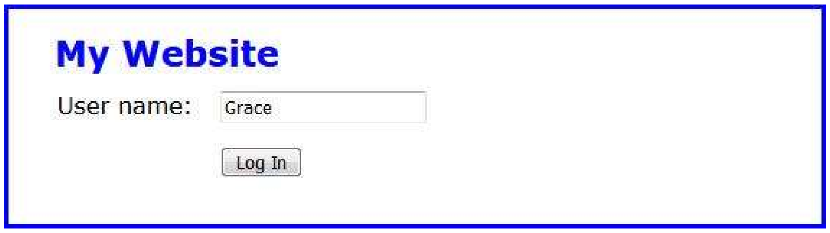
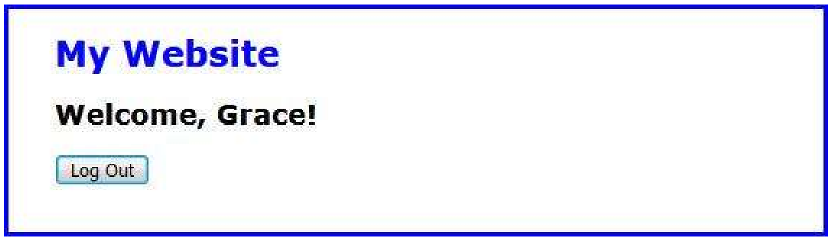
Extra 14-1 Navigate between pages and use a cookie

In this exercise, you'll develop an application that allows you to log in, save the user data in a cookie, navigate to a new page, and log out.

The interface looks like this initially:



And the interface looks like this after you've logged in:



NOTE: When you run this application from the file system, it doesn't work in the Chrome browser, but it should work in the Firefox browser.

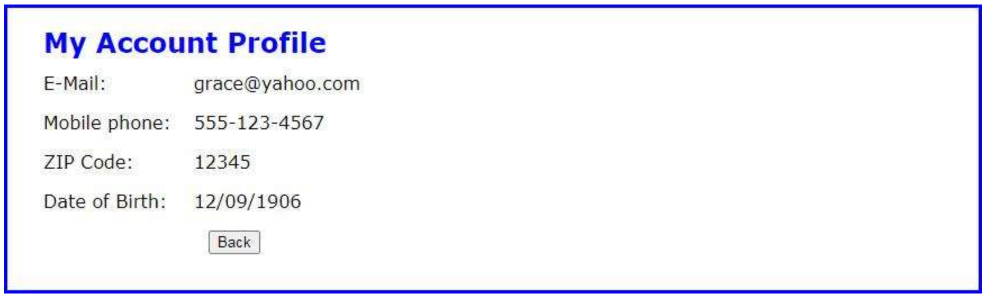
1. Open the application in this folder:

   `exercises_extra\ch14\login\`

   Then, run the application to see the user interface shown above.
2. Review the code in the login.js file. Note that it contains starts for four functions: getCookieByName(), setCookie(), deleteCookie(), and goToPage(). The getCookieByName() function returns an empty string, while the others contain no code.
3. Review the index.html file and notice that its embedded JavaScript code uses the functions in the login.js file.
4. Review the login.html file and notice that its embedded JavaScript code uses the functions in the login.js file.
5. In the login.js file, update each of the functions so they perform the tasks described by their names. Use the examples in figures 14-5 through 14-7 as a guide for the first three, and figure 14-1 for the last one.
6. Run the application, enter a user name, and click Log In. When the login.html page displays, press F12 to open the developer tools and display the Storage panel (in FireFox) to view the cookies for this application. This should show the cookie for this application.
7. Click on the Log Out button. When the index.html page is displayed, use the Storage panel of the developer tools to view the cookies for this application. This shouldn't show any cookies.

Extra 14-2 Navigate between pages and use session storage

In this exercise, you'll enhance the Account Profile application to save its data in session storage, navigate to a new page, and allow you to navigate back to the original page. When you click on the Save button, a new page gets the data from session storage and displays it like this:



My Account Profile

E-Mail:	grace@yahoo.com
Mobile phone:	555-123-4567
ZIP Code:	12345
Date of Birth:	12/09/1906

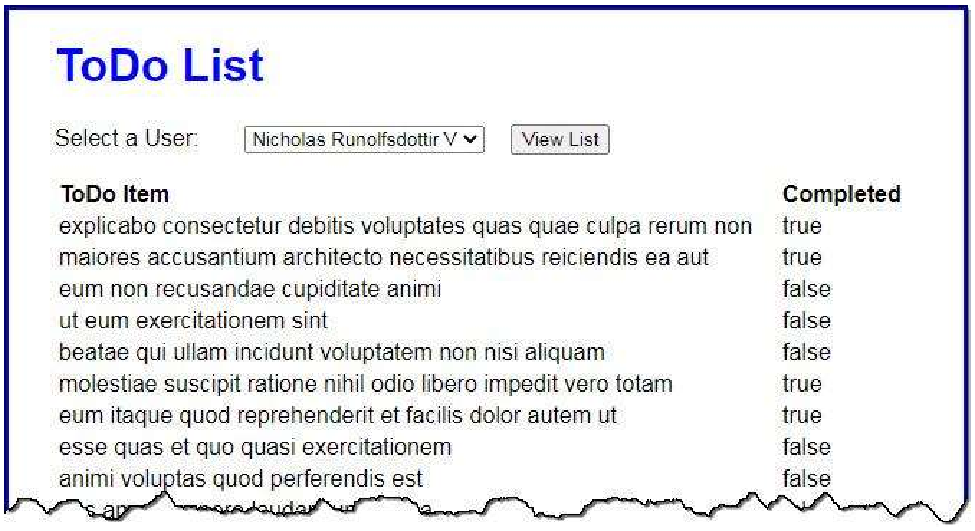
1. Open the application in this folder:

   `exercises_extra\ch14\profile\`

2. In the save_profile.js file, find the code in the handler for the click event of the Save button that validates the user entries. Then, find the if statement that checks the value of the isValid variable.
3. Add code to the if statement that saves the values in the email, phone, zip, and dob constants to session storage. Then, add code that uses the location object to navigate to the profile.html file.
4. Review the code in the display_profile.js file. Note that it contains the jQuery ready event handler and a handler for the click event of the Back button.
5. In the ready event handler, add code that retrieves the profile information from session storage and displays it in the span elements whose id attributes are "email", "phone", "zip", and "dob". Use the jQuery text() method of the span elements to do this.
6. In the handler for the click event of the Back button, add code that uses the history object to go back to the previous page.
7. Run the application, enter valid data, and click Save. After you review the data that's displayed on the profile.html page, press F12 to open the developer tools and display the Application panel to view the data in session storage for this application.
8. Click on the Back button, make a change, and click Save. Then, display the Application panel of the developer tools again to see how the data in session storage has changed.

Extra 18-1 Use data from the JSON Placeholder API

In this exercise, use data from the JSON Placeholder API to create an application that displays a list of to-do items for a specified user. After selecting a user and clicking "View List", the application should look like this:



ToDo Item	Completed
explicabo consectetur debitis voluptates quas quae culpa rerum non	true
maiores accusantium architecto necessitatibus reiciendis ea aut	true
eum non recusandae cupiditate animi	false
ut eum exercitationem sint	false
beatae qui ullam incidunt voluptatem non nisi aliquam	false
molestiae suscipit ratione nihil odio libero impedit vero totam	true
eum itaque quod reprehenderit et facilis dolor autem ut	true
esse quas et quo quasi exercitationem	false
animi voluptas quod perferendis est	false

1. Open the application in this folder:

   `exercises_extra\ch18\todo_list\`

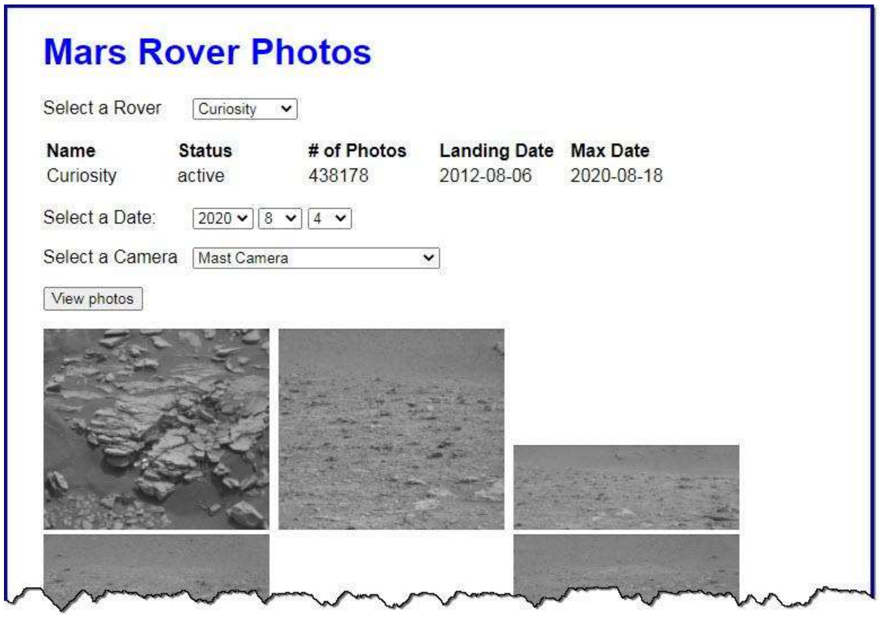
2. In the todo_list.js file, there's a const named domain that contains the URL for the JSON Placeholder API.
3. Add an asynchronous function named displayUsers() that makes an asynchronous request from the API for all ten users.
4. Display the users in the select element whose id is "users". Make sure the option elements have a value attribute that contains the id value for each user.
5. Add an asynchronous ready event handler. It should call the displayUsers() function and add an asynchronous event handler for the click event of the button.
6. The click event handler should get the user id for the selected user from the select element and use it to make an asynchronous request from the API for all the to-do items associated with that user. To do that, you can use add a querystring to the API URL, like this:

   `https://jsonplaceholder.typicode.com/todos/?userId=<id goes here>`

7. Display the to-do items in the div element whose id is "list". Format the items as an HTML table.

Extra 18-2 Use data from NASA's Mars Rover Photos API

In this exercise, you'll modify an application that allows you to view photos from any of the rovers sent to Mars by NASA. After selecting a rover, a date, and camera and clicking the "View" button, the application should look like this:


Mars Rover Photos

Select a Rover: Curiosity

Name	Status	# of Photos	Landing Date	Max Date
Curiosity	active	438178	2012-08-06	2020-08-18

Select a Date: 2020 8 4

Select a Camera: Mast Camera

1. Open the application in this folder:

   `exercises_extra\ch18\mars_rover\`

2. Open the rover.js file. Note that it already contains all of the JavaScript for working with the user interface, except the asynchronous getJson() function.
3. Add the function named getJson() that makes an asynchronous request for the specified URL and returns an object that's created from the JSON that's returned by the request.
4. Run the application to make sure it works correctly. It should.
5. Add a console.log() statement to the getJson() function that prints the request URL to the console.
6. Run the application. When you select a rover, note that the URL that's passed to the getJson() function is:

   `https://api.nasa.gov/mars-photos/api/v1/rovers?api_key=DEMO_KEY&page=1`

7. When you click the View button, note that the end of the URL that's passed to the getJson() function uses a format like this:

   `.../rovers/Spirit/photos/?api_key=DEMO_KEY&page=1&earth_date=2010-3-21`

8. To see all the options available for the Mars Rover Photos API, visit https://api.nasa.gov/, scroll down, and expand the Mars Rover Photos item.