WPL ASSIGNMENT 2

Name: Sanket Prabhu
Net id: srp140430

Questions 1-3 are implemented and direction to run the html file is given in readMe file.

4.

(a) (5 points) Should we place JavaScripts in the HTML head or body? Please Specify the necessary arguments for your choice.

A JavaScript script file has to be loaded completely before a web browser even begins on the next JavaScript file. The effect of this, if the JavaScript files are included at the top of the document, is that it will be a visual delay before the end user sees the actual page. This is completely avoided if you include the JavaScript files at the end of the document. Hence, according my choice the scripts are to be placed at the end of the document.

(b) (5 points) Which JQuery event allows for proper handling of JavaScript code present in the HTML document head section? Why is this event handling required?

Document Ready Event allows for JavaScript code to be present in the HTML document head section. It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.

All jQuery methods, are inside a document ready event to prevent any jQuery code from running before the document is finished loading, is the reason why this event handling is required/used.

c. (5 points) Provide example web application use cases that require:
· Asynchronous communication
· Synchronous communication
Synchronous communication

In traditional Web-based applications, a user input triggers a number of resource requests. Once the requests have been answered by the server, no further communication takes place until the user's next input. Such communication between client and server is known as *synchronous communication*.

the open-source Apache AXIS toolkit to take care of the Web Services mechanics. We do not want this to be an exercise in Java development and chose this tool to abstract the complexity of designing and implementing a synchronous Web Service.

Here is an example of traditional synchronous communication passing between a browser and a Web server:

1. The user clicks a UI control in a browser-based web application.

2. The browser converts the user's action into one or more HTTP requests and passes them along to the Web-application server.

3. The application server responds to the user's requests by returning the requested data to the user. At this point the application is updated and the synchronous communication loop is complete. A new synchronous communication loop will begin when the user next clicks a UI control in their browser.

Asynchronous communication

There are three main types of asynchronous request and response sequences: *push, poll.* When building a test script for a web-based application, it is essential that you understand which type of asynchronous communication is in use.

Polling Communication Model

Comet is a web application model in which a long-held HTTP request allows a web server to push data to a browser, without the browser explicitly requesting it. Comet is anumbrella term, encompassing multiple techniques for achieving this interaction. All these methods rely on features included by default in browsers, such as JavaScript, rather than on non-default plugins. The Comet approach differs from the original model of the web, in which a browser requests a complete web page at a time.

Push Communication Model

We present a communication and component model for push systems. Surprisingly, despite the widespread use of many push services on the Internet, no such models exist. Our communication model contrasts push systems with client-server and event-based systems. Our component model provides a basis for comparison and evaluation of different push systems and their design alternatives. We compare several prominent push systems using our component model. The component model consists of producers and consumers, broadcasters and channels, and a transport system. We detail the concerns of each of these components. Finally, we discuss a number of open issues that challenge the widespread deployment of push or any other system on an Internet-wide scale. Payment models are the most important among these and are not adequately addressed by any existing system. We briefly present the payment approach in our Minstrel project.