BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Prof. Dr.-Ing. Kristian Hildebrand

Computergrafik 2 / WarmUp-Aufgabe
Funktionsobjekte in JavaScript

SoSe 2016

Diese kleine Warmup-Aufgabe soll Ihnen etwas Gelegenheit zum Kennenlernen von JavaScript als Programmiersprache geben und Sie mit der Browser-Entwicklungsumgebung vertraut machen.

Inhaltlich geht es darum, Ihnen das Konzept von Funktionen und Closures näherbringen. Sie werden sehen, wie man mit Hilfe von Closures zustandsbehaftete Funktionen ('Funktionsobjekte') schaffen kann.

Diese Aufgabe ist rein freiwillig und muss nicht abgegeben werden. Nutzen Sie die Chance - JavaScript und Closures sind wichtig und werden in der Klausur gefragt!

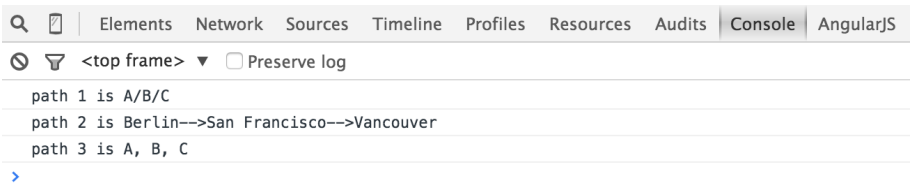

Abbildung 1: Ergebnisausgabe

Voraussetzungen Sie sollten zuerst das Merkblatt zur Entwicklungsumgebung gelesen und durchgearbeitet haben. Somit sollte ein funktionsfähiges cg2-Projekt vorliegen.

Aufgabe 1: Funktionsobjekte und Closures

In dem Verzeichnis *cg2/cg2-a00-warmup/* finden Sie in *main.js* einen Programmrumpf, der beim Laden Datei index.html im Browser aufgerufen wird. Durch das onLoad-Statement in index.html wird automatisch die *main()*-Funktion aufgerufen (später werden wir dies mit dem Framework require.js eleganter lösen).

Das Ziel der Aufgabe ist es, dass Sie die Funktion makePath() so implementieren, dass jeder Aufruf von *makePath()* ein neues Pfad-Objekt erzeugt.

Jede Instanz eines Pfad-Objekts soll eine Funktion mit einem optionalen Argument sein. Wenn beim Aufruf der Funktion ein Argument angegeben wird, so wird dieses als neuer Wegpunkt interpretiert, der hinten an den Pfad angefügt wird. Wenn kein Argument angegeben wird (also der Parameter undefined zurückliefert), soll der Pfad nicht verändert werden.

Die Pfad-Funktion soll immer als Ergebnis den aktuellen Pfad als String zurückliefern.

Beispiel und Test: Folgendes Beispiel / Test-Code ist bereits in *main.js* enthalten:

```
1
2 // the main() function is called when the HTML document is loaded
3 var main = function() {
4
5   // create a path, add a few points on the path, and print it
```

```
6   var path1 = makePath();
7
8   path1("A");
9   path1("B");
10  path1("C");
11
12  window.console.log("path 1 is " + path1() );
13
14 };
```

Sie sehen, dass in Zeile 6 ein neues Pfadobjekt angelegt wird. In Zeilen 8-10 wird dieses Objekt wie eine Funktion aufgerufen, um jeweils einen Wegpunkt hinzuzufügen. In Zeile 12 wird kein Wegpunkt hinzugefügt, sondern nur der Rückgabewert auf der Konsole ausgegeben.

Die Ausgabe dieses Tests könnte z.B. sein:

path1 is A, B, C

wenn Sie sich entscheiden, die Wegpunkte durch ', ' zu trennen.

Erweiterungen und Tests des Programms:

- Erlauben Sie, dass man bei *makePath()* optional einen beliebigen Trennstring angibt. Dieser soll immer zwischen zwei Wegpunkten eingefügt werden, aber nicht am Anfang oder am Ende des Pfades.
- Testen Sie mehrere Pfadobjekte mit verschiedenem Inhalt und Trennstrings, und stellen Sie sicher, dass jedes Pfadobjekte seinen eignen unabhängigen Zustand enthält.

Aufgabe 2: Tipps: Wie können Sie vorgehen?

Laden Sie index.html in Ihren Browser (dazu müssen Sie nicht den lokalen Webserver betreiben) und sehen Sie sich die Ausgabe des Programms in der Entwicklerkonsole an. Sie werden folgenden Fehler erhalten:

Uncaught TypeError: undefined is not a function

Finden Sie heraus, an welcher Stelle dieser Fehler auftritt und versuchen Sie zu ergründen, was diese Fehlermeldung bedeuten könnte. Wenn Sie durch Klicken auf die Fehlermeldung an die entsprechende Codezeile springen (je nach Browser verschieden), sehen Sie, wo der Fehler auftritt.

Sie sehen, dass *path1* wie eine Funktion aufgerufen wird; aber *path1* ist gar keine Funktion. –> Sie müssen also dafür sorgen, dass makePath() eine Funktion zurückliefert.

Diese Funktion benötigt einen Zustand, den Sie bei jedem Funktionsaufruf verändern und zurückliefern können. –> Dies können Sie über eine lokale Variable in der Closure der Funktion lösen. Siehe Beispiel im Handout.

Als Typ für diese Variable reicht ein String. Bei jedem Aufruf der Funktion können Sie Zeichen an den String anhängen (mit + oder +=).

Um zu testen, ob für einen Funktionsparameter ein Wert übergeben wurde oder nicht, vergleichen Sie den Parameter einfach mit dem Wert *undefined*.

Zum Lösen dieser kleinen Aufgabe benötigen Sie keinerlei fortgeschrittene Funktionalität von JavaScript und auch keinerlei Frameworks oder sonstige Zusatzfunktionen. Alle benötigten Konstrukte finden Sie im JavaScript-Handout.

Aufgabe 3: JavaScript - Laufzeitkonzept und Eventloop

Machen Sie sich im Folgenden mit dem JavaScript Eventloop bekannt. Lesen Sie dazu bitte `https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop` um das in der Vorlesung Be-

sprochene zu rekapitulieren. (Bitte in der englischen Version.) Der vorgegebene Source Code enthält weitere folgende Zeilen.

```
{
    window.console.log('This is the start.');

    // sets a timeout and calls the callbackFunction
    // after the timeout.
    // The specified callback is 0!!! milliseconds
    setTimeout(function callbackFunction() {
        window.console.log('This is a msg from call back.');
    }, 0);

    window.console.log('This is just a message.');
}
```

Erkären Sie warum die Ausgabe auf der Konsole in der folgenden Reihenfolge geschieht:

This is the start.

This is just a message.

This is a msg from call back.