30th November 2019

Form properties and methods

Forms and control elements, such as `<input>` have a lot of special properties and events.

Working with forms will be much more convenient when we learn them.

Navigation: form and elements

Document forms are members of the special collection `document.forms` .

That's a so-called "named collection": it's both named and ordered. We can use both the name or the number in the document to get the form.

```
document.forms.my - the form with name="my"
document.forms[0] - the first form in the document
```

When we have a form, then any element is available in the named collection `form.elements` .

For instance:

```
<form name="my">
  <input name="one" value="1">
  <input name="two" value="2">
</form>

<script>
  // get the form
  let form = document.forms.my; // <form name="my"> element

  // get the element
  let elem = form.elements.one; // <input name="one"> element

  alert(elem.value); // 1
</script>
```

There may be multiple elements with the same name, that's often the case with radio buttons.

In that case `form.elements[name]` is a collection, for instance:

```
<form>
  <input type="radio" name="age" value="10">
  <input type="radio" name="age" value="20">
</form>
```

```

<script>
let form = document.forms[0];

let ageElems = form.elements.age;

alert(ageElems[0]); // [object HTMLInputElement]
</script>
```

These navigation properties do not depend on the tag structure. All control elements, no matter how deep they are in the form, are available in `form.elements` .

Fieldsets as “subforms”

A form may have one or many `<fieldset>` elements inside it. They also have `elements` property that lists form controls inside them.

For instance:

```
<body>
  <form id="form">
    <fieldset name="userFields">
      <legend>info</legend>
      <input name="login" type="text">
    </fieldset>
  </form>

  <script>
    alert(form.elements.login); // <input name="login">

    let fieldset = form.elements.userFields;
    alert(fieldset); // HTMLFieldSetElement

    // we can get the input by name both from the form and from the fields
    alert(fieldset.elements.login == form.elements.login); // true
  </script>
</body>
```

Shorter notation: `form.name`

There's a shorter notation: we can access the element as `form[index/name]` .

In other words, instead of `form.elements.login` we can write `form.login` .

That also works, but there's a minor issue: if we access an element, and then change its `name` , then it is still available under the old name (as well as under the new one).

That's easy to see in an example:

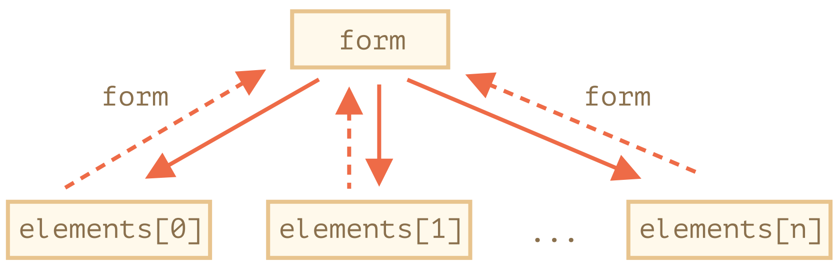
```
<form id="form">
  <input name="login">
</form>

<script>
  alert(form.elements.login == form.login); // true, the same <input>

  form.login.name = "username"; // change the name of the input

  // form.elements updated the name:
  alert(form.elements.login); // undefined
  alert(form.elements.username); // input

  // form allows both names: the new one and the old one
  alert(form.username == form.login); // true
</script>
```

That's usually not a problem, because we rarely change names of form elements.

Backreference: element.form

For any element, the form is available as `element.form` . So a form references all elements, and elements reference the form.

Here's the picture:



For instance:

```
<form id="form">
  <input type="text" name="login">
</form>

<script>
  // form -> element
  let login = form.login;

  // element -> form
  alert(login.form); // HTMLFormElement
</script>
```

Form elements

Let’s talk about form controls.

input and textarea

We can access their value as `input.value` (string) or `input.checked` (boolean) for checkboxes.

Like this:

```
input.value = "New value";
textarea.value = "New text";

input.checked = true; // for a checkbox or radio button
```

Use `textarea.value`, not `textarea.innerHTML`

Please note that even though `<textarea>...</textarea>` holds its value as nested HTML, we should never use `textarea.innerHTML` to access it.

It stores only the HTML that was initially on the page, not the current value.

select and option

A `<select>` element has 3 important properties:

1. `select.options` – the collection of `<option>` subelements,
2. `select.value` – the value of the currently selected `<option>`,
3. `select.selectedIndex` – the number of the currently selected `<option>`.

They provide three different ways of setting a value for a `<select>`:

1. Find the corresponding `<option>` element and set `option.selected` to `true`.
2. Set `select.value` to the value.
3. Set `select.selectedIndex` to the number of the option.

The first way is the most obvious, but `(2)` and `(3)` are usually more convenient.

Here is an example:

```
<select id="select">
  <option value="apple">Apple</option>
  <option value="pear">Pear</option>
  <option value="banana">Banana</option>
</select>

<script>
  // all three lines do the same thing
  select.options[2].selected = true;
  select.selectedIndex = 2;
  select.value = 'banana';
</script>
```

Unlike most other controls, `<select>` allows to select multiple options at once if it has `multiple` attribute. That's feature is rarely used. In that case we need to use the first way: add/remove the `selected` property from `<option>` subelements.

We can get their collection as `select.options`, for instance:

```
<select id="select" multiple>
  <option value="blues" selected>Blues</option>
  <option value="rock" selected>Rock</option>
  <option value="classic">Classic</option>
</select>

<script>
  // get all selected values from multi-select
  let selected = Array.from(select.options)
    .filter(option => option.selected)
    .map(option => option.value);

  alert(selected); // blues,rock
</script>
```

The full specification of the `<select>` element is available in the specification https://html.spec.whatwg.org/multipage/forms.html#the-select-element.

new Option

This is rarely used on its own. But there's still an interesting thing.

In the specification there's a nice short syntax to create `<option>` elements:

```
option = new Option(text, value, defaultSelected, selected);
```

Parameters:

- `text` – the text inside the option,
- `value` – the option value,
- `defaultSelected` – if `true`, then `selected` HTML-attribute is created,
- `selected` – if `true`, then the option is selected.

There may be a small confusion about `defaultSelected` and `selected`. That's simple: `defaultSelected` sets HTML-attribute, that we can get using `option.getAttribute('selected')`.

And `selected` – whether the option is selected or not, that's more important. Usually both values are either set to `true` or not set (same as `false`).

For instance:

```
let option = new Option("Text", "value");
// creates <option value="value">Text</option>
```

The same element selected:

```
let option = new Option("Text", "value", true, true);
```

Option elements have properties:

`option.selected`

Is the option selected.

`option.index`

The number of the option among the others in its `<select>` .

`option.text`

Text content of the option (seen by the visitor).

References

- Specification: https://html.spec.whatwg.org/multipage/forms.html.

Summary

Form navigation:

`document.forms`

A form is available as `document.forms[name/index]` .

`form.elements`

Form elements are available as `form.elements[name/index]` , or can use just `form[name/index]` . The `elements` property also works for `<fieldset>` .

`element.form`

Elements reference their form in the `form` property.

Value is available as `input.value` , `textarea.value` , `select.value` etc, or `input.checked` for checkboxes and radio buttons.

For `<select>` we can also get the value by the index `select.selectedIndex` or through the options collection `select.options` .

These are the basics to start working with forms. We’ll meet many examples further in the tutorial.

In the next chapter we’ll cover `focus` and `blur` events that may occur on any element, but are mostly handled on forms.

Tasks

Add an option to select

importance: 5

There’s a `<select>` :

```html
<select id="genres">
  <option value="rock">Rock</option>
  <option value="blues" selected>Blues</option>
</select>
```

Use JavaScript to:

1. Show the value and the text of the selected option.
2. Add an option: `<option value="classic">Classic</option>` .
3. Make it selected.

Note, if you’ve done everything right, your alert should show `blues` .

solution

Previous lesson Next lesson

Share Tutorial map

Comments

- If you have suggestions what to improve - please submit a GitHub issue or a pull request instead of commenting.
- If you can't understand something in the article – please elaborate.
- To insert a few words of code, use the `<code>` tag, for several lines – use `<pre>` , for more than 10 lines – use a sandbox (plnkr, JSBin, codepen…)