Node.js-Tutorium: Hello World: Konsole

Wechseln zu:Navigation, Suche

Dieser Artikel erfüllt die GlossarWiki-Qualitätsanforderungen **nur teilweise**:

Korrektheit: 4 (großteils überprüft)	**Umfang**: 1 (zu gering)	**Quellenangaben**: 5 (vollständig vorhanden)	**Quellenarten**: 5 (ausgezeichnet)	**Konformität**: 5 (ausgezeichnet)

Node.js-Tutorium Hello World

Übersicht: Teil 1: Konsole | Teil 2: HTTP | Teil 3: TCP

Inhaltsverzeichnis

1 Use Cases

Es soll eine einfache Node.js-Anwendung erstellt werden, die `Hallo, Welt!` auf der Konsole ausgibt.

Anschließend soll der Benutzer nach einen Nanmen gefragt werden. Das Programm reagiert darauf mit der Begrüßung `Hallo, <BENUTZERNAME>!`. Dieser Vorgang wiederholt sich solange, bis der Benutzer `Ende` als „Namen“ eingibt.

2 Node.js installieren

Für die Realisierung von Node.js-Projekten benötigt man außerdem auf jeden Fall eine Konsole, am Besten eine Unix-Konsole wie die Bash. Unter Linux und Mac gibt es eine derartige Konsole bereits, unter Window kann man die „Git BASH“ verwenden:

MsysGit: http://msysgit.github.io/

Des weiteren benötigt man natürlich Node.js:

Node.js: http://nodejs.org/

2.1 Anmerkungen zur Git-Bash unter Windows

In der Git-Bash können Sie mit den Cursortasten „nach oben“ und „nach unten“ zwischen den schon eingebenen Bash-Befeheln hin und her blättern.

In die Git-Bash können Bash-Befehle allerdings nur etwas umständlich per Copy und Paste eingefügt werden:

Bash-Befehl per `Strg-C` aus dem Tutorial kopieren.
Klick auf das kleine Icon in der linken oberen Ecke des Bash-Fensters
Klick auf `Bearbeiten`
Klick auf `Einfügen`

Auch beim Aufruf des Linux-Editors `vi` von der Bash-Konsole aus muss man diesen umständlichen Weg gehen. Wenn allerdings die Node.js-Konsole läuft (siehe nächsten Abschnitt), geht es etwa einfacher:

JavaScript-Befehl per `Strg-C` kopieren.
Rechtsklick in die Node.js-Konsole
Klick auf `Einfügen`

3 Node.js in der Bash ausführen

Öffnen Sie ein Bash-Konsolfenster und starten Sie Node.js, indem Sie den Befehl `node` eintippen:

```
$ node
```

Es öffnet sich die Node.js-Konsole. Tippen Sie in diese Konsole `"Hallo, Welt!"` ein:

```
> "Hallo, Welt!"
```

Das Programm antwortet mit der Ausgabe:

```
'Hallo, Welt!'
```

Tippen Sie nun folgende Befehle ein:

```
> var benutzer = "Wolfgang"
> 'Hallo, ' + benutzer + '!'
```

Das Programm antwortet nach dem ersten Befehl mit der Ausgabe:

```
undefined
```

und nach dem zweiten Befehl mit der Ausgabe:

```
'Hallo, Wolfgang!'
```

3.1 Read–Eval–Print-Loop

Die Node.js-Konsole realisiert eine sogenannte Read–Eval–Print-Loop. Das bedeutet, solange die Node.js-Konsole läuft, führt sie immer wieder folgende drei Befehle aus:

1. Read: Einlesen, parsen und übersetzen der Benutzereingabe.
2. Eval: Ausführen (“evaluate”) des übersetzten Codes.
3. Print: Ausgeben des Ergebnises auf der Konsole.

Falls ein Befehl kein Ergebnis liefert, wird `undefined` als Ergebnis ausgeliefert. Das ist z.B. bei der Definition und Initialisierung der Variablen `benutzer` der Fall.

3.2 `console.log`

Mit Hilfe des Befehls `console.log()` ist es möglich, Informationen schon während der Eval-Phase auf der Konsole auszugeben.. Tippen Sie in diese `console.log('Hallo, Welt!')` ein:

```
> console.log('Hallo, Welt!')
```

Das Programm antwortet mit der Ausgabe:

```
Hallo, Welt!
undefined
```

Die erste Ausgabe erfolgt während der Eval-Phase, die zweite Ausgabe erfolgt während der Print-Phase und zeigt das Ergebnis des Aufrufs der Funktion `console.log` an.

3.3 Aufgaben

Was gibt der Befehl `console.log('Hallo, ' + benutzer + '!')` auf der Konsole aus in folgenden Fällen aus?

1. Der Befehl wird direkt im Anschluss an die obigen Befehle in die Node.js-Konsole eingegeben.
2. Der Befehl wird nach einem Neustart der Node.js-Konsole eingegeben. (Sie können die Konsole mit `Strg-C` beenden.)

Was geben folgende Befehle auf der Konsole aus?

1. `3+4`
2. `3+` gefolgt von der Return-Taste und anschließend `4`
3. `console.log(3+4)`
4. `console.log(3+4, 3*4)`
5. `(3+4).toString()`
6. `3+4 .toString()` (Das Leerzeichen ist wichtig!)
7. `3+4..toString()` (Alternative des vorherigen Befehls bei der das Leerzeichen entfällt)

Tipp: Auch in der Node.js-Konsole können Sie mit den Cursortasten „nach oben“ und „nach unten“ zwischen den schon eingegebenen Konsol-Befehlen hin und her blättern.

Sehr spannend sind auch die folgenden vier Befehle, wobei die Ergebnisse nur von JavaScript-Engine-Programmierern zu verstehen sind. Für jeden Normalbürger sind sie nicht nur überraschend, sondern auch noch vollkommen unintuitiv:

1. `[] + []`
2. `[] + {}`
3. `{} + []`
4. `{} + {}`

Übrigens reagiert hier jede JavaScript-Engine anders! Man gebe zum Vergleich die vier Befehle mal in die JavaScript-Konsole von Firefox ein.

4 Ausführen von Dateien

Gehen Sie mit Hilfe des Bash-Befehls `cd` in das Verzeichnis, indem Sie Ihre Dateien aus dem ersten Node.js-Tutorium ablegen. **Zum Beispiel**:

```
cd /C/web/node/tutorium/hello_world_console
```

Wenn es dieses Verzeichnis noch nicht gibt, müssen Sie es vorher erstellen:

```
mkdir -p /C/web/node/tutorium/hello_world_console
```

Legen Sie eine Datei namens `hello-world-console-01.js` an und schreiben Sie in diese Datei mit Hilfe Ihres Lieblings-Texteditors (für Bash-Benutzer ist dies selbstverständlich `vi`) folgende drei Zeilen (GitHub):

```
console.log('Hallo, Welt!');

var v_user = 'Wolfgang';
console.log('Hallo, ' + v_user + '!');
```

Mit dem Bash-Befehl `less` können Sie den Inhalt auflisten:

```
less hello-world-console-01.js
```

Und mit Hilfe des Befehls `node` können Sie den Inhalt der Datei übersetzen und ausführen:

```
node hello-world-console-01.js
```

Folgende Unterschiede zur Read-Eval-Print-Loop sind zu beobachten:

Alle Befehle werden auf einmal übersetzt.
Die einzelnen Befehle müssen daher durch Strichpunkte voneinander getrennt werden.
Die Ergebnisse einzelnen Befehle werden nicht auf der Konsole ausgegeben. Es werden nur die Ausgaben der `console.log`-Befehle in der Konsole angezeigt.

4.1 Benutzereingaben

Es ist auch möglich, Benutzereingaben über die Konsole zu tätigen.

Erstellen Sie eine Datei `hello-world-console-02.js` (GitHub) und fügen Sie folgenden Code ein:

```
// see
http://stackoverflow.com/questions/8128578/reading-value-from-consol
e-interactively
// see http://nodejs.org/api/readline.html

'use strict';

var v_readline = require('readline');
var v_rl       = v_readline.createInterface
                           ({ input:  process.stdin,
                              output: process.stdout
                           });

console.log('Hallo, Welt!');

v_rl.setPrompt('Name oder "Ende": ');
v_rl.prompt();
v_rl.on('line',
        function(p_input)
        { if (p_input === 'Ende' || p_input === '"Ende"')
          { v_rl.close(); }
          console.log('Hallo, ' + p_input + '!');
          v_rl.prompt();
        }
       )
    .on('close',
        function()
        { console.log('Servus!');
          process.exit();
        }
       );
```

Starten Sie nun dieses Programm mit Hilfe des Befehls

```
node hello-world-console-02.js
```

Sie werden nach einem Namen gefragt. Wenn Sie diesen eingeben, werden die zugehörige Person namentlich begrüßt und Sie werden nach einem weiteren Namen gefragt. Das Programm endet, sobald Sie `Ende` als Namen (oder `Strg-D`, aber das wird in den Uses Cases nicht gefordert) eingeben.

4.2 Ausführen des Programms in WebStorm

Sie können das Programm auch innerhalb von WebStorm ausführen, indem Sie WebStorm installieren und das Tutoriumsprojekt von GitHub herunterladen, die Datei `hello-world-console-02.js` öffnen und mittels „Run-Icon“ ausführen. Das „Run-Icon“ ist ein kleines grünes Dreieck, das Sie finden, wenn Sie im Projekt-Explorer mit der rechten Maustaste auf die Datei `hello-world-console-02.js` klicken.

4.2.1 Analyse des Programms

In `hello-world-console-02.js` wird das Readline-Modul von Node.js verwendet. Das Realine-Modul wird mit Hilfe des Node.js-Befehls `require` eingelesen. Das zugehörige Modulobjekt wird in der Variablen `v_readline` gespeichert. (Der Präfix `v_` des Variablennamens deutet dabei an, dass es sich bei `v_readline` um eine Modul**v**ariable handelt, auf die nur innerhalb des aktuellen Moduls `hello-world-console-02.js` zugegriffen werden kann; vgl. Multimedia-Programmierung: Style Guide.)

Das Laden des Readline-Moduls hat zur Folge, dass das Node.js-Programm nicht mehr automatisch beendet wird, nachdem alle Befehle einmal abgearbeitet wurden. Man muss das Programm also explizit beenden. Die kann entweder durch den Benutzer mittels `Strg-D` oder mit Hilfe eines Javascript-Befehls wie `process.exit()` erfolgen. Die globale Variable `process` enthält dabei ein Objekt, das dem Node.js-Programm Zugriff auf den aktuellen, durch den Befehl `node` gestarteten Prozess gestattet (siehe Node.js-Manual).

Das Objekt `process` stellt nicht nur Methoden wie `exit` zur Verfügung, sondern bietet auch Zugriff auf die Standard-Streams des Prozesses: `stdout` (Ausgabe), `stderr` (Ausgabe von Fehlermeldungen) und `stdin` (Eingabe).

Beispielsweise könnte `console.log` folgendermaßen definiert werden:

```
console.log = function(p_info)
              { process.stdout.write(p_info + '\n'); };
```

(Der Präfix `p_` deutet dabei an, dass es sich bei `p_info` um einen Funktiosnparameter handelt; vgl. abermals Multimedia-Programmierung: Style Guide.)

Mit Hilfe des Befehls `var v_rl = v_readline.createInterface(...)` wird ein Wrapper für die Standardstreams `stdin` und `stdout` erstellt und in der lokalen Variablen `v_rl` gespeichert. Das Readline-Interface bietet einige Methoden und Events, die über die Methoden und Events der Standardstreams hinausgehen. Im Hello-Beispiel werden die Methoden zum Setzen (`setPrompt`) und Schreiben (`prompt`) eines Promt-Textes sowie die Events zum Behalndeln von Benutzereingabe-Ereignissen (`line`) und von Stream-Enede-Eregnissen (`close`) eingesetzt.

4.3 Aufgabe

Implementieren Sie die Use Cases diese Tutoriums indem Sie kein Modul wie `Readline` benutzen, sondern direkt auf `stdin` und `stdout` zugreifen.

Musterlösung

5 Fortsetzung des Tutoriums

Sie sollten nun Teil 2 des Tutoriums bearbeiten.

6 Quellen

Kowarschick (MMProg): Wolfgang Kowarschick; Vorlesung „Multimedia-Programmierung“; Hochschule: Hochschule Augsburg; Adresse: Augsburg; Web-Link; 2018; Quellengüte: 3 (Vorlesung)
HOW TO NODE: Hello Node!

7 Siehe auch

Beispiele und Musterlösungen (auf GitHub)

Kategorien:
Node.js-Tutorium: Hello World
Node.js-Beispiele