JavaScript

3. Events, Forms and Servers

Practice

Alan Rodas Bonjour

Part 1: This part can be done in visual studio code, using both Node.js and a web browser for testing.
Parte 1:Esta parte puede ser realizada en visual studio code, utilizando tanto Node.js como un navegador web para verificar el código.

1) Write a web server using Node.js, add express and serve a simple HTML page.
Escriba un servidor web utilizando Node.js, agregue express y sirva una página HTML simple.

a) Express can serve multiple static files instead of one at a time through the "express.static" function. See https://expressjs.com/en/starter/static-files.html for more information. Use express.static to serve the files in a particular folder and test that you can serve multiple HTML files.
Express puede servir multiples archivos de forma estática en lugar de uno a la vez utilizando "express.static". Vea https://expressjs.com/en/starter/static-files.html para más información. Use express.static para servir archivos de una carpeta en particular y pruebe que puede acceder a todos los archivos HTML de la carpeta.

b) Write a form in your HTML page with a simple text input and a submit button. The method should be GET, and the action should point to a particular endpoint in your server. In your server create the endpoint that receives the information, and that prints the text input value upon action.
EscribE un formulario en tu HTML con un simple input de texto y un botón de submit. El método debe ser GET, y la acción debe apuntar a un endpoint en el servidor. En tu servidor crea el endpoint que recibe la información del formulario, y que imprima el valor del cuadro texto en la impresión.

c) Now change the previous form to use POST. Note that you are required to use the body-parse package in order to parse the body and access the information sent by the client.
Ahora cambia el formulario anterior para que use POST. Notar que se requiere el uso del paquete body-parse para que se pueda leer el body y se acceda correctamente a la información enviada por el cliente.

d) Let's do something more fun. Create a form that includes an email and a password fields, along with a submit button. Upon sending the form, the username and password should be validated (We will consider these two users only: admin@page.com with password "1234" and user@page.com with password "0000"). If the username and password are invalid we should present the same page again (we will leave the error message for later), if not, the user should be redirected to the "/dashboard" page. To redirect you may use "res.redirect(path)" method.
Vamos a hacer algo más divertido. Cree un form que incluya campos para un email y una contraseña, junto con un botón de envío. Cuando se envíe el formulario, el usuario y la contraseña deben ser validados (Vamos a considerar solo dos usuarios, admin@page.com con contraseña "1234" y user@page.com con contraseña "0000"). Sí el usuario y la contraseña son inválidos debemos presentar la misma página

nuevamente (dejaremos el error para después), sino, debemos redireccionar al usuario a la página de "/dashboard". Para redireccionar se puede usar el método "res.redirect(path)".

e) Now let's make the error message appear. For this we are going to use a simple trick: instead of presenting the same page, let's redirect to a page that is practically similar, but that has the error message alert.
Ahora vamos a hacer aparecer el mensaje de error. Para esto vamos a recurrir a un truco simple: en lugar de presentar la misma página, vamos a redireccionar el usuario a una página similar pero que contenga un alert con el mensaje de error.

f) Now let's make the dashboard different for the different users. Although this is usually done through dynamic HTML pages, we are going to use different HTML pages for the different users. The dashboard should present a welcome message with the username.
Ahora vamos a hacer que el dashboard muestra diferente información para diferente usuario. Aunque esto es generalmente realizado mediante páginas HTML dinámicas, vamos a usar dos páginas diferentes, una por usuario. El dashboard debería presentar un mensaje de bienvenida con el nombre de usuario.

2) Let's start a new server from scratch. Now we are going to perform the same as before with a login, but using AJAX. Let's make the main HTML page contain two elements, the "error-message" div is hidden by default (using CSS) and should be shown when the login fails (instead of redirecting, we should present the error message). Another, "main-content" contains all contents, and should be overwritten with the results if the message succeeds. Easy enough, but how to know if there was a successful login or not? We need to instruct the server and client to send/check for the HTTP status code of the response. A status of 200 is success, while a status of 401 means "unauthorized". See for more information https://expressjs.com/en/4x/api.html#res.status.
Vamos a comenzar un nuevo servidor desde cero. Esta vez vamos a realizar lo mismo que en el ejercicio anterior de login, pero usando AJAX. Vamos a hacer que la página principal contenga dos elementos, un div "error-message" se presenta oculto por defecto (usando CSS) y se debe mostrar cuando el login falla (en lugar de redirigir al usuario, mostramos el mensaje). Otro "main-content" contiene todo el contenido de la página, y se debe sobreescribir con los resultados sí el login tiene éxito. Suficientemente sencillo, pero ¿Cómo saber sí el login terminó con éxito o no? Para ello se debe instruir al servidor y el cliente de responder con/analizar el código de estado HTTP en la respuesta. Un estado de 200 significa que fue exitoso, mientras que un estado de 401 significa "no autorizado". Vea para más información https://expressjs.com/en/4x/api.html#res.status.

3) Let's add a validation factor to our form. We want to present a "captcha" like element and only send the form if the user inputs a correct response in the captcha.
Vamos a agregar un factor más de validación a nuestro formulario. Queremos presentar un elemento similar a un "captcha" y solo permitir el envío del formulario cuando el usuario ingresa los datos correctamente.

a) First let's create an endpoint in the server that returns a random number between 1 and 30. It should respond text only when GET is asked to the endpoint.
Primero vamos a crear un endpoint en el servidor que devuelva un número aleatorio entre 1 y 30. Debe responder como olo texto, cuando se accede mediante GET.

b) Now let's create a function in the client's JS that aks for two numbers (two requests to the server) and presents in a label in a new input at the HTML such as "What's the sum between <first number> and <second number>" where <first number> is the first element returned by the server, and <second number> is second number returned by the server.
Ahora creemos una función en JS que pida al servidor dos números (dos requests al servidor) y presente un label asociado a un nuevo input en el HTML con la forma "¿Cuál es la suma entre <primer número> y <segundo número>?" donde <primer número> es el primer número que devolvió el servidor y <segundo número> el segundo.

c) Let's call that function every 10 seconds. For this we can use the window function setInterval which works similarly to setTimeout, receiving a callback, but executing the callback every number of milliseconds passed. The function should also be called when the page is loaded.
Ahora vamos a llamar a la función cada 10 segundos. Para ellos podemos usar la función de window setInterval, que funciona de forma similar a setTimeout, recibiendo un callback, y ejecutando el callback cada cierta cantidad de milisegundos. La función también se debe llamar cuando la página se carga.

d) Finally we need to hook on the submit logic. If the input of the new text is not the number that matches the sum, the submit should be canceled and an error presented in an alert to the user. Else, the submit logic should continue.
Ahora vamos a colgarnos de la lógica de envío. Sí el nuevo input no contiene un número que coincida con la suma, entonces el envío debe cancelarse y se debe mostrar un error al usuario. Caso contrario debe enviarse el formulario normalmente.

e) Note that you will to store locally the numbers received, transform them into numbers and not text, and do other additional logic in order to operate the previous behavior. Finally, think about it... is the server endpoint really needed? What should the server do to validate the values?
Note que deben almacenarse localmente los números recibidos, transformarlos a número y no texto, y realizar otra lógica adicional para operarlos y así lograr el comportamiento antes mencionado. Finalmente se pide que piense al respecto... ¿Realmente deben estar los endpoints en el server? ¿Qué debería hacer el server para validar dichos valores?