

Also available via the ***oninvalid*** event handler property.

search

Fired when a search is initiated on an <input> of type="search".

Also available via the ***onsearch*** event handler property.

5.- <input type="hidden">

https://developer.mozilla.org/es/docs/Web/HTML/Element/input/hidden.

<input> elements of type "hidden" let web developers include data that cannot be seen or modified by users when a form is submitted. For example, the ID of the content that is currently being ordered or edited, or a unique security token. Hidden inputs are completely invisible in the rendered page, and there is no way to make it visible in the page's content.

(ver ejemplo en la url indicada)

Para acceder a las propiedades:

```
[window.]document.nombreformulario.nombrecampo.propiedad
[window.]document.nombreformulario.elements[n].propiedad
[window.]document.forms[m].nombrecampo.propiedad
[window.]document.forms[m].elements[n].propiedad
```

Son útiles para las aplicaciones CGI que implican múltiples pantallas y se suele utilizar para conservar información de estado entre las páginas.

CGI:

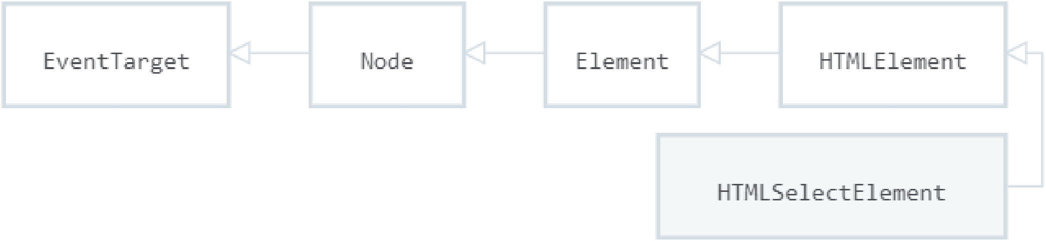
Interfaz de entrada común (en inglés *Common Gateway Interface*, abreviado CGI) es una importante tecnología de la World Wide Web que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa cuyo resultado final de la ejecución son objetos MIME.

Las aplicaciones que se ejecutan en el servidor reciben el nombre de CGIs.
Las aplicaciones CGI fueron una de las primeras prácticas de crear contenido dinámico para las páginas web.
En una aplicación CGI, el servidor web pasa las solicitudes del cliente a un programa externo. Este programa puede estar escrito en cualquier lenguaje que soporte el servidor, aunque por razones de portabilidad se suelen usar lenguajes de script. La salida de dicho programa es enviada al cliente en lugar del archivo estático tradicional.
CGI ha hecho posible la implementación de funciones nuevas y variadas en las páginas web, de tal manera que esta interfaz rápidamente se volvió un estándar, siendo implementada en todo tipo de servidores web.

6.- HTMLSelectElement

6.1.-Introducción.

The HTMLSelectElement interface represents a <select> HTML Element. These elements also share all of the properties and methods of other HTML elements via the HTMLElement interface.



6.2.- Properties

This interface inherits the properties of HTMLElement, and of Element and Node.

autofocus	A Boolean reflecting the autofocus HTML attribute, which indicates whether the control should have input focus when the page loads, unless the user overrides it, for example by typing in a different control. Only one form-associated element in a document can have this attribute specified.
disabled	A Boolean reflecting the disabled HTML attribute, which indicates whether the control is disabled. If it is disabled, it does not accept clicks.
form Read Only	An HTMLFormElement referencing the form that this element is associated with. If the element is not associated with of a <form> element, then it returns null.
labels Read Only	A NodeList of <label> elements associated with the element.
length	An unsigned long The number of <option> elements in this select element.
multiple	A Boolean reflecting the multiple HTML attribute, which indicates whether multiple items can be selected.

name	A DOMString reflecting the name HTML attribute, containing the name of this control used by servers and DOM search functions.
options Read Only	An HTMLOptionsCollection (en-US) representing the set of <option> elements contained by this element.
requiered	A Boolean reflecting the required HTML attribute, which indicates whether the user is required to select a value before submitting the form.
selectedIndex	A long reflecting the index of the first selected <option> element. The value -1 indicates no element is selected.
selectedOptions Read Only	An HTMLCollection representing the set of <option> elements that are selected.
size	A long reflecting the size HTML attribute, which contains the number of visible items in the control. The default is 1, unless multiple is true, in which case it is 4.
type Read Only	A DOMString represeting the form control's type. When multiple is true, it returns "select-multiple"; otherwise, it returns "select-one".
validationMessage Read Only	A DOMString representing a localized message that describes the validation constraints that the control does not satisfy (if any). This attribute is the empty string if the control is not a candidate for constraint validation (willValidate is false), or it satisfies its constraints.
validity Read Only	A ValidityState (en-US) reflecting the validity state that this control is in.
value	A DOMString reflecting the value of the form control (the first selected option). Returns the value attribute of the option element or if it is missing, the text attribute.
willValidate Read Only	A Boolean that indicates whether the button is a candidate for constraint validation. It is false if any conditions bar it from constraint validation.

6.3.- Métodos.

This interface inherits the methods of HTMLElement, and of Element and Node.

add()	Adds an element to the collection of option elements for this select element.
checkValidity()	Checks whether the element has any constraints and whether it satisfies them. If the element fails its constraints, the browser fires a cancelable invalid (en-US) event at the element (and returns false).
item()	Gets an item from the options collection for this <select> element. You can also access an item by specifying the index in array-style brackets or parentheses, without calling this method explicitly.
namedItem()	Gets the item in the options collection with the specified name. The name string can match either the id or

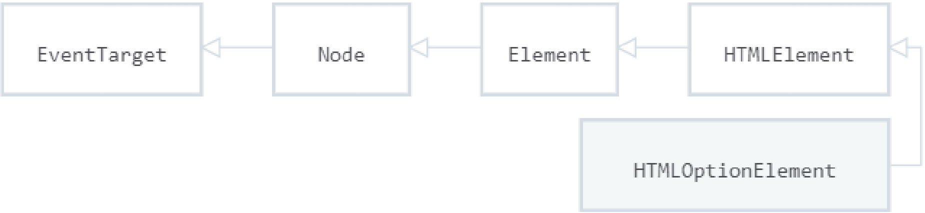
	the name attribute of an option node. You can also access an item by specifying the name in array-style brackets or parentheses, without calling this method explicitly.
remove()	Removes the element at the specified index from the options collection for this select element.
setCustomValidity()	Sets the custom validity message for the selection element to the specified message. Use the empty string to indicate that the element does *not* have a custom validity error.

A better way to track changes to the user's selection is to watch for the change (en-US) event to occur on the <select>. This will tell you when the value changes, and you can then update anything you need to. See the example provided in the documentation for the change event for details.

7.- HTMLOptionElement

https://developer.mozilla.org/en-US/docs/Web/API/HTMLOptionElement
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/option

The HTMLOptionElement interface represents <option> elements and inherits all properties and methods of the HTMLElement interface.



The <option> HTML element is used to define an item contained in a <select>, an <optgroup>, or a <datalist> element. As such, <option> can represent menu items in popups and other lists of items in an HTML document.

Option() is the Constructor.
Returns a newly created HTMLOptionElement object.
It has ***four parameters***: the text to display, ***text***, the value associated, **value**, the ***value of defaultSelected***, and the ***value of selected***. The last three parameters are optional.

7.1.- Properties

Inherits properties from its parent, HTMLElement.

defaultSelected	Has a value of either true or false that shows the initial value of the selected HTML attribute, indicating whether the option is selected by default or not.

disabled	Has a value of either true or false representing the value of the disabled HTML attribute, which indicates that the option is unavailable to be selected. An option can also be disabled if it is a child of an <optgroup> element that is disabled.
form Read only	Is a HTMLFormElement representing the same value as the form of the corresponding <select> element, if the option is a descendant of a <select> element, or null if none is found.
index Read only	Is a long representing the position of the option within the list of options it belongs to, in tree-order. If the option is not part of a list of options, like when it is part of the <datalist> element, the value is 0.
label Read only	Is a DOMString that reflects the value of the label HTML attribute, which provides a label for the option. If this attribute isn't specifically set, reading it returns the element's text content.
selected	Has a value of either true or false that indicates whether the option is currently selected.
text	Is a DOMString that contains the text content of the element.
value	Is a DOMString that reflects the value of the value HTML attribute, if it exists; otherwise reflects value of the Node.textContent property.

7.2.- Methods

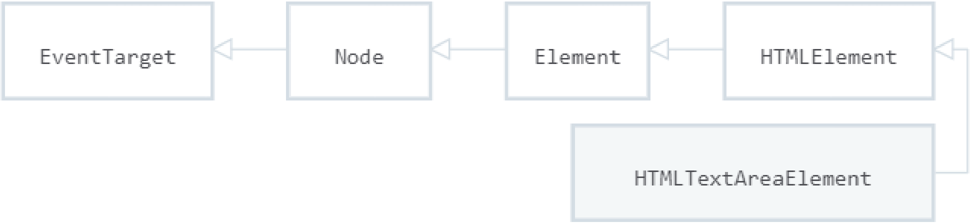
Doesn't implement any specific method, but inherits methods from its parent, HTMLElement.

(hacer ejemplo menús)

8.- HTMLTextAreaElement

https://developer.mozilla.org/en-US/docs/Web/API/HTMLTextAreaElement

The HTMLTextAreaElement interface provides special properties and methods for manipulating the layout and presentation of <textarea> elements.



8.1.- Properties.

accessKey	*string*: Returns / Sets the element's accesskey attribute.
autofocus	*boolean*: Returns / Sets the element's autofocus attribute, indicating that the control should have input focus when the page loads
cols	*unsigned long*: Returns / Sets the element's cols attribute, indicating the visible width of the text area.
defaultValue	*string*: Returns / Sets the control's default value, which behaves like the Node.textContent property.
disabled	*boolean*: Returns / Sets the element's disabled attribute, indicating that the control is not available for interaction.
maxLength	*long*: Returns / Sets the element's maxlength attribute, indicating the maximum number of characters the user can enter. This constraint is evaluated only when the value changes.
minLength	*long*: Returns / Sets the element's minlength attribute, indicating the minimum number of characters the user can enter. This constraint is evaluated only when the value changes.
name	*string*: Returns / Sets the element's name attribute, containing the name of the control.
placeholder	*string*: Returns / Sets the element's placeholder attribute, containing a hint to the user about what to enter in the control.
readOnly	*boolean*: Returns / Sets the element's readonly attribute, indicating that the user cannot modify the value of the control.
required	*boolean*: Returns / Sets the element's required attribute, indicating that the user must specify a value before submitting the form.
rows	*unsigned long*: Returns / Sets the element's rows attribute, indicating the number of visible text lines for the control.
selectionDirection	*string*: Returns / Sets the direction in which selection occurred. This is "forward" if selection was performed in the start-to-end direction of the current locale, or "backward" for the opposite direction. This can also be "none" if the direction is unknown.
selectionEnd	*unsigned long*: Returns / Sets the index of the end of selected text. If no text is selected, contains the index of the character that follows the input cursor. On being set, the control behaves as if setSelectionRange() had been called with this as the second argument, and selectionStart as the first argument.
selectionStart	*unsigned long*: Returns / Sets the index of the beginning of selected text. If no text is selected, contains the index of the character that follows the input cursor. On being set, the control behaves as if setSelectionRange() had been called with this as the first argument, and selectionEnd as the second argument.
tabIndex	*long*: Returns / Sets the position of the element in the tabbing navigation order for the current document.

value	*string*: Returns / Sets the raw value contained in the control.

8.2.-Métodos

blur()	Removes focus from the control; keystrokes will subsequently go nowhere.
focus()	Gives focus to the control; keystrokes will subsequently go to this element.
select()	Selects the contents of the control.
setRangeText()	Replaces a range of text in the element with new text.
setSelectionRange()	Selects a range of text in the element (but does not focus it).

8.3.- Events

Listen to these events using ***addEventListener()*** or by assigning an event listener to the ***oneventname*** property of this interface:

input

Fires when the value of a <textarea> element has been changed.

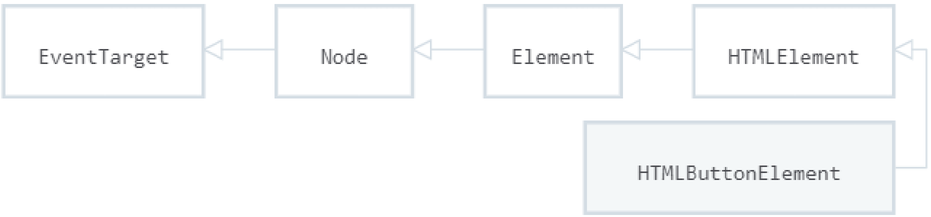
selectionchange

Fires when the text selection in a <textarea> element has been changed.

9.- HTMLButtonElement

https://developer.mozilla.org/en-US/docs/Web/API/HTMLButtonElement

The HTMLButtonElement interface provides properties and methods (beyond the regular HTMLElement interface it also has available to it by inheritance) for manipulating <button> elements.



9.1.- Properties

accesKey	Is a DOMString indicating the single-character keyboard key to give access to the button.
autofocus	Is a boolean value indicating whether or not the control should have input focus when the page loads, unless the user overrides it, for example by typing in a different control. Only one form-associated element in a document can have this attribute specified.
disabled	Is a boolean value indicating whether or not the control is disabled, meaning that it does not accept any clicks.

menu	Is a HTMLMenuElement representing the menu element to be displayed if the button is clicked and is of type="menu".
name	Is a DOMString representing the name of the object when submitted with a form. If specified, it must not be the empty string.
tabIndex	Is a long that represents this element's position in the tabbing order.
type	Is a DOMString indicating the behavior of the button. This is an enumerated attribute with the following possible values: • **submit**: The button submits the form. This is the default value if the attribute is not specified, or if it is dynamically changed to an empty or invalid value. • **reset**: The button resets the form. • **button**: The button does nothing. • **menu**: The button displays a menu.
value	Is a DOMString representing the current form control value of the button.