LAB 5: Ajax functionality

Contents

Introduction

In this lab you will practice using the HTML 5 history API and building real-time apps with SignalR.

The lab consists of two tasks.

Resources

- Lecture notes and sample code from Module Website
- Visual Studio and Firefox/Firebug (or any combination of editor/browser)
- JavaScript reference, e.g. **https://developer.mozilla.org/en/JavaScript/Reference**
- Other references as required, e.g. jQuery documentation.

Solutions

Solutions will be available from the Module Website.

Task 1: A multi-part form

In this task you will create a page which contains a multi-part form, in which each part is a state of a single page application. You will use the HTML 5 history API to allow navigation between parts using links in the page and using the browser back/forward buttons.

1. Create a web page containing the following HTML. You can create this within a Visual Studio project or within any suitable environment.

```
<p class="formnav">
    <a id="personal" href="personal">Personal details</a>
    <a id="payment" href="payment">Payment details</a>
    <a id="summary" href="summary">Summary</a>
</p>
<div id="pers">
    <h3>
        Personal details</h3>
    <p>
        Full name
        <input type="text" id="fullname" /></p>
    <p>
        Address
        <input type="text" id="address" /></p>
</div>
<div id="pay">
    <h3>
        Payment details</h3>
    <p>
        Name on card
        <input type="text" id="cardname" /></p>
    <p>
        Card number
        <input type="text" id="cardno" /></p>
</div>
<div id="sum">
    <h3>
        Summary</h3>
    <p id="alldetails">
    </p>
    <p>
        <input type="button" value="Submit" /></p>
</div>
```

2. Add a script element to your page. Create a function called *swapForm* which sets the state of the page depending on a parameter, the value of which can be one of *personal*, *payment* or *summary* – the three *href* values in the anchor tags in the HTML.

   Changing the state of a page in a single-page application will often involve retrieving server-side data to populate the required state. In this simple case, changing the state simply involves setting the appropriate div element to be

visible and hiding the others. You can use jQuery for DOM manipulation.When changing to the summary state you should populate the paragraph in the summary state with the data entered in the text boxes in the other states. The parameter which this function receives will be the a full URL – you can retrieve the part which refers to the state using:

```
var state = href.split("/").pop();
```

3. Add a function called *addClicker*, which takes a single parameter. This parameter, which you should call *link*, will be a reference to an anchor tag. In this function, use *addEventListener* to add a listener to the click event of the object. The function defined as the listener should take a single parameter, which will be the event object – call this parameter *e*. This method should:

    - Call *swapForm*, passing in *link.href* as the parameter
    - Add a state to the *history* object using *pushState*. This example will work if you set only the URL (third) parameter to *link.href*
    - Call the *preventDefault* method of the event object – this will prevent the browser attempting to load a new page when the link is clicked

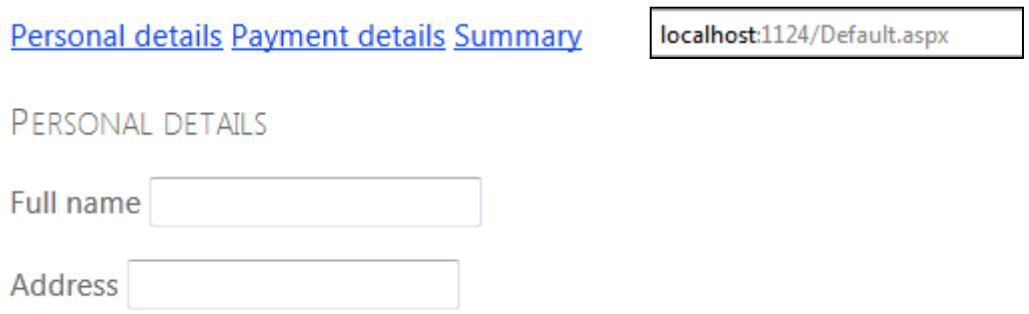
4. Add the following function which adds event listeners to the three anchor elements:

```
function setupHistoryClicks() {
    addClicker(document.getElementById("personal"));
    addClicker(document.getElementById("payment"));
    addClicker(document.getElementById("summary"));
}
```

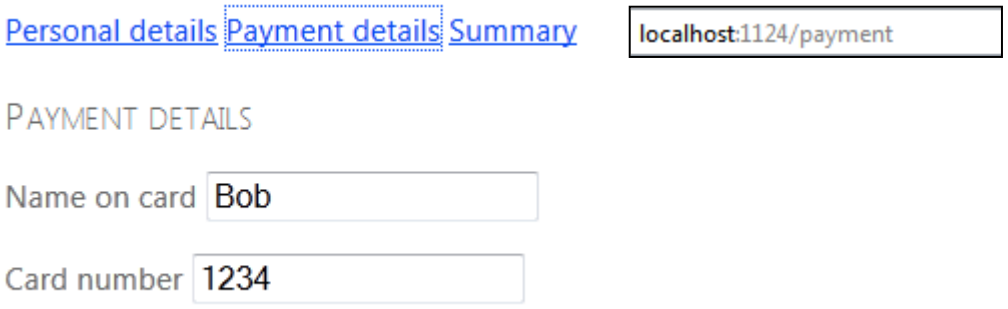
5. Add a function which runs when the document is loaded. This function should set the initial state of the page and call *setupHistoryClicks*. It should then use the following code to add a listener to the window for the *popstate* event, which will be fired whenever the URL changes, for example when the user clicks the browser's back button. This listener will read the current URL (*location.pathname*) in the address bar, which will contain a segment which identifies which state should be displayed, and will pass the URL to *swapForm* to change the state.

```
window.setTimeout(function () {
       window.addEventListener("popstate", function (e) {
           swapForm(location.pathname);
       }, false);
   }, 1);
```

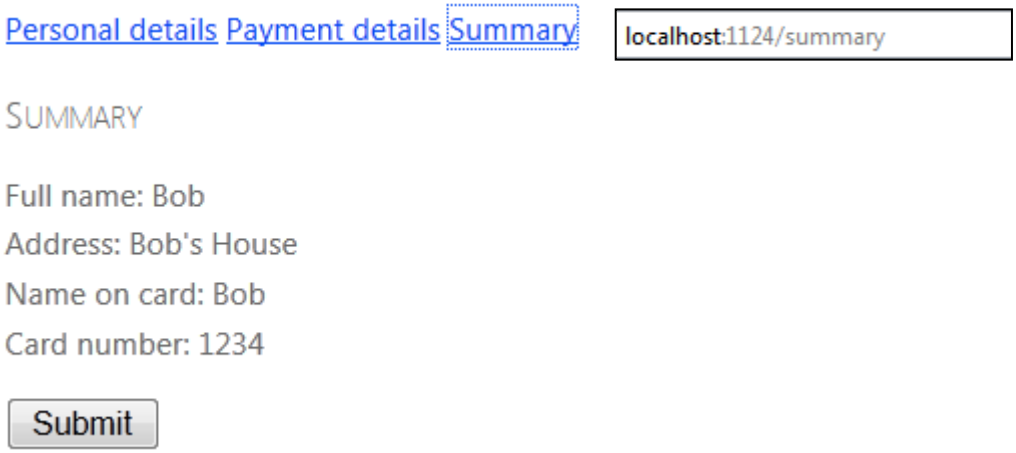
6. Test your page in a browser which supports HTML 5 history.. On loading the page only the Personal details form should be shown. The URL in the address bar will be something like the value shown:



Clicking Payment details (after entering some data) hides the Personal details form and shows the Payment details form. The URL changes, as shown:



Clicking Summary shows the summary form and the URL changes again



Clicking the browser's Back button shows the Payment details form and URL.

Clicking the browser's Forward button shows the Summary form and URL.

Note - this example is a variation of the code which you can see at: http://diveintohtml5.info/history.html. The code explanation there may be useful.

Task 2: Extending a chat application

In this task you will make some modifications to a simple chat app which has been built using SignalR.

1. Download and unzip *Mvc3SignalR.zip*, and open the Visual Studio project which it contains. This project includes the simple chat application shown in the lecture. In this app, when a user enters a message, the message is immediately displayed on all connected clients.

2. Modify the chat app so that:

   - The most recent five messages from all clients are displayed on all clients
   - Each message has a username included in its text. This username can simply be a value entered in a text box which you can add to the client page.

   You can look at the code for the Ticket app, also included in this project, which may help.

3. Test your application. Look at the behaviour of the application using the Net tab in Firebug. Can you tell what transport has been negotiated? Look at the response to each request to see how the call to the client side code is communicated.