Inhoud

Object-oriented Programming with JavaScript

A sample text to demonstrate syntax representation and syntax editing in the Visual Editor.

Namespace

A namespace is a container in which developers can combine functionalities under a unique, application-specific name. **In JavaScript, a namespace is a common object that contains methods, properties, and objects.**

Unlike some other object-oriented programming languages, there is no difference in the JavaScript language level between a regular object and a namespace.

The idea behind creating a namespace in JavaScript is simple: create a global object that has all variables, methods, and functions as properties. In addition, the use of namespaces can prevent naming conflicts in the application.

To create a global object called MYAPP:

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

The above code first checks if MYAPP has already been defined (either in the same or a different file). If MYAPP has already been defined then the global object MYAPP will be used. Otherwise, an empty object called MYAPP is created, which later can encapsulate methods, functions, variables and other objects.

Within a namespace, additional namespaces can be created:

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

The following code creates a namespace and adds variables, functions, and methods to it:

```
// Create container called MYAPP.commonMethod for common method and properties
MYAPP.commonMethod = {
  regExForName: "", // define regex for name validation
  regExForPhone: "", // define regex for phone no validation
  validateName: function(name){
    // Do something with name, you can access regExForName variable
    // using "this.regExForName"
  },
```

```
  validatePhoneNo: function(phoneNo){
    // do something with phone number
  }
}

// Object together with the method declarations
MYAPP.event = {
    addListener: function(el, type, fn) {
    //  code stuff
    },
   removeListener: function(el, type, fn) {
    // code stuff
   },
   getEvent: function(e) {
   // code stuff
   }

   // Can add another method and properties
}

//Syntax for Using addListner method:
MYAPP.event.addListener("yourel", "type", callback);
```

Source for this sample text: https://developer.mozilla.org/de/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript