Inhaltsverzeichnis

Objektorientierte Programmierung mit JavaScript

Ein Beispieltext zur Demonstration der Syntaxdarstellung und der Syntaxbearbeitung im Visuellen Editor.

Namespace

Ein Namespace ist ein Container in dem Entwickler Funktionalitäten unter einem eindeutigen, applikationsspezifischen Namen zusammenfassen können. **In JavaScript ist ein Namespace ein gewöhnliches Objekt, welches Methoden, Eigenschaften und Objekte enthält.**

Im Gegensatz zu manchen anderen objektorientierten Programmiersprachen gibt es in der Sprachebene von JavaScript keinen Unterschied zwischen einem regulären Objekt und einem Namespace.

Die Idee hinter der Erstellung eines Namespaces in JavaScript ist simpel: es wird ein globales Objekt erstellt, welches alle Variablen, Methoden und Funktionen als Eigenschaften besitzt. Zusätzlich kann die Verwendung von Namespaces Namenskonflikten in der Applikation vorbeugen.

Es wird ein globales Objekt names MYAPP erstellt:

```
1 // global namespace
2 var MYAPP = MYAPP || {};
```

Im obigen Code wird zuerst geprüft, ob MYAPP bereits definiert wurde (entweder in derselben oder einer anderen Datei). Wenn MYAPP bereits definiert wurde, wird das globale Objekt MYAPP verwendet. Anderenfalls wird ein leeres Objekt names MYAPP erstellt, welches später Methoden, Funktionen, Variablen und andere Objekte kapseln kann.

Innerhalb eines Namespaces können weitere Namespaces erstellt werden:

```
1 // sub namespace
2 MYAPP.event = {};
```

Der folgende Code erstellt einen Namespace und fügt diesem Variablen, Funktionen und Methoden hinzu:

```
 1 // Create container called MYAPP.commonMethod for common method and properties
 2 MYAPP.commonMethod = {
 3   regExForName: "", // define regex for name validation
 4   regExForPhone: "", // define regex for phone no validation
 5   validateName: function(name){
 6     // Do something with name, you can access regExForName variable
 7     // using "this.regExForName"
 8   },
 9
10   validatePhoneNo: function(phoneNo){
```

```
11      // do something with phone number
12    }
13 }
14
15 // Object together with the method declarations
16 MYAPP.event = {
17      addListener: function(el, type, fn) {
18      //  code stuff
19      },
20     removeListener: function(el, type, fn) {
21      // code stuff
22     },
23     getEvent: function(e) {
24     // code stuff
25     }
26
27     // Can add another method and properties
28 }
29
30 //Syntax for Using addListner method:
31 MYAPP.event.addListener("yourel", "type", callback);
```

Quelle dieses Beispieltextes: https://developer.mozilla.org/de/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript