Frontend development course

Review of tasks

Author: Daniel Chabr

Task 1

Define a function sum() and a function multiply() that sums and multiplies (respectively) all the numbers in an array of numbers.

For example, sum([1,2,3,4]) should return 10, and multiply([1,2,3,4]) should return 24.

Task 1 - solution

```
function sum(numbers) {
    var result = 0;
    for (var i = 0; i < numbers.length; i++) {
        result += numbers[i];
    }
    return result;
}

function multiply(numbers) {
    var result = 1;
    for (var i = 0; i < numbers.length; i++) {
        result *= numbers[i];
    }
    return result;
}
```

Task 2

Represent a small bilingual lexicon as a Javascript object in the following fashion {"merry”:”veselé", "christmas”:"Vánoce", "and”:"a"} and use it to translate the text “Merry Christmas and Happy New Year".

You can access object attributes by variable with this notation: myObject[word]. The variable word here for example contains string “christmas" and so it would return string "Vánoce" as a result. More info at https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Operators/Property_accessors.

You might also find the following methods useful:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/split

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/join

Task 2 - solution

```
var text = "Merry Christmas and Happy New Year";
var engToCze = {
    "Merry"     : "Veselé",
    "Christmas" : "Vánoce",
    "and"       : "a",
    "Happy"     : "šťastný",
    "New"       : "Nový",
    "Year"      : "rok"
};

function translate (text) {
    var words = text.split(' ');
    for (var i = 0; i < words.length; i++) {
        //words[i] = engToCze[words[i]];
        for (var j = 0; j < engToCze.
    }
    return words.join(' ');
}
```

Frontend development course

jQuery

Author: Daniel Chabr

What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers

How to use jQuery

jQuery is a library, thus it needs to be loaded before we start using it. The library can be downloaded at http://jquery.com/ and then imported into website as following:

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Demo</title>
</head>
<body>
    <a href="http://jquery.com/">jQuery</a>
    <script src="jquery.js"></script>
    <script>

    // Your code goes here.

    </script>
</body>
</html>
```

jQuery vs $

The jQuery library provides the jQuery function, which lets you select elements using CSS selectors.

```
var listItems = jQuery( 'li' );
```

Of course, if you've seen any jQuery code, you're probably more accustomed to seeing something like this:

```
var listItems = $( 'li' );
```

The $ in the code above is just a shorter, more convenient name for the jQuery function

Launching code at the right moment

To run code as soon as the document is ready to be manipulated, jQuery has a statement known as the ready event:

```
$( document ).ready(function() {
    // Your code here.
});
```

Selecting DOM elements

DOM definition: The Document Object Model (DOM) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML, and XML documents. The nodes of every document are organized in a tree structure, called the DOM tree.

jQuery provides a convenient way for selecting the element in the DOM structure.

```
$( '#header' ); // select the element with an ID of 'header'
$( 'li' );      // select all list items on the page
$( 'ul li' );   // select list items that are in unordered lists
$( '.person' ); // select all elements with a class of ‘person'
```

The jQuery selector returns an array of matched elements. So to check if the selection returned any elements, we can use do following:

```
if ( $( '#nonexistent' ).length ) {
  // This code will only run if there's a matching element
}
```

Chaining methods

One of the most lucrative parts of jQuery is the ability to "chain" methods together. This means that we can call a series of methods on a selection without having to repeat the selection or store the selection in a variable.

```
var listItems = $( 'li' );
var spans = listItems.find( 'span' );

listItems
  .click(function() {
    $( this ).addClass( 'clicked' );
  });

spans.attr( 'title', 'Hover over me' );
```

Element Manipulation

jQuery's manipulation methods allow you to alter the DOM of your page using a syntax that's much friendlier than the one provided by native DOM manipulation methods.

Class manipulation

```
$( 'li' ).addClass( 'hidden' );
$( 'li' ).removeClass( 'hidden' );
$( 'li' ).toggleClass( 'hidden' ); // The following code adds the class hidden if it is not present, and removes it if it is present.
```

Changing style

```
$( 'li' ).eq( 1 ).css({
  'font-size': '20px',
  'padding-left': '20px'
});
var color = $( 'li' ).css(‘color’); // returns current color
```

Changing other attributes

```
$( 'a' ).attr( 'href', function(index, value) {
  return value + '?special=true';
});
```

Creating new elements

The $ function has one last role: creating new elements. If you pass an HTML snippet to $(), it will create a new element in memory — that is, the element will be created, but it won't be placed on the page until you place it on the page.

```
$( '<p>' ); // creates a new <p> element with no content
$( '<p>Hello!</p>' ); // creates a new <p> element with content
$( '<p class="greet">Hello!</p>' ); // creates a new <p> with content and class
```

You can also create an element by passing an object with information about how to create the element:

```
$( '<p>', {
  html: 'Hello!',
  'class': 'greet'
});
```

Placing elements in the document

You could append the item to the list by calling .appendTo() on the list item:

```
var listItem = $( '#my-unordered-list li' ).first();
listItem.appendTo( '#my-unordered-list' );
```

Or you could append the item to the list by calling .append() on the list:

```
var listItem = $( '#my-unordered-list li' ).first();
$( '#my-unordered-list' ).append( listItem );
```

Copying and Removing elements

You can make a copy of an element or a set of elements using jQuery's .clone() method.

```
var clones = $( 'li' ).clone();
$( '#my-unordered-list' ).append( clones );
```

To remove an item from the DOM tree, you can use the .remove() method.

```
var removedListItem = $( '#my-unordered-list
li' ).first().remove();
```

Events #1

jQuery makes it easy to respond to user interaction with a web page. This means that you can write code that runs when a user clicks on a certain part of the page, or when she moves her mouse over a form element.

Methods such as .click(), .blur(), .change(), and others are "shorthand" methods for event binding. jQuery provides a number of these shorthand methods, each of which corresponds to a native DOM event:

Native Event Name	Shorthand Method
click	.click()
keydown	.keydown()
keypress	.keypress()
keyup	.keyup()
mouseover	.mouseover()
mouseout	.mouseout()
mouseenter	.mouseenter()
mouseleave	.mouseleave()
scroll	.scroll()
focus	.focus()
blur	.blur()
resize	.resize()

```
$( 'li' ).click(function( event ) {
  console.log( 'clicked', $( this ).text() );
});
```

Events #2

Under the hood, all of the shorthand methods make use of jQuery's .on() method. You can use the .on() method in your own code; indeed, doing so gives you a lot more flexibility. When you use the .on() method, you pass the native event name as the first argument, and then the handler function as the second argument:

```
$( 'li' ).on( 'click', function( event ) {
  console.log( 'clicked', $( this ).text() );
});
```

Exercise: enable submit button when both email and password input values are at least 6 characters long

```
var signIn = $('.btn[type="submit"]');
signIn.attr("disabled", true);
var valid = {
    'email' : false,
    'password' : false
}
$('#inputEmail').on('input', function () {
    if ($(this).val().length > 5) {
        valid.email = true;
        if (valid.email && valid.password) {
            signIn.attr("disabled", false);
        }
    }
});
$('#inputPassword').on('input', function () {
    if ($(this).val().length > 5) {
        valid.password = true;
        if (valid.email && valid.password) {
            signIn.attr("disabled", false);
        }
    }
});
```

Resources

http://jquery.com/

http://jqfundamentals.com/