

Assignment #2

JavaScript

CS193C Summer 2015, Young

In this assignment we get a chance to work with JavaScript. This assignment is due Thursday, July 16th at 3:15pm.

For this assignment, you are not allowed to use WYSIWYG editors (e.g., no Dreamweaver) and should only use a text editor to compose your assignment. In addition, your files should work on both the latest version of Mozilla Firefox and Google Chrome.

Car Options

In this part of the assignment you will create a webpage which allows a user to configure a car to see how much it costs. Here is a screenshot showing my version of the assignment:


The screenshot shows a web browser window titled "Configure Your GT Super Sportscar - Windows Internet Explorer". The address bar shows a local file path. The page content is as follows:

Configure Your GT Super Sportscar

CONFIGURATION	
<input checked="" type="radio"/> GT Manual	\$17,790.00
<input type="radio"/> GT Automatic	\$18,590.00
<input type="radio"/> GT-S Manual	\$22,455.00
<input type="radio"/> GT-S Sportshift	\$23,155.00

CHOOSE A COLOR

☒ Red
☐ Blue
☐ Silver
☐ White
☐ Black



FACTORY OPTIONS	
<input type="radio"/> Option Combo #1	\$1235.00
• Power Windows, Doors, Cruise Control	
<input type="radio"/> Option Combo #2	\$3354.00
• Rear Spoiler and Fog Lamps	
• Keyless Entry	
• Power Tilt & Slide Moonroof	
• Power Windows, Doors, Cruise Control	
<input checked="" type="radio"/> No Combo	\$0

DEALER OPTIONS	
<input type="checkbox"/> Upgraded Stereo System	\$550.00
<input type="checkbox"/> VIP Security System	\$399.00
<input type="checkbox"/> Auto-Dimming Mirror	\$295.00

TOTAL PRICE	
<input type="text"/>	<input type="button" value="Calculate Total"/>

As you can see, the car comes in four different configurations:

- GT Manual \$17,790.00
- GT Automatic \$18,590.00
- GT-S Manual \$22,455.00
- GT-S Sportshift \$23,155.00

The user chooses a basic configuration and then chooses a factory options package and adds on any dealer options desired. Here are the factory options:

- Option Combo #1 \$1235.00
- Power Windows, Doors, Cruise Control
- Option Combo #2 \$3354.00
- Rear Spoiler and Fog Lamps
- Keyless Entry
- Power Tilt & Slide Moonroof
- Power Windows, Doors, Cruise Control
- No Combo \$0

and here are the dealer options:

- CD Autochanger \$550.00
- VIP Security System \$399.00
- Auto-Dimming Mirror \$295.00

When the user clicks on the “Calculate Total” button, you should determine the price of the car, given their configuration and option choices.

In addition, as you can see on the right of the webpage I’ve given the user an option to choose a color for their car. When they change the color in the list, the picture of the car should automatically change. I’ve included five car photos for you to use with this assignment.

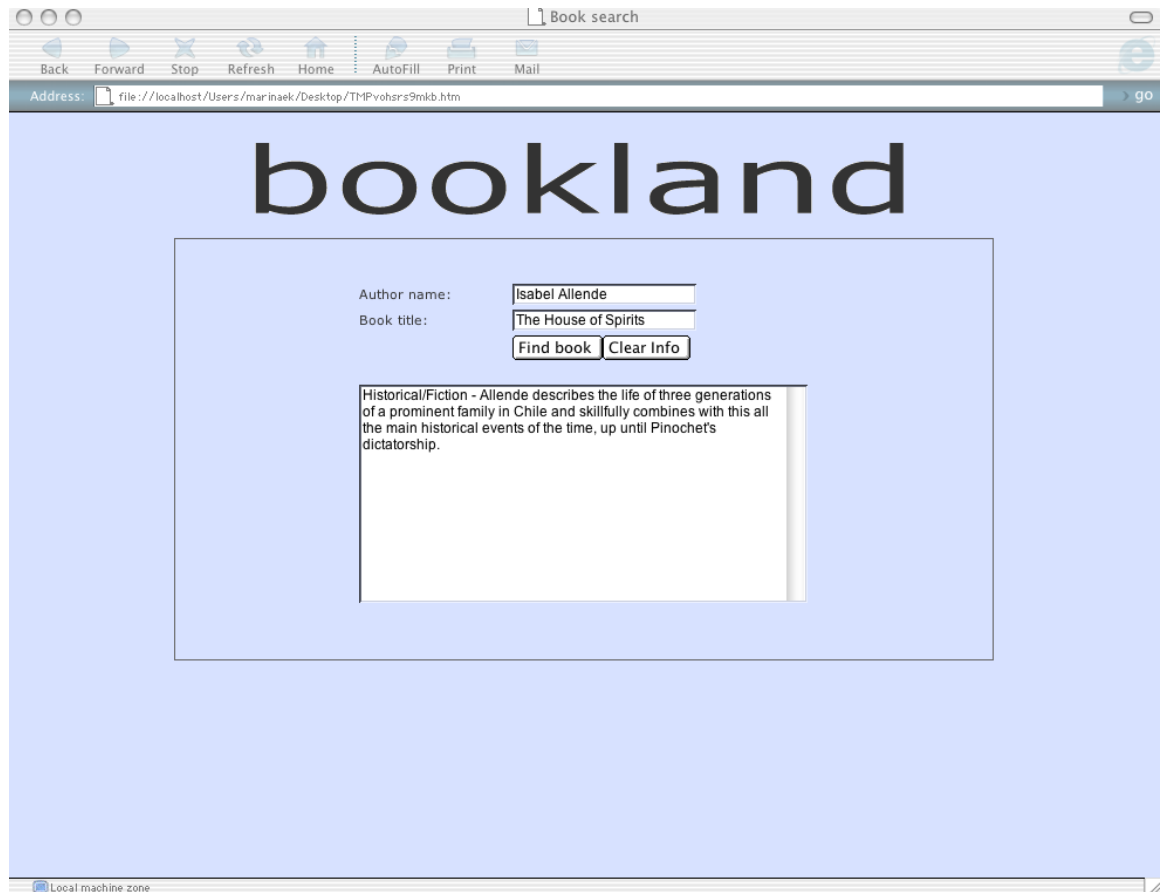
I’ve gussied up the webpage using stylesheets. Make sure you get the layout right—heading across the top of the webpage and then two columns. Additional styling isn’t strictly necessary, but is a nice opportunity for CSS practice. In addition, you may notice that I’ve converted the total cost back to a price (\$20,375.00 instead of simply 20375). You may want to consider doing this using JavaScript strings.

Bookland

For this part of the assignment, you will create a webpage that allows the user to search for a book with a specific title and/or by a specific author. Then, you will display information about the book on the web page. We are providing book data that you can use in a file called books.txt. For each book, we provide the author’s name, the book’s title, and a short description of the book.

The way this should work is as follows: You should store the book data we provide in arrays. You can use multiple synchronized arrays or one array of objects. The user of your webpage will type in a name and/or a title. You will search for this specific book (or books), using the given name and/or title, in your array(s). Once you find a match, you should display its information on the page. You may need to display information for several books if there are several matches. Please see the *Things to note* paragraph later on to find out exactly what you need to display.

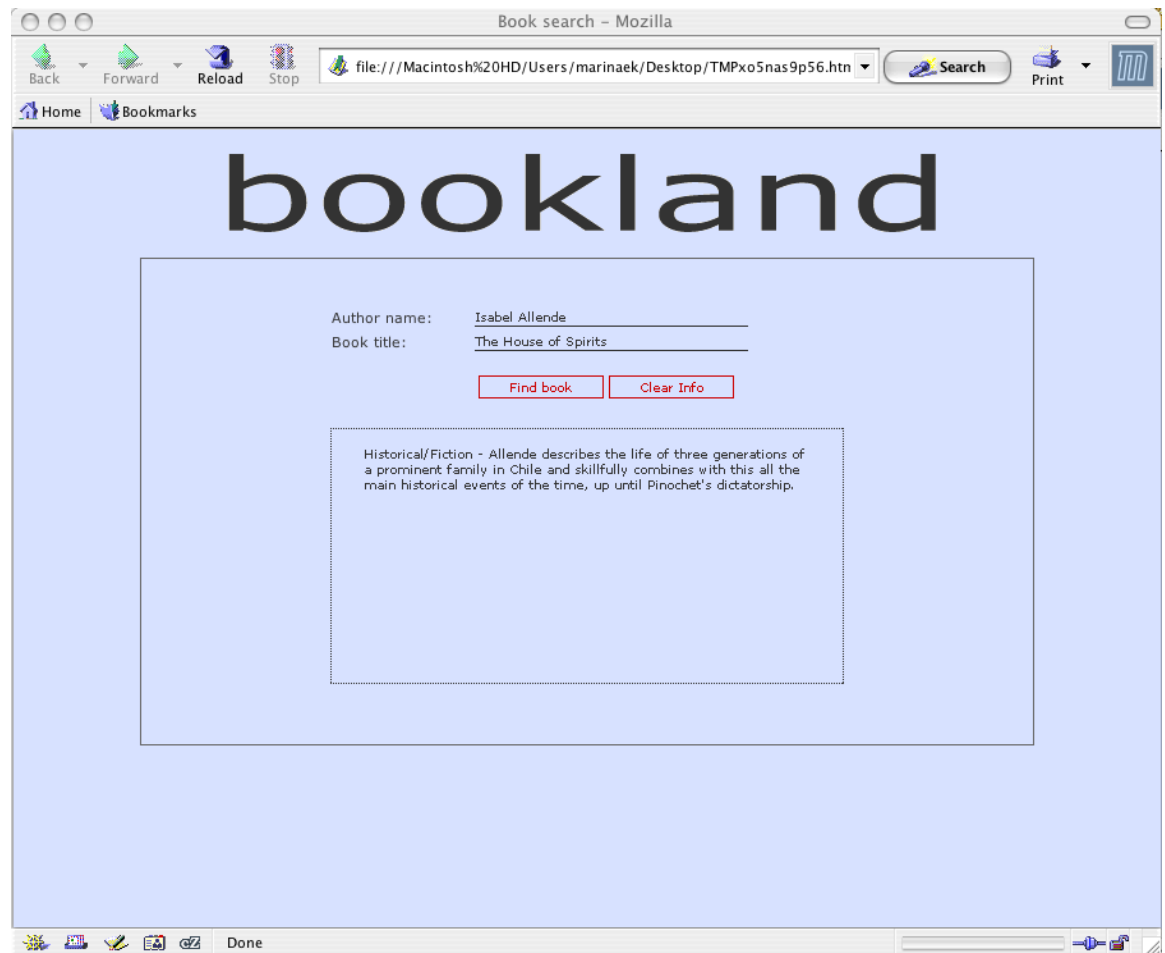
Since we are only displaying text on the page, we can just use a textarea in which to display the information. Here’s what the page might look like:



In this example, the user typed in “Isabel Allende” as the author’s name and “The House of Spirits” as the book title. When the user clicked “Find book”, the short description for this specific book was displayed in the textarea.

Things to note: You do not have to worry about case sensitivity. If you want to make the search case insensitive, go for it but it is not required. If a book is not found, your page should do something reasonable, i.e. either display “Book not found” in the textarea or display nothing. If the user enters both a name and a title, you should display the description of the first match, i.e. the first book found that satisfies the given criteria. If the user only enters an author name, you can either display the title and description of the first match, i.e. the first book that you found by that author, or you can display the titles and descriptions of all books by that author. Similarly, if the user only enters a title, you can display the author and description of the first match, i.e. the first book that you found with that title, or you can display the authors and descriptions of all books with that title. All information must be displayed in the textarea. Assume the user has to enter full names to get a match – so Isabel will not get a match, but Isabel Allende will.

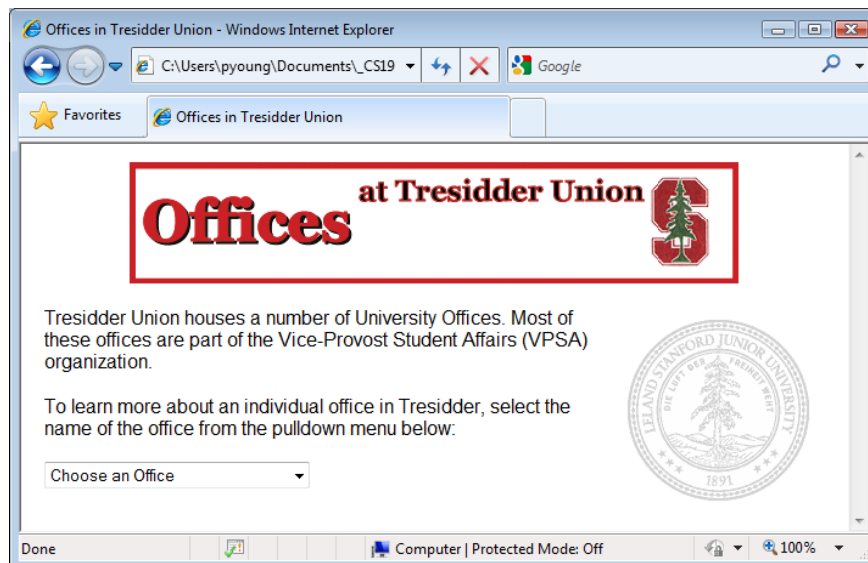
Once you get the array search working, try to see how you can improve the page using style sheets. Several professional web pages actually use form elements to display text information on pages, as we are doing in this problem. However, they also use CSS to hide the form elements and to make their pages look better. We’d like you to experiment with different CSS to see how you can change the form elements. Here’s our version of the web page with CSS:



We have changed the appearance of the text fields, buttons and textarea using several different style attributes. Your page does not have to look like ours. Feel free to change things any way you want to, as long as you experiment with CSS and form elements.

Offices at Tresidder Union

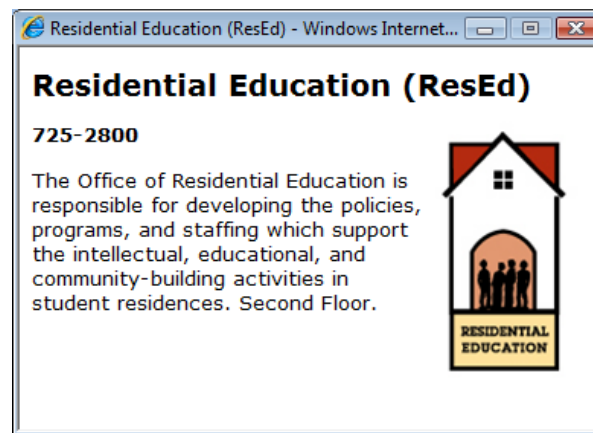
In this problem we experiment with dynamically creating a new webpages using the Document object's `writeln` method. To start out with create the following webpage:



Your webpage should have the following characteristics:

- A banner centered horizontally at the top of the webpage. Use the `offices-gnr.gif` file provided.
- Center the webpage and set the webpage contents to a width of 640 pixels with the remaining width of the window evenly split between the left and right margins. You can do this by setting `margin-left` and `margin-right` to the special value `auto` instead of providing an actual measurement for them.
- Place the Stanford seal floated to the right side of the window.
- Create a pulldown menu listing the following items: “Choose an Office”, “Residential Education (ResEd)”, “Student Services Center”, “Office of Student Affairs (OSA)”, and “Vice-Provost of Student Affairs”.

Now, write JavaScript code such that when any of the offices are selected, a popup window is created displaying information about that office. Here is an example of the popup window displayed when the “Residential Education” menu item is selected:



As you can see, each popup window should display the name of the office, the phone number, a brief description, and a logo. Use the ResEd logo for Residential Education use the Stanford Seal for the remaining offices. Unformatted text descriptions for each are included with this assignment’s downloads. I’ve also included a large version of the ResEd logo and a small-sized version of the Stanford Seal in the downloads.

If a previous office’s popup window is still up when the user selects a new item make sure you reuse the window (instead of generating an additional popup window). Also make sure your popup window ends up on top of the main window (use the popup window’s `focus` method to bring it to the forefront).

The webpage should be dynamically generated (i.e., don’t create four different HTML files). Instead store the information about each office in an array in JavaScript and then creating the contents of each popup window dynamically using the `document.writeln` function. Remember, the `window.open` function returns the `Window` object associated with the popup. From the popup’s `Window` object you can access the popup window’s document object. Use the document’s `writeln` method to generate the contents. Creating the popup locally will reduce the load on your webserver and may increase the user’s response time.

Credits

The Car photos (and the car prices) are from a Toyota Celica and are taken from www.toyota.com. I've eliminated car options and car colors to keep the assignment simple. The Stanford Seal used for the Offices webpage was taken from the Stanford Administration homepage. In some cases, the images were modified by cropping or resizing.

Text for Tresidder was again borrowed from the official "Tresidder Memorial Union" pamphlet. Additional text was taken from the VPSA organizational chart found at adminguide.stanford.edu. The ResEd description was taken from resed.stanford.edu.

Book problem created by Marina Kassianidou, former CS193C TA. The short description of each book in the books.txt file is taken from the books' back cover.