Project 2
Storyteller: Functions, Strings, and Characters

In this project, you will see that the advantages of programming with functions are many. Functions help us write larger programs without getting ourselves lost in the complexity. They also help us write code that's easier for us (as humans) to read and understand. These benefits are immediate, but functions also provide future benefit by making programs easier to fix, change, and extend. In fact, only the smallest computer programs are written without using this technique. Meanwhile, you will also gain experience with some new JavaScript features, including working with strings and characters.

Vocabulary

All key vocabulary used in this project are listed below, with closely related words listed together:

> function definition vs. function call
> parameter

Post-Project Questions

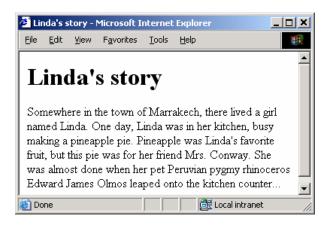Write your answers after completing the project, but read them carefully now and keep them in mind during the project.

1. In an error-free JavaScript program, can you have a function definition without a call to it?

2. Syntactically speaking, i.e., looking at JavaScript code, how can you distinguish a function definition from a function call?

3. In this lab, you produce a storyteller page that allows the user to "fill in the blanks" in a short story structure that you provide. Describe how "filling in the blanks" is one way to think of what happens when a function with parameters is called. In the context of calling functions, what are the blanks, and what is used to fill them?

4. What was the most difficult part of the project? What was the easiest?

5. What changes would you make to the project in the future??

Discussion and Procedure

By the end of this project, you will write a fun web page that lets the user help create a short story. You, as the programmer, will decide how most of the story goes, leaving some important blanks for the user to fill in using a web form. For instance, the web form might look like this…

…and when the user clicks the "Tell a story button", a new window like this might pop up:

Notice that the information the user filled in on the web form has been incorporated into the story (and the title), filling in the important details (characters, events, etc.). In most cases, this is as simple as taking the user input and inserting it directly into the story. Examples of this above include the inclusion of the name Linda and the first occurrence of "pineapple." There are, however, also some places where the user input has been altered slightly. The third sentence begins with "Pineapple," where the first letter of the input has been capitalized, because it appears at the beginning of a sentence in the story.

Notice that the story includes words that vary with gender, such as "girl," "her," and "she." The use of the masculine, feminine, or other form of these words depends on user's choice on the web form. You will be writing the JavaScript code to incorporate or otherwise take into account the input the user provides on the web form and produce the story. As we build this web page using HTML and JavaScript, we'll see how functions can come in handy in the various ways discussed in the lab introduction.

Throughout this project, we will provide a few pieces of code so you can focus on practicing the skills that are the focus of the project. For instance, we'll give you the code to pop up a new window for showing the short story.

Part 1. Starting the form

As usual, let's start by getting something very simple working properly.

1. *Create a new web page with a form that has a text entry and a button as shown below.* See examples from Chapters 17 and 18, as well as the previous lab, if you need a review of HTML tags for web form elements.

In the labs, when we wanted a button to cause code to run, we put all of that code in the button's **input** tag, using the **onchange** attribute. In the next steps, we'll see how we can organize our HTML and JavaScript more neatly, putting almost all of the code in a single **script** section at the top of the HTML file but still being able to associate code with the button.

2. *Add a script section to your HTML file before the body section.* Just leave it empty for now.

Let's consider what we want to happen when the Tell a story button is clicked. First, the value in the first name text box needs to be retrieved so we can use it in the story. Then, a new browser window containing the story needs to be created. You know how to do the first part, and we'll provide code to do the second part. More specifically, we'll provide a function that does that for you. Before we move onto that, let's write our first function.

One way to think of a function is as a section of code that's packaged up and named to make it easy to run multiple times. (In this sense, it's a lot like a small "program in a

program.") In this case, you're going to start writing a function that should be run whenever the Tell a story button is clicked.

3. *Type or copy-and-paste the code shown below into your* **script** *section.* We're going to put an **alert** in the function definition for now, so you can test to make sure it works.

```
function tellStory() {
   alert("function tellStory() was called");
}
```

4. *Load (or reload) your web page in a browser.* Does the **alert** dialog box appear?

How about when you click the Tell a story button?

At this point, the **tellStory** function has been defined, but it isn't called from anywhere, so the code in the function definition (the **alert** line) doesn't ever run. (If you don't remember the difference between defining and calling a function, reviewing Chapter 19 will be essential to getting through the rest of this lab.) In the next step, you'll set up the Tell a story button to call the function.

5. *Add the attribute shown below to the Tell a story button's input tag.*

```
onclick='tellStory();'
```

Do you expect the dialog box to appear when you reload the page now? How about when you click the button?

The nice thing about setting up your HTML and JavaScript this way is that even if the **tellStory** function definition gets long, that code will be in the **script** section, clearly separated from the HTML, and the button tag doesn't have to change. Before going on, reload the page and confirm that clicking the button brings up the **alert** dialog box.

Next, we'll provide you with a function for popping up a new browser window for your story.

6. *Type or copy-and-paste the code shown below into your* **script** *section.* Although we've provided comments to explain roughly what this code does, don't

expect to understand every part of this code at this point.

```
// opens new window for story contents, using given
// name to personalize title, "name's story"
function setupStoryWindow(name, contentString) {
  // some HTML to put around the story itself,
  // including a page title incorporating user's name
  var header = "<head><title>" + name
    + "'s story</title></head>"
    + "<h1>" + name + "'s story</h1>";

  // open new window and "fill" it w/ HTML, including
  // the story
  var storyWindow = window.open('', 'storyWindow');
  storyWindow.document.write(header);
  storyWindow.document.write(contentString);
  storyWindow.document.close();

  // raise this window, in case it's not visible
  storyWindow.focus();
}
```

This function is the one you should call from your **tellStory** function. Notice that this function has parameters: **name** and **contentString**. In the next few steps, we'll experiment with calling this function.

7. *In the* **tellStory** *function definition, replace the* **alert** *line with a call to the* **setupStoryWindow** *function.* For now, just pick two strings to pass to **setupStoryWindow**. You can use your name and **"Once upon a time..."**, for instance.

   Reload the page and click the Tell a story button. Where do the strings you pass in for **name** and **contentString** appear in the pop up window?

8. *Modify* **tellStory** *to pass the name the user enters in the web form to the* **setupStoryWindow** *function.* We recommend you store this value in a variable, then pass it to the function.

Part 2.   The story structure

All that remains now is to extend the web form and, simultaneously, begin crafting your story structure. You may start with the example story structure shown at the beginning of this project or begin creating one of your own. If you use the example story structure, change it in some way to make it your own.

9. *Add a gender identity drop-down list to the web form.* In the **tellStory** function, add code to retrieve the value chosen in the list and store it in an appropriately named variable.

To get you started and for the purposes of working through an example of using the gender choice, let's start your story with this sentence, where curly braces mark off parts that depend on user input in some way:

Once upon a time, there lived a **{girl or boy or person}** named **{name}**.

In the **tellStory** function, the idea is to build up the story, a few words at a time, and store them into a string variable, as shown below. When we're done storing the story in a long string, we pass that on to the **setupStoryWindow** function as the second argument.

```
// retrieve values from web form into variables above

var story = "";
story = story + "Once upon a time, there lived a ";

// extend story further, adding strings to var story
```

We've added the beginning of the first sentence to the variable. The next word depends on the user's input for gender. You already know how to write conditionals, so you could just put an **if** statement here to take care of this, but this would be a good time to consider whether calling a small function wouldn't be a more appropriate way of doing so, as shown below:

```
story = story + identity(gender)
   + " named " + name + ".  ";
```

The above code assumes that you saved the user input for gender and name in the variables **gender** and **name**, respectively. If you wrote a little function **identity** that takes one parameter and returns one of these string **"boy", "girl", "other"**, depending on the value passed in, then the above code would work as intended.

Why bother doing it this way? If you needed to use the word "boy", "girl", or "person" elsewhere in the story, you could just use the call **identity(gender)** again, instead of copying the lines of code that your conditional would require. This has a number of benefits. First, by separating the conditional code into a function definition, it keeps the code in **tellStory** simpler and easier to read. Second, it makes your code easier to update. Suppose you end up using "boy" or "girl" in several places in your story, and instead of using the **identity** function, you copied and pasted the lines of code for the conditional in multiple places in **tellStory**. If you decided you wanted to change the words "girl" to "gal" or "dude"instead, you would have to comb through your **tellStory** function and change those strings in every place you copied the conditional code. If you just have multiple calls to the **identity** function instead,

you would only have to change the function definition—one change in one place, and you'd be done.

10. *Write a definition for the* **identity** *function in your* **script** *section, and add a call to the function to the* **tellStory** *function as shown above.* You might want to look back at your web form's gender drop-down list to remind yourself of what values each of your options corresponds to. You can also refer to Chapter 19 for how to write functions that take parameters and return values.

Remember to reload and test the page. At this point, you should have three function definitions in your script section: **tellStory**, **setupStoryWindow**, and **identity**, not necessarily in that order. (Remember, the order of function definitions doesn't matter.)

As you extend your story, you might find that you'll need a few other similar functions for gender-related words like he/she and his/her. We'll leave writing those to you, since their definitions will look just like **identity**'s. In the next part of the project, you'll write some other useful functions for incorporating user input into your story.

Part 3.   Helper functions for grammar

If you look back on the example story, you'll notice a few other places where incorporating the user's input requires adjusting words and capitalization. We've underlined a couple examples here:

Linda was in her kitchen, busy making a pineapple pie. Pineapple was Linda's favorite fruit, but this pie was for her friend Ralph. She was almost done when her pet Peruvian pygmy rhinoceros Edward James Olmos leaped onto the kitchen counter...

The first case is where "a" or "an" has to be chosen, depending on whether the word the user provided begins with a vowel. The second case is where a word the user provided begins a sentence, so its first letter must be changed to upper case. Both of these suggest short functions, which we'll guide you through implementing and calling in this final part of the project.

We'll start with the task of capitalizing a word. The goal here is to implement a function that takes a word (a string) as a parameter and returns the word with the first letter capitalized. You'll find the following hints helpful:

The suggested algorithm is to split the word into two strings: one with just the first letter and the other with the rest of it. Then, you change the first letter to upper case and concatenate it with the rest of the word to get the desired result.

If you have a word stored in a variable, you can get its first letter like this:

7

```
var word = "pineapple";
var firstLetter = word.charAt(0);
```

That's a zero being passed to **charAt**. The individual characters of a string are numbered starting at zero, so the above code gets the zero-th character.

You can also get everything except the first letter like this:

```
var restOfWord = word.substr(1, word.length – 1);
```

The right-hand side is a little complicated, so don't worry if you find it a little puzzling. Basically, it asks for all of the characters in the string, starting from the one at 1 (which is actually the second character, because we number them starting at 0 instead of 1).

Now, the only other trick you need to know is how to capitalize a string. If you have the string stored in some variable **someString**, you can do this:

```
someString = someString.toUpperCase();
```

11. *Add a function for capitalizing a word and test it by calling the function from* **tellStory** *to generate your story.* In the meanwhile, you might need to add some more text boxes and/or drop-down lists to your web form to have more input from the user to work with in your story.

The other function suggested by the example story excerpt above is one that appropriately chooses to put "a" or "an" before a word, depending on whether the word begins with a vowel. We'll provide you with a helper function that is a little complicated to implement:

```
// takes a character c and returns true if it is a
// vowel and false if not; case-insensitive
function isVowel(c) {
  // basic strategy:  make character lower case,
  // then see if it's in a string that contains all
  // of the vowels
  var vowels = "aeiou";
  if ( vowels.indexOf(c.toLowerCase()) == -1 ) {
    // indexOf returns -1 if char. not found in string
    return false;
  } else {
    return true;
  }
}
```

There are at least two ways you could implement the "a"/"an" function. In either case, you need to pass the word to the function. On the one hand, you could implement a function that just returns **"a"** or **"an"**. Alternatively, your function could return a longer string: **"a "** or **"an "** followed by the word that was passed in.

12. *Add a function choosing "a" or "an" and test it by calling the function from* `tellStory` *to generate your story.*

Another benefit of writing functions for these tasks is that it makes it easier to combine them. For instance, suppose you needed to start a sentence with "a"/"an".

How you would use both the capitalizing and "a"/"an" functions together to do this? You can explain by giving a code example or using words.

You now have the tools necessary to easily (and grammatically correctly) incorporate user input you have gathered from your web form into a short story. As you complete your story, you'll probably find that you'll need to call your helper functions again, but for practice's sake, make sure you have at least two calls for each of the functions you write.

Deliverables

For Project 2A, create an HTML page that includes:

1. The text of the story you will use. If you are using the example story, you must change it in some way. The story must be at least two paragraphs long. You may also use some story or document that is in the public domain, such as the Declaration of Independence or a fairy tale.

2. Mark the words in your story that will be filled in by the user. A nice way to do that is to use span tags and a style at the top of your document. For example, this **** word**** will be replaced. In the header section of your HTML page, a style section would define the class "replace", which always has a dot in front of it in a styles listing:

   ```
   <style type="text/css">
           .replace {color: #ccffff;}
   </style>
   ```

3. Create your user input form as described in the project description above.

4. Set a background color for the page.

5. Add an image that enhances your story.

6. Provide copyright information for the story and image you use. If you use the example story, state "By permission of instructor." Otherwise, cite the source and link to it on your html page. If you wrote the story yourself, state "© 2008 Your Name." If you don't include the source of the story, that is considered plagiarism, using someone else's work for your benefit, and it is also considered cheating. This course has a zero-tolerance policy on cheating. (See Academic Conduct section of course Web site for more information.)

7. Validate the HTML page.

8. For extra credit, embellish the page to make it attractive and enhance your story. Use CSS styles in the header section. Use CSS styles for html elements such as <p> and for classes you define, in addition to .replace. See the W3 Schools tutorial on CSS for more information.

9. Create a **project2A** folder in your **fit100** directory. Save the page as **fit100/project2A/project2A.html**

Rubric

Project 2A: 50 points plus 30 points extra credit

Item	Points
2-paragraph story	10
Mark names in story to change with span tags	10
Image	5
Set background color	2
Input form for user to fill out including input fields and submit button	13
Validates according to XHTML 1.0 transitional	10
Extra Credit: Validates according to XHTML 1.0 strict	10
Extra Credit: More embellishments to make an attractive page using CSS	20

Do not continue working on any project part after the deadline or it will be considered late!

If you want to make further changes to the project after the deadline, copy the project folder and save it as a different name before uploading it.

Project 2B: 100 points

In this part of the project, you will implement the coding to make the story happen, according to the steps in the project description.

1. Create a new folder in your `fit100` directory, called `project2A`. Copy everything in your `fit100/project2A/` directory to your new `project2B` directory. Rename your `project2A.html` to `project2B.html` before you begin work on Project 2.

2. Add the tellStory function as described in the project description.

3. Add the Capitalization function as described in the project description.

4. Add the a/an function as described in the project description.

5. Add the identity function as described in the project description.

6. When the Submit button is clicked, the story appears in another window, as described in the project description.

7. When the Submit button is clicked, some word in the <h1> heading should change, for example, placing the user's name in "Linda's story" in the project description. If you chose a different story, use a different title and have some word change in the <h1>.

8. When that is working, add a function to change the background color of the page when the Submit button is clicked.

9. For extra credit, add a question that uses radio buttons to give the user a choice of answers. Incorporate the answer into your story.

10. For extra credit, add a question that can have multiple answers by using checkboxes. Incorporate the answers into your story.

Item	Points
tellStory function	15
capitalization function	15
a/an function	15
identity function	15
Change word in <h1> based on user input from form	10
Change background color of page when Submit button is clicked	10
Submit button calls tellStory and creates story in other window	15
Answers to post-project questions	5
Extra Credit: Add a question that uses radio buttons	15 points
Extra Credit: Add a question that uses checkboxes	15 points

Do not continue working on this project part after the deadline or it will be considered late!

If you want to make further changes to the project after the deadline, copy the project folder and save it as a different name before uploading it.

Turn-in

Submit your Projects 2A and 2B through Catalyst Collect-It. For Project 2A and 2B, please turn in only a Word document or .txt file with the URL for your html file, your name, UW NetID, Student ID No., and Lab section, and the answers to the Post-Project Questions listed at the beginning of this lab.

- **Project 2A Due Date: Friday, February 22, before 11:00 pm.**

- **Project 2B Due Date: Wednesday, February 27, before 11:00 pm.**

Upload only the Word or .txt file to the Catalyst Collect-It Turn-in Area on the Course Calendar for this project part (2A or 2B).

Do not continue working on the project after the deadline or it will be considered late!

If you want to make further changes to the project after the deadline, copy the project folder and save it as a different name before uploading it.