WA2185 Platform Independent Mobile Development with jQuery Mobile and Apache Cordova

Student Labs

Web Age Solutions Inc.

Table of Contents

Lab 1 - Getting Started with jQuery Mobile

In this lab, we will setup a basic development environment for jQuery Mobile.

Part 1 - A Note About Browsers

For the following labs we recommend that you use the latest version of these browsers:

1. Internet Explorer
2. FireFox
3. Chrome

Some of the labs will not work with very old browser, like IE 6.

Part 2 - Looking at Apache

You will use Apache 2.2.x as the web server for this class. This is already installed for you. Please verify the installation folder. It should be **C:\Program Files\Apache Software Foundation\Apache2.2**

From now on, we will refer to this folder as **APACHE_DIR**.

We will now validate that Apache is working properly.

__1. Using Notepad, open **APACHE_DIR/htdocs/index.html**.

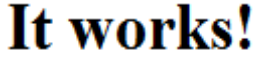
__2. Note that the file currently contains one line:

```
<html><body><h1>It works!</h1></body></html>
```


__3. From a web browser, enter the URL:

http://localhost

__4. Make sure that you see:



Part 3 - Using a Text Editor

You can use any text editor for this class. You can use Notepad or Wordpad. You can also download and install your favorite text editor.

Part 4 - Install jQuery Mobile

jQuery Mobile depends on jQuery so you will need to accees both libraries. There are a few ways you can use jQuery Mobile.

1. Over the web from a content delivery network (CDN).
2. From your web server. Multiple web applications can share the same jQuery Mobile installation.
3. Embedded within each web application. This way, each application can independently use a different version of jQuery.

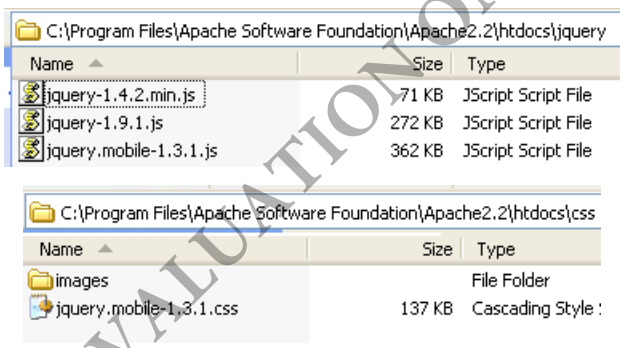
We will take the second approach for these labs.

In Windows XP and Vista, you may have to be an Administrator to create folders within APACHE_DIR.

__1. Copy **C:\LabFiles\jquery** to **APACHE_DIR/htdocs**.

__2. Copy **C:\LabFiles\css** to **APACHE_DIR/htdocs**.

__3. Examine the contents of the directories you just copied.



Part 5 - Write a Basic jQuery Application

__1. In the **APACHE_DIR/htdocs** folder, create a new file called **test.html**

__2. Open the file with an editor and write the basic structure.

```
<!DOCTYPE HTML>
<html>

  <head>
  </head>

  <body>
  </body>

</html>
```

__3. Import jQuery Mobile JavaScript and CSS files as shown in bold face.

```
<head>
  <link rel="stylesheet" href="css/jquery.mobile-1.3.1.css" />
  <script src="jquery/jquery-1.9.1.js"></script>
  <script src="jquery/jquery.mobile-1.3.1.js"></script>
</head>
```

__4. Add a tag with the special jQuery Mobile attributes as shown. Then add an empty paragraph within the .

```
  <body>
    <div data-role="page" data-theme="b">
      <p id="p1"/>
    </div>
  </body>
```

__5. Below the existing <script> tag, add another one.

```
<script type="text/javascript">
  $(function(){
    $("#p1").text("Hello, World!");
  });
</script>
```

We will inspect the code later. For now, continue with testing.

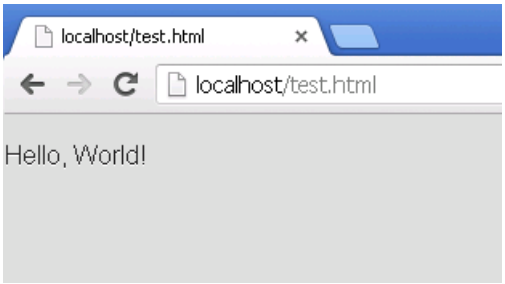
__6. Save changes.

Part 6 - Test

__1. From the browser, enter the URL:

```
http://localhost/test.html
```

__2. Make sure that you see:



Part 7 - What Just Happened?

All the magic happens in the script:

```
<script type="text/javascript">
  $(function(){
    $("#p1").text("Hello, World!");
  });
</script>
```

First, let's talk about the core function:

```
$()
```

Almost all jQuery API is accessible through this one single function. You can also call the core function through these names:

```
jQuery()
window.jQuery()
```

But, $() is short and sweet and that's what most developers use.The core function behaves completely differently depending on the input parameters you pass to it. In our case, we passed an anonymous function:

```
function(){
    $("#p1").text("Hello World");
}
```


When you pass a function as input argument, jQuery treats it as a DOM ready event handler. This event is fired when the DOM document of the page is fully ready. This can happen before the "onload" event of the body. For example, if there are many images in the document, the DOM ready event fires before all the images are actually downloaded.

You can start manipulating the DOM as soon as the DOM is ready.

When the DOM document is ready, FireFox fires the “DOMContentLoaded” event and IE the “onreadystatechange” event for the document object. jQuery makes it easy to attach an handler for that event in a browser independent manner.

Now, let's look at what our DOM ready event handler function does. Firstly, we see that the core function is used again.

```
$("#p1")
```

This time, the core function is taking a string as an argument. This signifies to the core function that the argument is an element selector. jQuery borrows heavily from CSS selector model. According to CSS, "#p1" select a DOM element with the ID of "p1". That will be the empty paragraph we had added earlier.

$("#p1") will return a jQuery object for the paragraph. We then call the text() function for the object. This sets the inner text of the element to "Hello World".

What does the core function - $("#p1") – actually return? It returns a jQuery object that is a collection of all DOM element objects that match the supplied selector rule. This jQuery object builds on top of the JavaScript array API. For example, you can obtain the length:

$("p").length; //Number of paragraphs in the page.

To obtain a DOM element object in the collection, simply call get(index).

alert($("p").get(1).innerHTML); //Show inner HTML of the second paragraph.

When you invoke a method of the collection object, jQuery internally applies it to all DOM elements in the collection:

$("p").text("Hello!"); //Change text of all paragraphs to Hello!.

A getter function normally works on the first DOM element in the collection only.

$("p").text(); //Returns the text of the first paragraph only.

To convert a DOM element object into a jQuery collection do:

$(elementObject)

For example:

$(document.getElementById("p1")); //Same as $("#p1").

Part 8 - Something More Advanced

So far, $("#p1") returned a single object. This expected since an element always has a unique ID. Now, we will use selector to return a collection of jQuery objects and work on them.

__1. Below the current paragraph add another one as shown in bold face below.

```
<p id="p1"/>
<p id="p2"/>
```

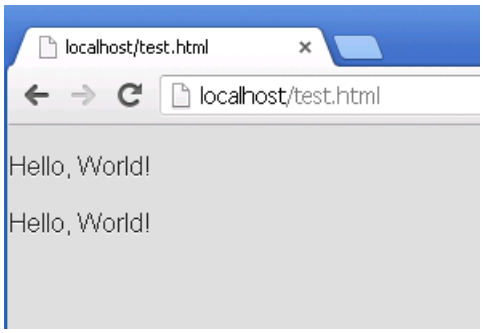
__2. Change the DOM ready event handler as shown below.

```
$("p").text("Hello, World!");
```

Now, $("p") will select all DOM elements with a <p> tag. Once again, jQuery borrows this from the CSS selector model.

__3. Save changes.

__4. Refresh the browser. Now, both paragraphs will say "Hello World".



What this means is that the text() function was called for the entire collection of jQuery objects returned by $("p"). Trying to do this using raw JavaScript will take quite a few lines of tedious coding.

__5. Close all open editors and browers.

Part 9 - Review

In this lab, we installed jQuery and wrote a very basic application. We learned about the core function $(). We wrote a DOM ready handler. We used a simple ID based selector first. And then a tag based selector.

Lab 2 - Exploring Mobile UI Patterns

When designing the pages, it is essential that we follow the well regarded UI patterns for mobile devices. This is more crucial for mobile than desktop. The patterns offer guidelines for most optimal user interactions in devices where size and input methods are limited.

We will use the jQuery Mobile toolkit. As we will see here, using a toolkit can significantly speed up mobile web site development.

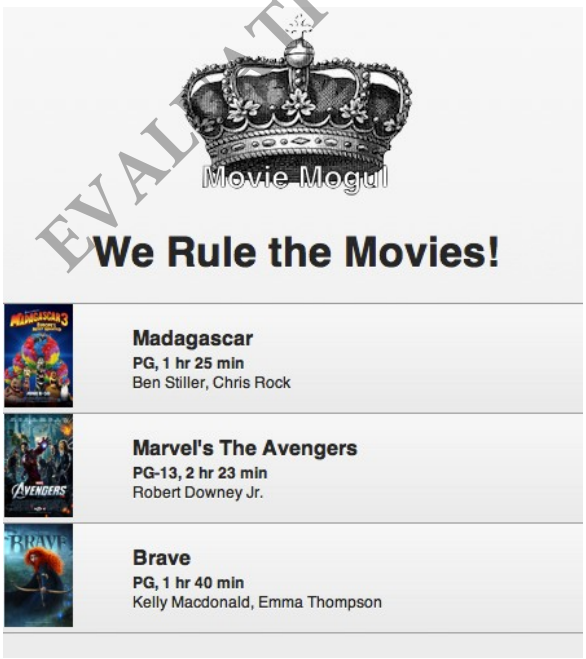
In this lab, we will explore List View and Master Detail Navigation patterns.

Part 1 - Business Requirement

We will build a web site for the startup company called Movie Mogul. The site offers information about all movies currently in theater or coming soon. Users can view images and trailers. The site shows showtimes based on user's current location.

In real life, this will be a database driven application. To do that in this lab will take the focus away from UI design to application programming. Hence, we will choose to create static web pages only. These static files can be easily converted to JSP, ASP, PHP or any other template.

Part 2 - Create the Home Page



The home page will have these two functions:

- It will show some branding information, like the logo and a blurb – "We Rule the Movies!". Branding takes up valuable screen real estate. This is something we

will do only for the home page and not for the inner pages.

- It will list the movies using the List View pattern. In the first phase (this lab), we will show only the movies currently playing in theater.

In this part, we will create the home page HTML file and get the branding content done.

__1. Open Windows file explorer.

__2. Copy the **C:\LabFiles\mm** folder to **APACHE_DIR\htdocs**

__3. Open **APACHE_DIR\htdocs\mm\index.html** in your favorite editor.

__4. Notice some of the boiler plate code already present here:

- The DOCTYPE is for HTML5.
- The "viewport" meta tag is already added. This is essential for many things to work in Android and iOS browsers.
- The jQuery Mobile toolkit files are imported as follows.

```
<link rel="stylesheet" href="/css/jquery.mobile-1.3.1.css" />
<script src="/jquery/jquery-1.9.1.js"></script>
<script src="/jquery/jquery.mobile-1.3.1.js"></script>
```

We will now start adding content for the home page. The concept of a "page" is a little different for jQuery Mobile than what we are used to. We can create multiple pages using tags within the same HTML file. jQuery Mobile shows and hides the page contents as the user interacts with the site. The main reason for creating multiple pages in the same HTML file is to download several pages in one shot. This improves performance (screen updates during site navigation can be done quickly). At the same time, be careful about the total size of the HTML file, considering the slow speed of the mobile networks.

Architecture Tip

Use multiple pages within the same HTML file when the sizes of these pages are small. If a page has long content, create only one jQuery Mobile "page" within the HTML file. Advanced applications can even ask jQuery Mobile to pre-fetch that page in background to speed up the UI update.

__5. Within the <body> tag, add these lines to define a "page".

```
<div data-role="page" id="home" data-theme="c">

</div>
```

The **data-role** and **data-theme** attributes are introduced by jQuery Mobile and have the following meaning:

- data-role – This designates a special behavior to the normal HTML tags. This is used to define the widget type. In this example, we are using the div tag as a page.
- data-theme – This sets the color scheme. There are five pre-defined themes – a, b, c, d and e.

Now, we will add the branding content. But, first let's discuss the anatomy of a page. In jQuery Mobile a page element has three well defined sections:

- Header – Displayed at the top.
- Content area – Occupies the middle portion of the screen.
- Footer – Displayed at the bottom.

In our home page, we only have the content area. We will create that now.

__6. Within the page div tag, add the content area:

```
<div data-role="content">

</div>
```

__7. Within the content div tag, add the branding content:

```
<div align="center">
<img src="assets/crown.png"/><br/>
<h1>We Rule the Movies!</h1>
</div>
```

This is pretty simple stuff and nothing special about it.

__8. Save changes.

Part 3 - Test

jQuery Mobile can be tested using a desktop browser. For better testing, you can use the browser from an emulator or better yet, a real device.

__1. Open a WebKit based browser like Chrome or Safari. They will closely mimic the browsers in Android and iOS.

__2. Enter the URL:

```
http://localhost/mm/
```

__3. Even for such simple content, jQuery Mobile is playing a role:

- The background is light grey as per the theme used for the page. (Feel free to experiment with theme b or e. But switch back to c to stay in sync with these labs).
- The <h1> tag has been styled with a nice embossed look.

Design Tip

Limit the height of the branding content. Users should be able to view good amount of actual content without having to scroll.

Part 4 - Display Movie List

We will now use a vertically scrollable list to show the movies. This is basically the List View pattern.

__1. Below the end tag of the branding content, add these lines:

```
<ul data-role="listview">

</ul>
```

```
<body>
<div data-role="page" id="home" data-theme="c">
<div data-role="content">
<div align="center">
<img src="assets/crown.png"/><br/>
<h1>We Rule the Movies!</h1>
</div>
<ul data-role="listview">

</ul>
</div>
</div>

</body>
```

That's all it takes to add a List View widget for jQuery Mobile.

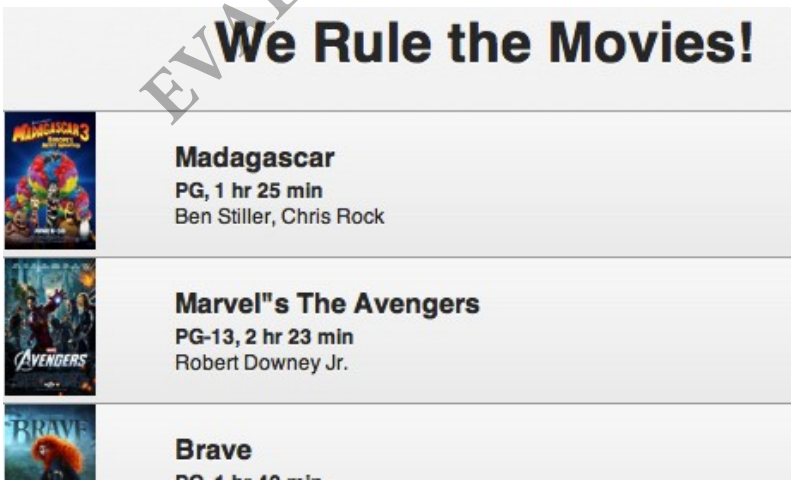
__2. Within the <ul> tag, add three rows using <li> tag as follows.

```
<li>
<img src="assets/madagascar.jpg"/>
<h3>Madagascar</h3>
<p><b>PG, 1 hr 25 min</b></p>
<p>Ben Stiller, Chris Rock</p>
</li>

<li>
<img src="assets/avengers.jpg"/>
<h3>Marvel&apos;s The Avengers</h3>
<p><b>PG-13, 2 hr 23 min</b></p>
<p>Robert Downey Jr.</p>
</li>

<li>
<img src="assets/brave.jpg"/>
<h3>Brave</h3>
<p><b>PG, 1 hr 40 min</b></p>
<p>Kelly Macdonald, Emma Thompson</p>
</li>
```

__3. Save changes.

Part 5 - Test

__1. Refresh the browser that was showing http://localhost/mm/.



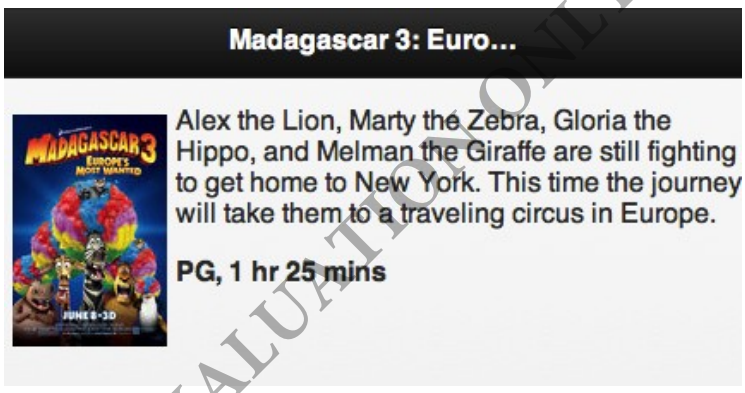
__2. Note the following aspects of the List View pattern:

- If a row has a thumbnail image, it is displayed at the far left end. If the first child of a <li> is an <img>, jQuery Mobile automatically uses it as a thumbnail image. All thumbnail images are resized to a consistent size.
- The main text of the row needs to stand out. In jQuery Mobile, we use a header (h1 to h3) for the main text. The detailed text, if present, is modeled using a plain <p> tag. Be consistent about this in all list views of the application.
- Although, strictly not a part of the pattern, jQuery Mobile is using a nice graduated background for the rows.

Part 6 - Implement Movie Details Page

We will now use Master Detail navigation pattern to show details about a movie. We will design the movie design page first. We will then link to it from the home page.

The layout of the page will look like this.



__1. In the **mm** folder of the document root, copy **template.html** and paste it as **madagascar.html**.

__2. Open **madagascar.html** in text editor.

__3. Within the <body> tag add a for the page.

```
<div data-role="page" id="details" data-theme="c">

</div>
```

A details page should show a title at the top. This keeps the user aware of where they are in the navigation flow.

__4. Within the for the page, add a header as follows.

```
<div data-role="header">
    <h2>Madagascar 3: Europe&amp;apos;s Most Wanted</h2>
</div>
```

Notice the data-role value here. Also, normally, you should use a head tag like h2 or h3 here. jQuery Mobile will then center the text and apply ellipsis to truncate long text.

We will now show the image and description of the movie side by side. This will save vertical space, a technique used sometimes in mobile sites. Two tags can be positioned side by side by using a float style of "left" for the left div.

__5. Below the header div, add these lines:

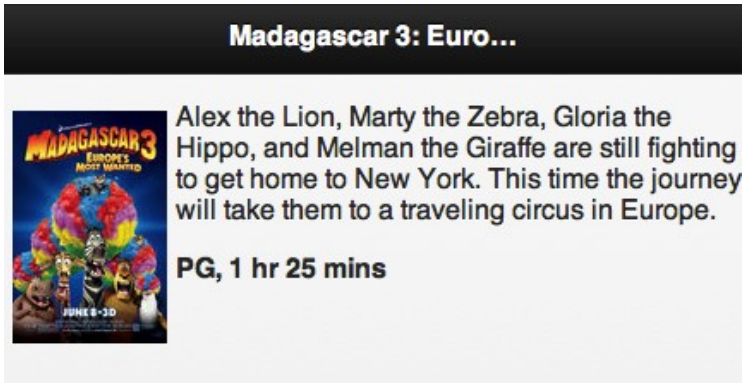
```
<div data-role="content">
<div>
  <div style="float:left;margin:5px;">
    <img src="assets/madagascar.jpg"/>
  </div>

  <div style="margin:5px;">
    <p>Alex the Lion, Marty the Zebra, Gloria the Hippo, and Melman the
Giraffe are still fighting to get home to New York. This time the
journey will take them to a traveling circus in Europe.</p>

    <p><b>PG, 1 hr 25 mins</b></p>
  </div>
</div>
</div>
```

__6. Save changes.

Part 7 - Test

__1. In the browser, enter the URL:

```
http://localhost/mm/madagascar.html
```



__2. Verify that it looks like above.

Part 8 - Implement Master Detail Navigation

We will now link the details page from the home page.

__1. Open **index.html** in the editor.

__2. Add a hyperlink for the row for the Madagascar movie as shown in bold face below:

```
<li>
<a href="madagascar.html">
<img src="assets/madagascar.jpg"/>
<h3>Madagascar</h3>
<p><b>PG, 1 hr 25 min</b></p>
<p>Ben Stiller, Chris Rock</p>
</a>
</li>
```

__3. Save changes.

As recommended by the Master Detail pattern, a detail page should have a back button to go back to the parent. This just gives a better orientation to users about where they are in the navigation and how to navigate back. Adding a back button is easy in jQuery Mobile.

__4. Open **madagascar.html** in editor.

__5. For the tag of the page, add the attribute shown in bold face below.

```
<div data-role="page" id="details" data-theme="c"
  data-add-back-btn="true">
```

Now the page will show a back button in the header.

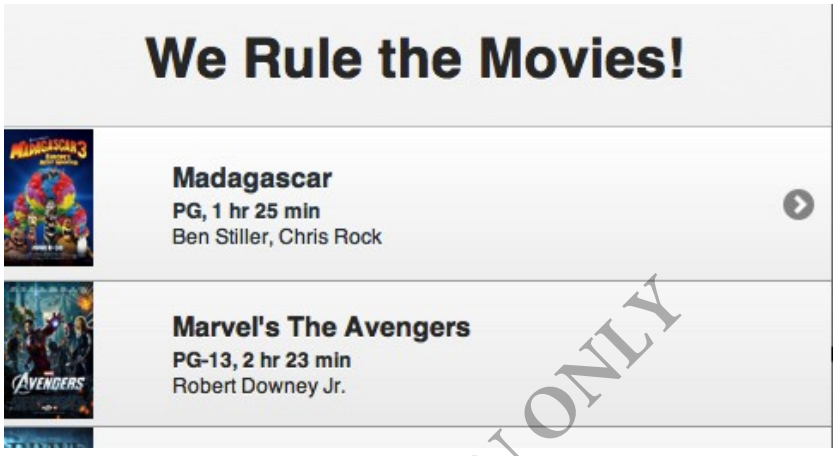
__6. Save changes.

Part 9 - Test

__1. Close all browser windows. Navigation may not work properly if browser has pages cached.

__2. Open a new WebKit based browser. Enter the URL:

```
http://localhost/mm/
```



__3. Notice, now the row with the hyperlink has a indicator accessory icon to the right. This is a hint to the user that tapping the row will show next level of details.

__4. Click the row for Madagascar.

__5. Note the back button in the header.

__6. Click the **Back** button and make sure you go back to the home page.

Part 10 - Apply Page Transition

Although animated page transition doesn't really help the user in any way, this is something done to liven up the experience. A simple right to left transition is good enough for master detail navigation. We will get that done now.

__1. Open **index.html** in editor.

__2. In the hyperlink for madagascar.html, add transition behavior as shown in bold face below.

```
<a href="madagascar.html" data-transition="slide">
```

__3. Save changes.

Part 11 - Test

__1. Close all open browsers.

__2. Open a new WebKit browser and enter the URL:

```
http://localhost/mm/
```

__3. Click the row for Madagascar. Make sure that details page slides in from right to left.

__4. Click the **Back** button from the details page. Make sure that the home page slides in from left to right.

Part 12 - Show Movie Trailer

We have already seen how to use the HTML5 <video> tag to embed video in a page. When you play the video, most mobile browsers will go fullscreen for the playback.

In many situations, it is better to just add a hyperlink to the movie file. When user taps on the link, the browser starts playing the movie in fullscreen. Advantage of this approach is that there is no rectangular space wasted to show the <video> element. We will take this approach now.

__1. Open **madagascar.html** in editor.

__2. Below the line:

```
<p><b>PG, 1 hr 25 mins</b></p>
```

Add these lines:

```
<p align="center">
  <a href="assets/madagascar.mp4" data-role="button" data-theme="b"
    data-inline="true" rel="external">Play Trailer</a>
</p>
```

There is a lot going on here. Let's go through all the attributes:

- data-role – We turn the <a> tag into a button. jQuery Mobile creates a nice sized button which makes it easy for people to tap on.
- data-theme – We render the button with a blue background to make it stand out a little.
- data-inline – We make the width of the button compact. Without this, the button will take up the whole width of the device.
- rel – Normally, jQuery Mobile fetches the linked page of a hyperlink using Ajax. Here, we need to have the browser play the MP4 movie file. By, setting rel="external", we are asking jQuery Mobile not to use Ajax and simply have the browser go to the linked document.

__3. Save changes.

Part 13 - Test

__1. Close all browsers.

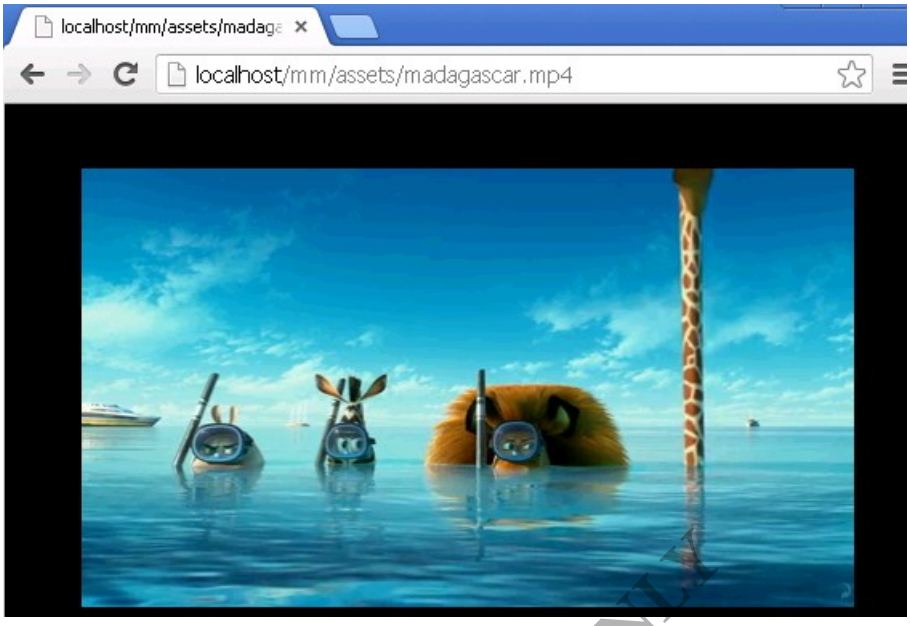
__2. Open a WebKit browser and enter the URL:

```
http://localhost/mm/
```

__3. Click the row for Madagascar.

__4. Click the **Play Trailer** button.

__5. Make sure that the browser begins to play the movie.



__6. Note, running remotely or from a VMWare image you may have problems to listen the movie.

__7. Close the browser and all open editors.

Part 14 - Review

In this lab, we started to build a realistic mobile web site. A key part of any mobile web design is UI patterns. You have to carefully go through the patterns catalog and choose the patterns applicable to your scenario. In the Movie mogul web site, we used the List View and Master Detail patterns. Also, we used jQuery Mobile to speed up development.

Lab 3 - More on Selectors

In this lab, we will learn a few more advanced uses of selectors. We will also learn about how to manipulate the CSS styles of elements.

Part 1 - Iterating Over a Selected Collection

We know that the core function returns a collection of DOM element objects. We can iterate over the collection using the each() function.

__1. Open **test.html** under **APACHE_DIR/htdocs/**.

We currently have two paragraphs in index.html.

```
<p id="p1"></p>
<p id="p2"></p>
```

Let us say that we want to iterate over them and change their text.

__2. Change the line:

```
$("p").text("Hello World");
```

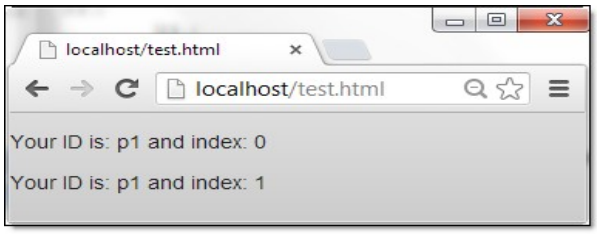
As follows.

```
$("p").each(function (index) {
    $(this).text("Your ID is: " + this.id + " and index: " + index);
});
```

__3. Save changes.

__4. Open a browser and enter the URL:

http://localhost/test.html



There is a lot going on with the each() function. It takes as argument a function object. This function is called repeatedly, once for each jQuery object in the collection. The function receives the index of the jQuery object in the collection starting with 0.

Important: The "this" keyword within the iteration function points to the DOM element. We can not directly call the text() method for it since that is not a DOM API. We must obtain the jQuery object using $(this) and then call the text() function to set the text.

Since, "this" is a DOM element, we can call this.id to get it's ID.

Part 2 - Class Selector

One of the common uses of selector is to select all elements that belong to a CSS class. This is done using the format $(.class_name). This is same as the CSS class selector.

Now, we will select one of the paragraphs by its class name and change its background color.

__1. Set the class of the second paragraph to "highlighted".

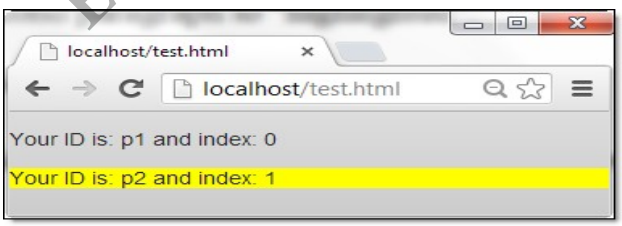
```
<p id="p2" class="highlighted"></p>
```

__2. At the end of the DOM ready handler function add the line shown in bold face.

```
$(function(){
    $("p").each(function (index) {
        $(this).text("Your ID is: " + this.id + " and index: " + index);
    });
    $(".highlighted").css("background", "yellow");
});
```

__3. Save changes.

__4. Refresh the browser. The second paragraph will now have a yellow background.



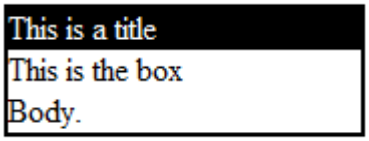
This was also a good example of how to tweak the style of an element using the css() function.

If you know that it is a "p" tag that has the "highlighted" class, you can help jQuery a little by using $(p.highlighted). This will have a performance improvement.

Part 3 - Child Pseudo Selector

So far, we have been using selectors that follow the same syntax as the CSS selector. Now, we will use pseudo selectors that are introduced by jQuery. A pseudo selector is always prefixed by ":".

We will now use jQuery to create a box widget that looks like this.



__1. Copy **C:\LabFiles\template.html** to **APACHE_DIR/htdocs/**.

__2. Rename **template.html** to **box.html** under APACHE_DIR/htdocs/ folder.

__3. Open **box.html** in an editor.

__4. Below the line:

```
<!-- Body HTML here -->
```

Add:

```
<div class="box">
<div>This is a title</div>
<div>
This is the box<br/>
Body.
</div>
</div>
```


Basically, any element with style "box" will be converted into a box. The first child element of the box will be converted into the title.

First, we will draw the border around the box.

__5. Below the line:

```
//jQuery code here
```

Add:

```
$("div.box").css("border-style", "solid");
$("div.box").css("border-width", "2px");
```

Now, we will change the background color of the first child element to black. We will change its foreground to white.

If we use $("div div") that will select all div child elements of another div. We don't want that. We only wish to select the first child.

__6. Continue to develop the DOM ready function by adding these lines:

```
$("div.box div:first-child").css("background", "black");
$("div.box div:first-child").css("color", "white");
```

Here, we use the "first-child" pseudo selector. This will select the first child element of any div element that has the box class.

__7. Save changes.

__8. In the browser, enter the URL:

http://localhost/box.html

__9. You should see the page as follows.

It's now easy to add as many boxes as we want.

__10. Below the last tag, enter:

```
<p>Some other content</p>

<div class="box">
<div>Company News</div>
<ul>
          <li>Stock price has gone up</li>
          <li>Tomorrow is a holiday</li>
</ul>
</div>
```

__11. Save changes.

__12. Refresh the browser.

Part 4 - Optimize the Code

Selectors require a lot of DOM tree navigation. This can be slow for a large document. Once you have retrieved a collection, you should reuse it as much as possible. In our case, we are running the same selector multiple times. We should be able to optimize that.

__1. Change the DOM ready function as follows.

```
$(function(){
	//jQuery code here
	var box = $("div.box");
	box.css("border-style", "solid");
	box.css("border-width", "2px");
	var title = $("div.box div:first-child");
	title.css("background", "black");
	title.css("color", "white");

});
```


Now, a selector is run only once.

__2. Save changes.

__3. Refresh the browser. It will look same as before.

Part 5 - Pseudo Selector for Table Rows

jQuery has :odd and :even pseudo selectors that apply to table rows. They are great for styling alternate rows differently.

__1. Copy **C:\LabFiles\table.html** to **APACHE_DIR/htdocs/**.

__2. Open **APACHE_DIR/htdocs/table.html** in your editor.

__3. Note that the header is created using <thead> and the body using <tbody>.

__4. View the page in a browser:

http://localhost/table.html

Name	Age
John Doe	19
Jane Doe	21
Mary Doe	22
Superman	222

Nothing spectacular. Now, we will apply different colors to every alternate rows.

__5. Below the line:

```
//jQuery code here
```

Add:

```
$("tr:even").css("background-color", "grey");
$("tr:odd").css("background-color", "yellow");
```

__6. Save.

__7. Refresh the browser.

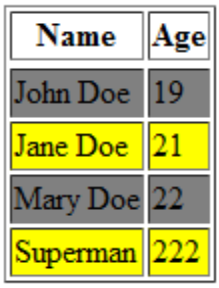
Name	Age
John Doe	19
Jane Doe	21
Mary Doe	22
Superman	222

It worked. But, the <tr> elements within <thead> are also being colored. We can fix that by combining the even and odd selector with child selector. We want only the <tr> elements that are children of <tbody> to be colored.

__8. Change the code as follows.

```
$("tbody tr:even").css("background-color", "grey");
$("tbody tr:odd").css("background-color", "yellow");
```

__9. Save and test again.



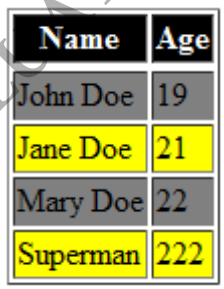
Name	Age
John Doe	19
Jane Doe	21
Mary Doe	22
Superman	222

Now the header is not colored any more. In fact, while we are at it, let's style the header row.

__10. Add this code.

```
$("thead tr").css("background-color", "black");
$("thead tr").css("color", "white");
```

__11. Save and test.



Name	Age
John Doe	19
Jane Doe	21
Mary Doe	22
Superman	222

__12. Close all your editors and web browsers.

Part 6 - Review

In this lab we played with more selectors. We also learned how to apply basic styling using the css() function. As you saw, with only a few lines of code, you can apply consistent look and feel throughout the site.