JS Eingaben und Strings

Lernziele

- Verschiedene Möglichkeiten zum Schreiben von Strings lernen
- String-Eigenschaften und -Methoden verwenden
- Mit Eingabeelementen arbeiten

Strings

Es gibt drei Möglichkeiten, Strings mit *String-Literalen* zu erstellen:

1. `'string'`: einfache Anführungszeichen
2. `"string"`: doppelte Anführungszeichen
3. `` `string` ``: Backticks oder **Template Literals**.

> 💡 Im Allgemeinen gibt es keinen bevorzugten Stil zwischen einfachen und doppelten Anführungszeichen, außer aus stilistischen Gründen. Tools wie Prettier konvertieren alle Strings, um denselben Stil von Anführungszeichen zu verwenden. Wir haben Prettier so konfiguriert, dass es standardmäßig doppelte Anführungszeichen verwendet. Ein Grund, einen Stil von Anführungszeichen gegenüber einem anderen zu bevorzugen, besteht darin, wenn ein Anführungszeichen Teil des Strings ist:
>
> - `"It's such a nice day!"`
> - `'"Nice work", they said.'` oder `'[data-js="foo"]'`
>
> Prettier erkennt diese Fälle automatisch.

Strings können miteinander verknüpft werden, indem der `+`-Operator verwendet wird (ja, derselbe wie der mathematische Operator). Dies wird **String-Konkatenation** genannt:

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

Template Literals

Die dritte Methode, um Strings zu schreiben, hat die nützliche Eigenschaft, dass du Variablen in den String einfügen kannst, indem du Platzhalter mit einem Dollarzeichen und geschweiften Klammern `${}` umschließt. Dies wird auch **String-Interpolation** genannt.

Auf diese Weise musst du nicht mehrere Strings verketten, wenn du eine Variable in deinem String verwenden möchtest:

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

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

Jeder **Ausdruck(expression)** kann in diese Platzhalter eingefügt werden:

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

Mit Template Literals kannst du auch **mehrzeilige Strings(multi-line strings)** schreiben:

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

String-Eigenschaften und -Methoden

Strings in JavaScript haben einige eingebaute **Eigenschaften(properties)** und Funktionalitäten, die als **Methoden(methods)** bezeichnet werden. Du kannst sie mit der Punktnotation gefolgt vom Namen der Eigenschaft / Methode aufrufen.

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

> 💡 Methoden sind Funktionen, daher müssen sie durch das Setzen von `()`-Klammern nach dem Namen der Methode aufgerufen werden.

Property / Method	Effect
`.length`	gibt die Anzahl der Zeichen in einem String zurück.
`.toUpperCase()`	gibt eine Version des Strings in Großbuchstaben zurück.
`.toLowerCase()`	gibt eine Version des Strings in Kleinbuchstaben zurück.
`.trim()`	gibt einen String ohne Leerzeichen am Anfang und Ende zurück.
`.replaceAll(oldString, newString)`	ersetzt alle Vorkommen von `oldString` durch `newString`.
`.startsWith(subString)`	gibt `true` zurück, wenn der String mit subString beginnt.
`.endsWith(subString)`	gibt `true` zurück, wenn der String mit subString endet.
`.includes(subString)`	gibt `true` zurück, wenn der String das subString enthält.

> 💡 Besuche die MDN-Dokumentation für noch mehr String-Methoden.

Eingabefelder

Jedes Eingabefeld in HTML enthält einen **Wert(value)** in Form eines Strings. Du kannst den Wert abrufen, indem du `.value` auf das Eingabeelement anwendest:

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

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

textInput.value; // ergibt 'test 123'
numberInput.value; // ergibt '42' (immer noch ein String!)
```

Du kannst den Wert des Eingabefelds auch ändern, indem du diesem Eingabeeigenschaft einen neuen Wert zuweist:

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

Diese Änderung ist sofort auf der Website sichtbar.

Zum Beispiel kannst du alle Großbuchstaben in einem Formular durch die Kombination dieser Funktionalität mit einem `input`-Event-Listener auf dem Eingabeelement erzwingen:

```
// transformiert bei jeder Änderung den Eingabewert in Großbuchstaben
textInput.addEventListener("input", () => {
  const oldValue = textInput.value;
  const newValue = oldValue.toUpperCase();
  textInput.value = newValue;
});
```

Ressourcen

String-Methoden

MDN Docs: String Methods