JS Inputs and Strings

Learning Objectives

- learning different ways of writing strings
- using string properties and methods
- working with input elements

Strings

There are three ways to create strings using *string literals*:

1. `'string'`: single quotes
2. `"string"`: double quotes
3. `` `string` ``: back ticks or **template literals**.

> 💡 In general there is no preference for using either single or double quotes, except for stylistic reasons. Tools like prettier convert all strings to use the same style quotes. We have configured prettier to use double quotes by default. One reason to prefer one style of quotes over another on a case-by-case basis is when a quotation mark is part of the string:
>
> - `"It's such a nice day!"`
> - `'"Nice work", they said.'` or `'[data-js="foo"]'`
>
> Prettier detects these cases automatically.

Strings can be chained together by using the `+` operator (yes, the same as the maths operator). This is called **string concatination**:

```js
const name = "Alex";
const stringConcatination = "Hello " + name + ", good to see you!";
```

Template Literals

The third method to write strings has the useful property that you can insert variables into the string by wrapping placeholders with a dollar sign and curly brackets `${}` . This is also called **string interpolation**.

This way you don't have to concat multiple strings if you want to use a variable in your string:

```js
const stringConcatination = "Hello " + name + ", good to see you!";

const withTemplateString = `Hello ${name}, good to see you!`;
```

Any **expression** can be placed into these placeholders:

```
const greeting = `Hello ${
  name !== null ? name : "mysterious person"
}, good to see you!`;
```

With template literals you can also write **multi-line strings**:

```
`Hello,
this is in a new line.
Good bye!`;
```

String Properties and Methods

Strings in JavaScript have some build-in **properties** and functionalities called **methods**. You can call them with the dot notation followed by the name of the property / method.

```
"A normal string".length; // evaluates to 15
"A normal string".toUpperCase(); // evaluates to "A NORMAL STRING"
```

> 💡 Methods are functions, thus they need to be invoked by placing `()` brackets after the name of the method.

Property / Method	Effect
`.length`	returns the number of characters in a string.
`.toUpperCase()`	returns a all uppercase version of the string.
`.toLowerCase()`	returns a all lowercase version of the string.
`.trim()`	returns a string with all whitespace removed from the beginning and end.
`.replaceAll(oldString, newString)`	replaces all occurrences of `oldString` with the `newString`.
`.startsWith(subString)`	returns `true` if the string starts with subString.
`.endsWith(subString)`	returns `true` if the string ends with subString.
`.includes(subString)`	returns `true` if the string contains the subString.

> 💡 Go to the MDN Docs for even more string methods.

Input Fields

Every input field in HTML holds a **value** in form of a string. You can access the value by using `.value` on the input Element:

```html
<form>
  <input data-js="textInput" type="text" value="test 123" />
  <input data-js="numberInput" type="number" value="42" />
</form>
```

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

textInput.value; // evaluates to 'test 123'
numberInput.value; // evaluates to '42' (still a string!)
```

You can also change the value of the input by assigning a new value to this input property:

```js
textInput.value = "changed value!";
```

This change is immediately visible on the website.

For example, you can enforce all uppercase letters in a form by combining this functionality with an `input` event listener on the input element:

```js
// transform on every change the input value to uppercase letters
textInput.addEventListener("input", () => {
  const oldValue = textInput.value;
  const newValue = oldValue.toUpperCase();
  textInput.value = newValue;
});
```

Resources

String Methods

MDN Docs: String Methods