JS Forms 2

Learning Objectives

- knowing ways for client-side form validation
- understanding the input event
- knowing how to focus an input field programmatically
- knowing how to reset a form

HTML Form Validation

Before submitting a form, it is important to ensure all required form fields are filled out, in the correct format. This is called **client-side form validation**.

HTML provides several form field attributes to enable validation features build into the browser.

Attribute	Description
`required`	if present, a form field needs to be filled in before the form can be submitted
`minlength` / `maxlength`	minimum and maximum length of textual data (strings)
`min` / `max`	minimum and maximum values of numerical input types
`type`	each input type has its own prefigured validation (like `email`)
`pattern`	a regular expression pattern the entered data needs to follow

The following input field is valid if its value exists and it is a string between 3 and 30 characters:

```
<input
  id="input-name"
  type="text"
  name="name"
  minlength="3"
  maxlength="30"
  required
/>
```

> ! If the `required` attribute is omitted, the field is valid if it is empty or has a content between 3 and 30 characters, but invalid if 1 or 2 characters are entered.

`type="email"` will check if the input is a valid email address.

```
<input id="input-email" type="email" name="email" />
```

The `input` Event

Occasionally, you may want to do something if the value of a single field changes even before the form is submitted.

The `input` event is fired every time when the value of a form field has been changed. For example, a `<textarea />` will fire this event with every keystroke.

```
const messageField = document.querySelector('[data-js="message"]');

messageField.addEventListener("input", (event) => {
  console.log(event.target.value);
});
```

> ! Don't confuse the `input` event with the `change` event, which is only fired after a field's content has been committed by the user by pressing enter or moving the focus to the next field.

Focus Input Fields

You can focus an input field with the `.focus()` method. This can be used to improve the user experience after submitting a form.

```
const messageField = document.querySelector('[data-js="message"]');

form.addEventListener("submit", (event) => {
  event.preventDefault();
  // [...] handle form data
  messageField.focus();
});
```

Instead of querying the input element using `querySelector`, it can also be obtained via the `event.target.elements` collection:

```
form.addEventListener("submit", (event) => {
  event.preventDefault();
  // [...] handle form data
  event.target.elements.message.focus();
});
```

This will focus a form field with the attribute `name="message"`.

Resetting Forms

You can reset all form fields to their default value with the `.reset()` method.

```
form.addEventListener("submit", (event) => {
  event.preventDefault();
  // [...] handle form data
  event.target.reset();
});
```

This often comes in handy in combination with `.focus()`. Think of a chat: After the message was send, the input field is cleared and re-focussed, so users can write the next message.

Resources

- MDN web docs: Client-side form validation
- MDN web docs: input event