3rd July 2019

Forms: event and method submit

The `submit` event triggers when the form is submitted, it is usually used to validate the form before sending it to the server or to abort the submission and process it in JavaScript.

The method `form.submit()` allows to initiate form sending from JavaScript. We can use it to dynamically create and send our own forms to server.

Let's see more details of them.

Event: submit

There are two main ways to submit a form:

1. The first – to click `<input type="submit">` or `<input type="image">` .
2. The second – press `Enter` on an input field.

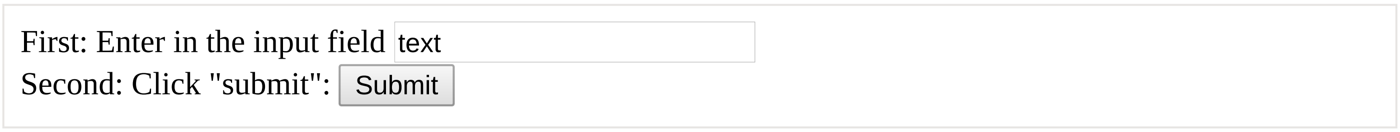
Both actions lead to `submit` event on the form. The handler can check the data, and if there are errors, show them and call `event.preventDefault()` , then the form won't be sent to the server.

In the form below:

1. Go into the text field and press `Enter`.
2. Click `<input type="submit">` .

Both actions show `alert` and the form is not sent anywhere due to `return false` :

```
<form onsubmit="alert('submit!');return false">
  First: Enter in the input field <input type="text" value="text"><br>
  Second: Click "submit": <input type="submit" value="Submit">
</form>
```


First: Enter in the input field text
Second: Click "submit": Submit

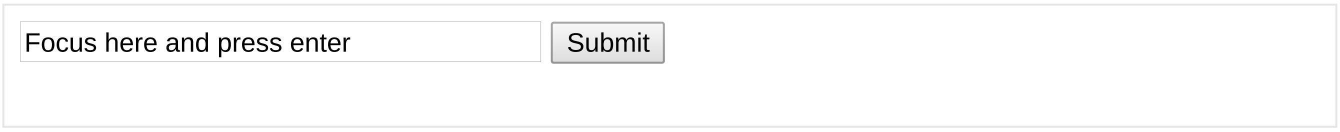
Relation between `submit` and `click`

When a form is sent using `Enter` on an input field, a `click` event triggers on the `<input type="submit">`.

That's rather funny, because there was no click at all.

Here's the demo:

```html
<form onsubmit="return false">
 <input type="text" size="30" value="Focus here and press enter">
 <input type="submit" value="Submit" onclick="alert('click')">
</form>
```



Method: submit

To submit a form to the server manually, we can call `form.submit()`.

Then the `submit` event is not generated. It is assumed that if the programmer calls `form.submit()`, then the script already did all related processing.

Sometimes that's used to manually create and send a form, like this:

```javascript
let form = document.createElement('form');
form.action = 'https://google.com/search';
form.method = 'GET';

form.innerHTML = '<input name="q" value="test">';

// the form must be in the document to submit it
document.body.append(form);

form.submit();
```

Tasks

Modal form

importance: 5

Create a function `showPrompt(html, callback)` that shows a form with the message `html`, an input field and buttons `OK/CANCEL`.

- A user should type something into a text field and press `Enter` or the OK button, then `callback(value)` is called with the value they entered.
- Otherwise if the user presses `Esc` or CANCEL, then `callback(null)` is called.

In both cases that ends the input process and removes the form.

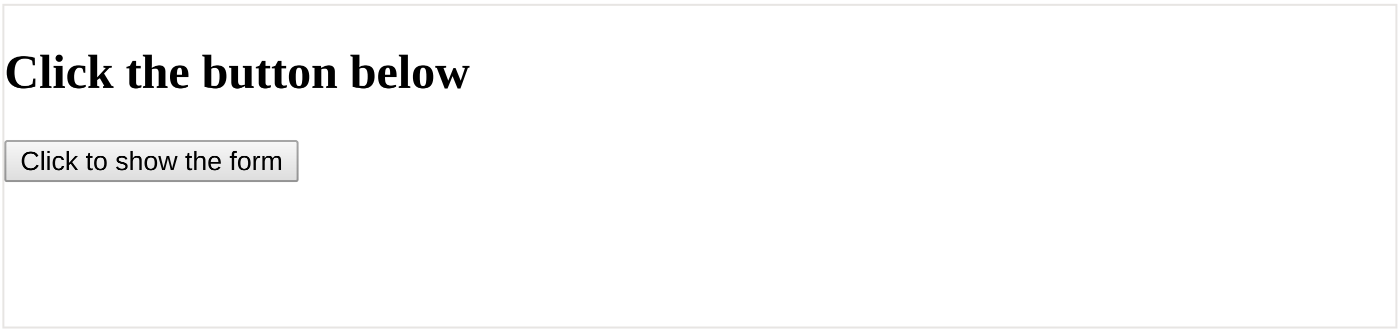
Requirements:

- The form should be in the center of the window.
- The form is *modal*. In other words, no interaction with the rest of the page is possible until the user closes it.
- When the form is shown, the focus should be inside the `<input>` for the user.
- Keys `Tab`/`Shift+Tab` should shift the focus between form fields, don't allow it to leave for other page elements.

Usage example:

```
showPrompt("Enter something<br>...smart :)", function(value) {
  alert(value);
});
```

A demo in the iframe:


Click the button below

P.S. The source document has HTML/CSS for the form with fixed positioning, but it's up to you to make it modal.

Open a sandbox for the task.

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…)