Guidelines

To answer the challenges below, you will need to create

1. An HTML5 document
2. A CSS file
3. A jquery plugin called jquery.challenges
   a. NOTE: if you are not comfortable creating a jquery plugin, just write your javascript code as you normally would without wrapping it as a plugin
4. A PHP REST API

NOTE: The Disney mail server will remove any javascript files found inside unencrypted zip files, so you have the following options for delivering your solutions:

1. upload your finished solution to your own webserver and email a link
2. share your web files via Dropbox, SugarSync, etc
3. encrypt a zip file with your code in it

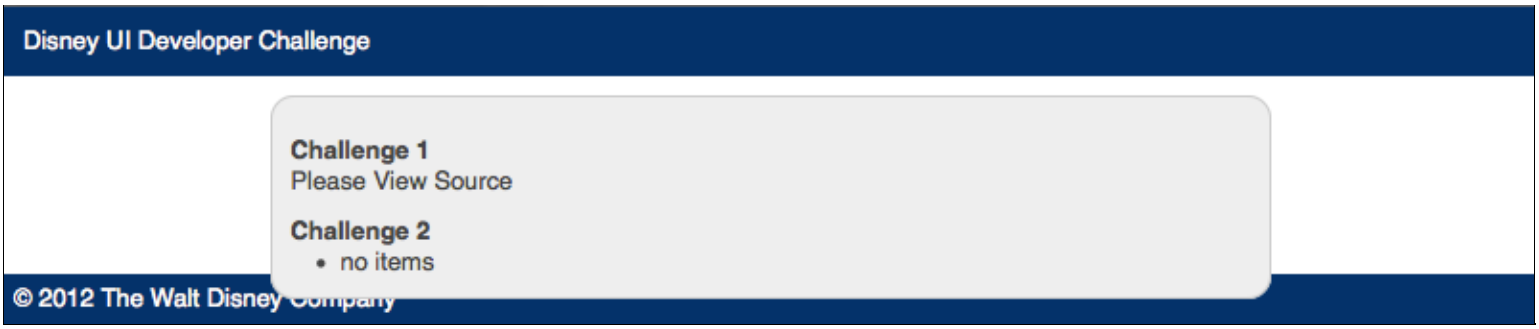
Challenge 1: Basic Framework

Create the HTML5 file, which will load jQuery and your jQuery plugin. You may use any industry standard templates, boilerplates, or libraries to help you make your document HTML5 compatible across browsers. Make sure you include your CSS file and any other JavaScript files you will be using.

The HTML for your page should contain ONLY enough to render the following:

The specific colors are not important, feel free to add whatever visual styles you like.

Use CSS3 to round the corners of the inner box and center it on the page. Also write the css needed to make it so that the gray content box overlaps the footer if the window is shrunk down so that they collide:



Challenge 2: Generate Sorted HTML

Populate the UL with a list of names using the following array in JavaScript, sorting them by last name, first name:

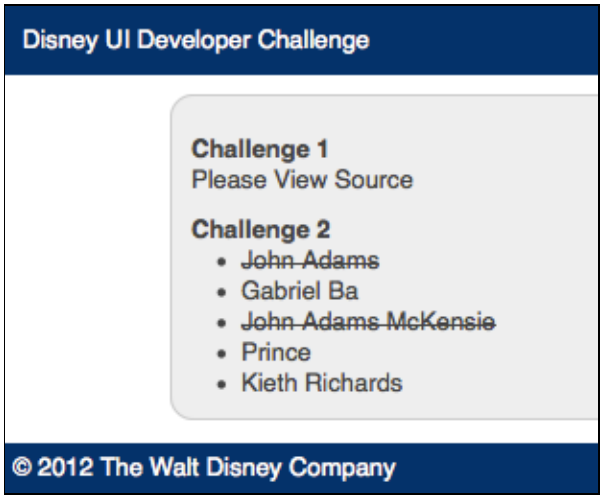
```
var aNames = ['Gabriel Ba','John Adams','Kieth Richards', 'Prince', 'John Adams McKensie'];
```

such that names becomes ordered like so:

```
['John Adams', 'Gabriel Ba', 'John Adams McKensie', 'Prince', 'Kieth Richards']
```

Challenge 3: Item Actions

- When a user clicks on a name in the list, the name should strike out
- When a user clicks on a name that is already stricken, it should un-strike out
- When the mouse hovers over an actionable item, the mouse cursor should have the appropriate style



Challenge 4: Values + Timing

Add the following function to your page and fix it.

```
/**
 * countdown logs a countdown from num to 0,
 * showing the step number before the value
 * with a 1 second delay between each log
 * ... or does it? Why?
 * TODO: fix this
 * @param num The starting number
 */
function countdown (num) {
    for (var i = 0; i <= num; i+1) {
        setTimeout(function () {
            console.log(i +': '+num - i);
        }, i * 1000);
    }
}

countdown(5);

// this should return:
// 1: 5
// 2: 4
// 3: 3
// 4: 2
// 5: 1
// 6: 0
```

Challenge 5: Array Max

Write a function that accepts an array and returns the maximum value contained in the array or in any array nested within.

Example Input => output

```
[[141,151,161], 2, 3, [101, 202, [303,404]]] => 404
```

Challenge 6: Strings

Write a function whose arguments are an arbitrary number of character strings. It should return an integer representing the length of the longest string passed as an argument

Example

```
"a", "aaa", "aa" => 3
"a", "bcd", "efgh", "ij", "" => 4
```

Challenge 7: Validation

Write a function whose argument will contain string data representing some phone number (entered by the user). A valid phone number may consist of between 7 and 12 digits (0..9). We are not concerned with validating the number itself is valid for any particular country (so don't worry about it beginning with a 0, for example). Assume that in between some adjacent digits there may optionally appear either a single space, or a single hyphen, but the first and the last character of the string should be a number. Any other phone number is invalid.

If the string is valid, the return value of your function should contain a string containing the 7-12 valid digits, representing the same phone number after removing all hyphens and spaces.

If the phone number is invalid, return false and throw an exception with the text "Invalid phone number".

Example:

```
'012-345 69' => '01234569'
any of the following throws an error:
'012345', '-012345 678', '01203- 34566', '123456678875432', '1234x567'
```

Challenge 8: PHP REST API

Create a PHP file (or set of files) that constitutes a REST API for your application. From the client, call the API to fetch a shuffled deck of cards.

In PHP, you will need to create a class (or set of classes) to represent a deck of cards and the actions you might take on the deck.

This class must have the following methods: shuffle(), draw(), repack()–and the PHP API should have interfaces that the UI can call.

Provide buttons in the UI to do the appropriate actions on the API and fetch an appropriate response:

1. A button action for "shuffle" will shuffle the deck on the server and return the number of cards in the deck.
2. A button for "draw" should relate to a numeric input field that allows the user to specify how many cards they want to draw. The API will then pull that many cards off the deck on the server and return the cards to the user. You can simply log the card value(s) to the console or put it on the page somewhere.
3. A button for "repack" will put all the cards back from the user's hand into the deck and pack it in factory order (A-K, grouped by suit)

You can use whatever methods you want on the server side to track the state of the deck of cards, which cards are drawn and in the player's hand–including a mysql db or redis store or some other form of persistence.

For extra credit/fun, you can add a 'discard' method and action for the user.

If there is any ambiguity or if you find that there are missing requirements, feel free to improvise. Be prepared to explain.