

**JavaScript**

4. Cookies, Complex Types and JS responses

**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. We are going to continue our login with AJAX, but now we are going to add cookies to the mix and JS responses. We are going to use two pages, the login page and the dashboard page.  
   Vamos a continuar nuestro login con AJAX, pero ahora vamos a agregar cookies a la mezcla y respuesta con JS. Vamos a usar dos páginas, la de login y la de dashboard.
   1. As of now, when you attempt to access the dashboard of a user, it doesn’t matter if you have logged in or not, you can always access. Now we are going to make it so the endpoint for the dashboard verifies if a cookie is present, if it’s not, it will redirect the user to the login page. If present, it will serve the dashboard page for that user.  
      Por ahora, cuando se intenta acceder al dashboard de un usuario siempre se puede acceder, independientemente de sí se está logueado o no. Ahora vamos a hacer que el endpoint para acceder al dashboard verifique sí una cookie está presente. Sí no está presente, entonces el usuario es redireccionado a la pantalla de login. Caso contrario, se muestra el dashboard del usuario.
   2. Change the endpoint for login to return a simple object that has either an error message (if the login fails) or the username (if the login succeeds). Make the client receive such response and present the error as an alert or load the dashboard page (by requesting the dashboard page to another endpoint).  
      Cambie el endpoint del login para retornar un simple objeto javascript que contenga o bien un mensaje de error (sí el login falla) o bien el nombre de usuario (sí el login tiene exito). Haga que el cliente reciba dicha respuesta y presente el error como un alert o cargue la página de dashboard (pidiendosela a otro endpoint).
   3. When the login succeeds we need to store a cookie with the username logged in and the timestamp of such login. This should be sent with every request to the server if present.  
      Cuando el login tiene exito necesitamos almacenar una cookie con el nombre de usuario y el timestamp de dicho login. Luego debemos enviar dicha cookie con cada request que se realiza al servidor.
   4. We now also want a logout button in the dashboard. The logout logic should remove the cookie, and redirect the user to the login page.  
      Ahora vamos a agregar un botón de logout a nuestro dashboard. El botón de logout debería eliminar la cookie y redireccionar al usuario a la página de login.
   5. Finally, we need to add some logic when the login page is loaded. If the cookie is present, we need to automatically redirect to the dashboard. Else, we stay on the page.  
      Finalmente vamos a agregar algo de lógica a la página de login cuando carga. Sí la cookie está presente debemos redireccionar al usuario automáticamente a la página del dashboard, sino, debemos mantenernos en la página de login.
2. Let’s add a purpose to our dashboard. We would like to play a game. When the dashboard loads, it should present the text of an event, and ask about the date of the same. The user may input the date, and if it’s correct, then points should be awarded.  
   Vamos a agregar algún propósito a nuestro dashboard. Queremos poder jugar un pequeño juego. Cuando el dashboard cargue, debería mostrar el texto de un evento y luego preguntar por la fecha del mismo. El usuario debería poder ingresar la fecha y, si acertó, entonces obtendrá puntos.
   1. First let’s add an endpoint that return a random event name and date. Create an array of ten events, where each event is an object that contains a name (string) and a date (Date type). e.g. [ { name: “Independence day”, date: Date.parse(‘1776-07-04’) }, ... ]. On accessing the endpoint select one of the elements randomly and return it.  
      Primero vamos a crear un endpoint que retorne un evento aleatorio con nombre y fecha. Crea un array de diez eventos, donde cada evento es un objeto que contiene nombre (string) y fecha (Date). e.g. [ { nombre: “Día de la independencia”, fecha: Date.parse(‘1816-07-09’) }, ...]. Al acceder al endpoint se selecciona uno de los elementos de forma aleatoria y se retorna.
   2. Now add an input with type date, and a label for it, along a “check” button in your dashboard page. When the dashboard loads, access the previous endpoint and retrieve an event. Update the label with the event name. Do this in a function that you can call whenever is needed.  
      Ahora vamos a agregar un input de tipo fecha y una etiqueta para el mismo, junto con un botón de “verificar”, a nuestra página de dashboard. Cuando el dashboard carga, se accede al endpoint anterior y se recupera la información del evento. Actualiza el texto de la etiqueta con el nombre del evento. Haz esto en una función que puedas reutilizar cuando necesites.
   3. When the user attempts to verify the date, we need to award point depending on how close it was on his answer. We award one point if the year matches, 1 if the month matches, 1 if the day matches, and 2 extra points if all of them matches. The total number of points awarded should be shown in the screen at some text element.  
      Cuando el usuario intente verificar la fecha, necesitamos otorgar puntos dependiendo de que tan acertada sea su respuesta. Otorgamos 1 punto sí coincide el año, 1 punto sí coincide el mes, 1 punto sí coincide el día, y 2 puntos extra sí coinciden los 3.
   4. Now, after adding the points, we should present a different event to the user, so he can keep on playing. Reload the event by asking the endpoint for a new event.  
      Ahora luego de otorgar los puntos debemos presentar un nuevo evento al usuario, para que pueda seguir jugando. Recarga el evento pidiendole uno nuevo al endpoint.