Aufgabenblatt 4 – Entwurfsbeschreibung

Projekt: Keynode 11
Gruppe: swp11-5
Projektleiter: Florian Golemo
Dokumentverantwortlicher: Andy Wermke

Inhaltsverzeichnis

1 Allgemeines

Das Keynode-Projekt basiert konsequent auf Javascript-Bibliotheken und -Frameworks, sowohl client- als auch serverseitig. Für die Präsentationsfunktionalität kommt die Bibliothek **Slidy clientseitig** zum Einsatz. Sie sorgt sowohl für die Optik als auch für das Verhalten der Seite als Präsentation mit allen gewohnten Features.
Serverseitig wird **Node.JS** verwendet, eine Umgebung zur Ausführung von Javascript mit Serverfunktionalität. Natürlich kann das serverseitige Skript auch mit Node.JS lokal auf dem Präsentationsrechner ausgeführt werden. Zum dynamischen Nachrichtenaustausch zwischen **Client und Server** wird **Socket.IO** verwendet. Im Gegensatz zu den herkömmlichen AJAX-Requests ermöglicht diese Bibliothek eine persistente Verbindung zwischen Client und Server über die bidirektional Daten ausgetauscht werden können, was insbesondere für die Push-Nachrichten vom Server an die Clients (z.B. Befehl zum Anzeigen der nächsten Folie) benötigt wird.
Die vorgegebene Aufgabenstellung war nun, einen Satz Slidy-Folien online verfügbar zu machen und stets bei allen Teilnehmern der Präsentation die gleiche Folie anzuzeigen. Eine Aktualisierung soll dynamisch, also ohne erneutes laden der gesamten Seite, funktionieren.

2 Übersicht

Entsprechend der vorgegebenen Aufgabenstellung haben wir einen Webserver eingerichtet, der eine Slidy-Präsentation hostet.
Folgende Funktionen wurden implementiert:

- Beim Laden der Seite wird automatisch die aktuell ausgewählte Folie angezeigt (d.h. diejenige Folie, die zuletzt von einem Client angezeigt wurde).
- Schaltet einer der Besucher auf eine andere Folie um, so wird diese Information an den Server übertragen.
- Der Server sendet daraufhin den Befehl zum Umschalten an die übrigen Clients, deren Ansicht sich dann sofort aktualisiert.

3 Struktur des Gesamtsystems

Als Basis für die Entwicklung des **serverseitigen** Prototypen diente uns die **Socket-Chat-Demo**. Diese ist eine Beispielanwendung für die Socket-Funktionen, die für die Kommunikation zwischen Client und Server in der zentral gesteuerten Präsentation benötigt werden. Die Demo wurde reduziert und an unsere Bedürfnisse angepasst, sodass sie nun nicht mehr die Chatnachrichten verteilt, sondern Befehle an die Präsentationslogik.
Als Basis für den **clientseitigen** Prototypen kam natürlich **Slidy** zum Einsatz. Die benötigten Funktionen zum Initialisieren der Socketverbindung, dem Senden und Empfang von Folien-Steuerfunktionen wurden vorerst „hart“ in das Slidy-Objekt hinein geschrieben.
Diese Funktionalität soll in Zukunft aber „von außen“ in das Slidy eingefügt werden, ohne den Slidy-Code selbst verändern zu müssen, da dies gerade bei Updates des Slidy-Frameworks zu Problemen führen wird.
Unsere **Demo-Präsentation** basiert auf dem **AKSW-Template**.

Wichtig: die QS-Konzepte wurden in diesem Protoyp nur soweit umgesetzt, wie dies der jew. bisherige Implementierung (also der bestehende Slidy-/Co-Code) zugelassen haben.
Das heißt: Wenn z.B. bei Slidy die Std.-Einrückug 2 Leerzeichen beträgt, dann wurde das so gelassen.

4 Struktur der Pakete

4.1 Struktur Server

Der Server erfüllt zwei Funktionen. Zum einen erlaubt er den **HTTP-Zugriff auf die benötigten HTML-, JS- und CSS-Dateien**, auf der anderen Seite initialisiert er Socket.IO und wartet auf Verbindungen mit den Clients. Verbindet sich ein Client, so schickt der Server über Socket.IO ein JSON-Objekt, das die aktuelle Folie angibt. Anschließend wartet er auf Nachrichten des Clients. Sendet ein Client die Information, dass auf eine andere Folie geschaltet wurde, so **speichert** die Serveranwendung diese **Foliennummer** und **broadcastet** sie **an alle Clients**.

4.2 Struktur Client

Das Socket-Objekt für die Kommunikation mit der Serveranwendung ist im gesamten Javascript global ansprechbar, da es dem globalen Slidy-Objekt hinzugefügt wird. Des weiteren wird dem Slidy-Objekt eine Socket-Initialisierungsfunktion hinzugefügt, die die notwendigen Eventlistener setzt. Der Aufruf der Socket-Initialisierungsfunktion ist dabei momentan noch fest in die normale init()-Funktion von Slidy integriert. Auch in die Eventlistener für Tastatureingaben ist nun Serverkommunikation mit eingebaut. Diese wird in Zukunft auch aus dem Slidy-Code in ein eigenes Objekt ausgelagert.