Lecture 2.7: JavaScript Libraries and jQuery

Motivation and advantages of using JavaScript Libraries

jQuery is a cross-browser Open Source JavaScript library that helps web developers to save time and create robust code when writing JavaScript. jQuery simplifies HTML document traversing, event handling, animation, and Ajax interactions for rapid development of interactive websites. jQuery was created by John Resig in 2005 in an attempt to simplify the management of cross-browser issues.
Mastering HTML and CSS is not enough to transform inspiring design ideas into gorgeous websites that enthral visitors. JavaScript is how you add complex behaviours, sophisticated interactions, and extra excitement to your web pages. However, writing JavaScript each time you create a new site can take a lot of effort and time. The advisable path is using suitable JavaScript libraries, created by JavaScript experts. Among those libraries, “just using jQuery” generally is the best option. Designers and developers over the world are using CSS and jQuery to elegantly and rapidly implement their interaction ideas. jQuery makes it easy client-side scripting. In fact, it makes it so easy that it’s downright good, nerdy fun. In addition jQuery offers complete cross-browser compatibility. Anyone who has written serious JavaScript in the past can attest that cross-browser inconsistencies will drive you mad. jQuery works around the caveats of cross-browser issues for you. Most of the code you write with jQuery will run exactly the same on all the major browsers, including Internet Explorer 6. Leaving the task of hunting down obscure browser bugs to the jQuery Team allows you more time to concentrate in implementing your ideas. Using the jQuery supports utility, you can test to see if a certain feature is available to the user, and easily build applications that degrade gracefully on older browsers. jQuery also supports all the CSS versions, including the CSS3 selector specification. Selecting elements you want to change lies at the heart of jQuery’s power. An additional advantage is that jQuery includes an assortment of utility functions that implementing common functions with JavaScript becomes trivial: string trimming, extending objects, chaining methods, etc. These functions by themselves are particularly handy, but they help promote a seamless integration between jQuery and JavaScript that results in code that’s easier to write and maintain.

Thanks to its ability to easily hook elements on the page and attach code to them in a natural CSS-like manner, jQuery enables the implementation of best practice of separation of page presentation from functionality. This separation of concerns leads to leaner, cleaner, and maintainable code. jQuery isn’t limited to meddling with a page’s existing HTML; it can also add new page elements and document fragments via a collection of useful functions. There are functions to insert, append, and prepend new chunks of HTML anywhere on the page. You can even replace, remove, or clone existing elements: all functions that help you to progressively enhance your sites, thus providing a fully featured experience to users whose browsers allow it, and an acceptable experience to everyone else. jQuery is known by its “Graceful Scripting”, because it gracefully will fail when it can't find anything to run against. In this sense, it works just like CSS - write your queries to match what you want, doesn't break when nothing is found.

Last but not least is its widespread adoption. The last year alone has witnessed jQuery's exponential rise to prominence. 67% of the top 10,000 websites, including Google, Microsoft, Amazon, BMW, IBM and Intel are using jQuery (source: BuiltWith's graphs). This is good for jQuery, as popularity equates to more active code development and plenty of interesting third-party plug-ins. jQuery's popularity has also spawned a large and generous community that's surprisingly helpful. No matter what your level of skill is, you'll find plenty tutorials, blog posts, and documentation as well as developers patient enough to help you out and work through any issues you have. jQuery is the beauty of Open Source in full.

In spite of its power, jQuery is lightweight. Its core size can be less than 40Kb, less than the average JPG image size. You can add any extras and plug-ins in a modular fashion, only adding what you need. Moreover, every new release of jQuery is faster than the previous one. In summary, when it comes to competing JavaScript libraries, today jQuery is the best at manipulating the DOM, adding effects, and making Ajax requests. Still, many of the libraries are of excellent quality and it's always worth looking at the alternatives, but in general you will find that jQuery is the way to go by now.

jQuery Essentials

Adding jQuery to Your Web Pages

There are a couple of ways to use jQuery on your web site. You can download the jQuery library from jQuery.com or you can just include jQuery from a CDN (Content Delivery Network), like Google or Microsoft. If you decide to download jQuery, you can choose from the Production version (it has been minified and compressed) and the Development version (this is an uncompressed and readable code for testing and development.)

The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag (normally the <script> tag should be inside the <head> section, but see the lecture slides for further discussion). Example of using a local copy of jQuery:

```
<head>
<script src="jquery-3.2.1.min.js"></script>
</head>
```

Example of using the Google CDN option (advisable for the reasons discussed in class):

```
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
</head>
```

How jQuery works

In order to traverse and manipulate the page we must wait until the page has uploaded on the browser and it's ready to be used. jQuery has a **ready event** that fires the instant the

DOM is ready to be worked with. You should normally put all your jQuery code in a ready block, so it can be executed after the DOM has loaded unobtrusively.

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

jQuery works in a very natural way. You find some elements and then do something with then. You can change the style of those elements (e.g. *addClass*), manipulate their content (e.g. *clone, append, remove*), handle events (e.g. *hover, toggle*), produce effects (e.g. *hide, slideDown, fadeout*) or do ajax calls (e.g. *load, get, post*). Example:

```
$("div").addClass("special");
```

Where:

`$` is the jQuery Object (also named `jQuery`)

"`div`" identifies all the *div* elements found in the page

`$("div")` returns a jQuery set (containing 0 to many *div* elements).

`addClass`(...) adds the class *special* to all the *div* elements

jQuery Selectors

jQuery selectors are used to “select” HTML elements using their name, id, class, types, attributes, and other conditions implemented by jQuery. The jQuery selectors are based on the standard W3C CSS Selectors. In addition jQuery has its own custom selectors that make developing ever easier (e.g. *:first, :last, :has(), :visible, :hidden*).

The capabilities offered by the CSS selectors are complemented with a full suite of methods for walking the DOM tree, such as:

```
    .parent(), .next(), .prev(), .children(), .siblings()
```

In jQuery, walking the DOM tree or HTML document with those method is known as traversing. For example:

```
$("button").parent().css("border", "3px solid red");
```

jQuery provides four ways to operate with child and sibling selectors: descendant selector (A B), child selector (A > B), adjacent sibling selector (A + B) and general sibling selector (A ~ B). A and B are selectors. Descendant selector is used to select all elements matched by B that are child, grandchild, great-grandchild, great-great-grandchild, etc. of A. Child selector is used to select all elements matched by B that are child of A. Adjacent sibling selector is used to select the immediately following next element matched by B that is a sibling of A. General sibling selector is used to select all the elements siblings of A matched by B.

jQuery Selectors

Selector	**Example**	**Selects**
*	$("*")	All elements
#*id*	$("#lastname")	The element with id="lastname"
.class	$(".intro")	All elements with class="intro"
.class,.class	$(".intro,.demo")	All elements with the class "intro" or "demo"
element	$("p")	All <p> elements
el1,el2,el3	$("h1,div,p")	All <h1>, <div> and <p> elements
:first	$("p:first")	The first <p> element
:last	$("p:last")	The last <p> element
:even	$("tr:even")	All even <tr> elements
:odd	$("tr:odd")	All odd <tr> elements
:first-child	$("p:first-child")	All <p> elements that are the first child of their parent
:first-of-type	$("p:first-of-type")	All <p> elements that are the first <p> element of their parent
:last-child	$("p:last-child")	All <p> elements that are the last child of their parent
:last-of-type	$("p:last-of-type")	All <p> elements that are the last <p> element of their parent
:nth-child(n)	$("p:nth-child(2)")	All <p> elements that are the 2nd child of their parent
:nth-last-child(n)	$("p:nth-last-child(2)")	All <p> elements that are the 2nd child of their parent, counting from the last child
:nth-of-type(n)	$("p:nth-of-type(2)")	All <p> elements that are the 2nd <p> element of their parent
:nth-last-of-type(n)	$("p:nth-last-of-type(2)")	All <p> elements that are the 2nd <p> element of their parent, counting from the last child
:only-child	$("p:only-child")	All <p> elements that are the only child of their parent
:only-of-type	$("p:only-of-type")	All <p> elements that are the only child, of its type, of their parent
parent > child	$("div > p")	All <p> elements that are a direct child of a <div>

		element
parent descendant	$("div p")	All <p> elements that are descendants of a <div> element
element + next	$("div + p")	The <p> element that are next to each <div> elements
element ~ siblings	$("div ~ p")	All <p> elements that are siblings of a <div> element
:eq(index)	$("ul li:eq(3)")	The fourth element in a list (index starts at 0)
:gt(no)	$("ul li:gt(3)")	List elements with an index greater than 3
:lt(no)	$("ul li:lt(3)")	List elements with an index less than 3
:not(selector)	$("input:not(:empty)")	All input elements that are not empty
:header	$(":header")	All header elements <h1>, <h2> ...
:animated	$(":animated")	All animated elements
:focus	$(":focus")	The element that currently has focus
:contains(text)	$(":contains('Hello')")	All elements which contains the text "Hello"
:has(selector)	$("div:has(p)")	All <div> elements that have a <p> element
:empty	$(":empty")	All elements that are empty
:parent	$(":parent")	All elements that are a parent of another element
:hidden	$("p:hidden")	All hidden <p> elements
:visible	$("table:visible")	All visible tables
:root	$(":root")	The document's root element
:lang(language)	$("p:lang(de)")	All <p> elements with a lang attribute value starting with "de"
[attribute]	$("[href]")	All elements with a href attribute
[attribute=value]	$("[href='default.htm']")	All elements with a href attribute value equal to "default.htm"
[attribute!=value]	$("[href!='default.htm']")	All elements with a href attribute value not equal to "default.htm"
[attribute$=value]	$("[href$='.jpg']")	All elements with a href attribute value ending with ".jpg"
[attribute|=value]	$("[title|='Tomorrow']")	All elements with a title attribute value equal to

		'Tomorrow', or starting with 'Tomorrow' followed by a hyphen
[attribute^=value]	$("[title^='Tom']")	All elements with a title attribute value starting with "Tom"
[attribute~=value]	$("[title~='hello']")	All elements with a title attribute value containing the word "hello"
[attribute*=value]	$("[title*='hello']")	All elements with a title attribute value containing the string "hello"
:input	$(":input")	All input elements
:text	$(":text")	All input elements with type="text"
:password	$(":password")	All input elements with type="password"
:radio	$(":radio")	All input elements with type="radio"
:checkbox	$(":checkbox")	All input elements with type="checkbox"
:submit	$(":submit")	All input elements with type="submit"
:reset	$(":reset")	All input elements with type="reset"
:button	$(":button")	All input elements with type="button"
:image	$(":image")	All input elements with type="image"
:file	$(":file")	All input elements with type="file"
:enabled	$(":enabled")	All enabled input elements
:disabled	$(":disabled")	All disabled input elements
:selected	$(":selected")	All selected input elements
:checked	$(":checked")	All checked input elements

Further Reading

Developing Web Pages with jQuery by Gosselin Dom

http://jquery.com/
http://en.wikipedia.org/wiki/Content_delivery_network
http://trends.builtwith.com/javascript/jQuery
http://www.w3.org/TR/CSS21/selector.html
http://api.jquery.com/category/selectors/
http://api.jquery.com/category/selectors/jquery-selector-extensions/

jQuery - Events

Event handling with jQuery

We have the ability to create dynamic web pages by using events. Events are actions that can be detected by the web application. Following are examples of HTML events:

- A mouse click
- A web page loading
- Taking mouse over an element
- Submitting an HTML form
- An image is loaded
- An input field is changed
- etc.

When these events are triggered we can execute a JavaScript function to do something with the event. These custom functions are called Event Handlers. jQuery provides the **`on()`** method to attach an event handler to one or more events. **`off()`** removes an event handler.

```
.on( events [, selector ] [, data ], handler(eventObject) )
```

Where:

`events`: One or more space-separated event types and optional namespaces, such as "click"

`selector`: A selector string to identify the selected elements that trigger the event.

`data`: Data to be passed to the handler in *event.data* (a jQuery object) when an event is triggered.

`handler(eventObject)`: A function to execute when the event is triggered. The value *false* is also allowed as shorthand for a function that simply does return *false*.

For example the following code will cause the division element to respond to the *click* event; thus, when a user clicks inside this division, the alert will be shown:

```
$('div').on('click', function( event ){
   alert('Hi there!');
});
```

jQuery uses shorthand event types that referencing to the **`on()`** method. Each of these shorthand event types correspond to a native DOM event:

`.click()`	`.keydown()`	`.keypress()`
`.keyup()`	`.mouseover()`	`.mouseout()`
`.mouseenter()`	`.mouseleave()`	`.scroll()`
`.focus()`	`.blur()`	`.resize()`

The shorthand event types also are associated with corresponding shorthand methods. For example "*click*" is associated with *click*(). You can use the shorthand methods, but internally, all of the shorthand methods make use of jQuery's `on()` method. Example:

```
$('button').click(function())
```

Is the same than

```
$('button').on('click', function())
```

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. It is advisable to use the `on()` method because it gives more flexibility and executes slightly faster.

Once you have bound an event, you can unbind the event using the `off()` method. This will remove any event handlers that were bound to the specified event:

```
$('button').off( 'click');
```

One advantage that `on()` offers is the ability to bind and unbind specific events. For example:

```
$('li').on( 'click.logging', function() {
  console.log( 'a list item was clicked' );
});
$('li').on( 'click.analytics', function() {
  registerClick();
});
$('li').off( 'click.logging' );
```

This code leaves the `analytics`-related click untouched, while unbinding the `logging`-related click.

Another benefit of using `on()` is the ability to bind to multiple events at once. For example, you might want to run the same code when a user hover the mouse or when the mouse is clicked:

```
$( 'button').on('hover click', function() {
  console.log( 'The user hovered or clicked the mouse');
});
```

<u>Further Optional Reading</u>

Event bubbling: "bubbling up" means that an event applied to a child element will propagate to its parent, grandparent and up to the top of the HTML document. jQuery provides *event.stopPropagation()* to stop the bubbling up of events. Some DOM events do not propagate – focus, blur, load, and unload don't.

event.stopPropagation() basically tells jQuery: *only apply this event to this child node and don't tell the parent containers anything because I don't want them to react*.

Event Delegation: Event delegation is an event handling technique where, instead of attaching event handlers directly to every element you want to listen to events on, you attach a single event handler to a parent element of those elements to listen for events occurring on its descendant elements. When handling the event, you check which element fired the event, and respond accordingly. Event delegation relies on event bubbling in the DOM. Event delegation uses less memory because you replace multiple event handlers with a single event handler.

Manipulating content on the DOM

JQuery provides methods to manipulate DOM in efficient way making easier to modify the value of any element's attribute or to extract HTML code from a paragraph or division. What is important is to understand the concept used by jQuery to identify the elements of the DOM tree. Taking into account their position/ hierarchy in the tree, elements can be grand-parents, parents, children, descendants, ancestors and siblings.

jQuery provides both (1) methods to "traverse" — or move through — the HTML elements, and, (2) methods to alter the value or content of the selected elements.

DOM Manipulation Methods:

Following table lists down all the methods which you can use to manipulate DOM elements:

Method	Description
after(content)	Inserts content after the selected elements
append(content)	Inserts content at the end of the selected elements.
appendTo(selector)	Append all of the matched elements to another, specified, set of elements.
before(content)	Inserts content before the selected elements
clone(bool)	Clone matched DOM Elements, and all their event handlers, and select the clones.
clone( )	Clone matched DOM Elements and select the clones.
empty( )	Removes the child elements from the selected element
html(val)	Set the html contents of every matched element.
html( )	Get the html contents (innerHTML) of the first matched element.
insertAfter(selector)	Insert all of the matched elements after another, specified, set of elements.

insertBefore(selector)	Insert all of the matched elements before another, specified, set of elements.
prepend(content)	Inserts content at the beginning of the selected elements
prependTo(selector)	Prepend all of the matched elements to another, specified, set of elements.
remove(expr)	Removes the selected element (and its child elements)
replaceAll(selector)	Replaces the elements matched by the specified selector with the matched elements.
replaceWith(content)	Replaces all matched elements with the specified HTML or DOM elements.
text(val)	Set the text contents of all matched elements.
text( )	Get the combined text contents of all matched elements.
wrap(elem)	Wrap each matched element with the specified element.
wrap(html)	Wrap each matched element with the specified HTML content.
wrapAll(elem)	Wrap all the elements in the matched set into a single wrapper element.
wrapAll(html)	Wrap all the elements in the matched set into a single wrapper element.
wrapInner(elem)	Wrap the inner child contents of each matched element (including text nodes) with a DOM element.
wrapInner(html)	Wrap the inner child contents of each matched element (including text nodes) with an HTML structure.

Adding simple effects

jQuery makes it trivial to add simple effects to your page. Effects can use the built-in settings provided by jQuery:

- `.show()` Show the selected elements.
- `.hide()` Hide the selected elements.
- `.fadeIn()` Animate the opacity of the selected elements to 100%.
- `.fadeOut()` Animate the opacity of the selected elements to 0%.
- `.slideDown()` Display the selected elements with a vertical sliding motion.
- `.slideUp()` Hide the selected elements with a vertical sliding motion.
- `.slideToggle()` Show or hide the selected elements with a vertical sliding motion, depending on whether the elements are currently visible.

Please refer to the slides for this lecture for an example of using jQuery to manipulate the DOM by adding simple effects to the DOM and, creating and appending content.

jQuery UI

jQuery UI is a library of widgets, "interactions", utilities, themes and special effects built on top of jQuery. jQuery UI is mainly used to quickly build impressive, robust and highly interactive web applications. jQuery UI is the official jQuery user interface library. The current stable version of jQuery UI is 1.10.3 and includes the following components:

UI Core

- Core
- Widget
- Mouse
- Position

Interactions

- Draggable
- Droppable
- Resizable
- Selectable
- Insertable

Effects

- Effect core
- Blind effect
- Bounce effect
- Clip effect
- Drop effect
- Explode effect
- Fade effect
- Fold effect
- Highlight effect
- Pulsate effect
- Scale effect
- Shake effect
- Slide effect
- Transfer effect

Widgets

- Accordion
- Autocomplete
- Button
- Datepicker
- Dialog
- Menu
- Progressbar
- Slider
- Spinner
- Tabs
- Tooltip

24 Themes to choose

Study the following working example:

http://www.icbl.hw.ac.uk/santiago/teaching/F28WP/jqueryui.html

Further variants and discussion about this example can be found at

http://keith-wood.name/uiTabs.html

jQuery and the Coursework

You can use sortable tables for your CMS:
Create the web page that would list the fields of each of your Twitter accounts stored in your MySQL table. Use jQuery to make the list sortable by each of those fields. Don't forget to include a link that allows users to go back to the input web page.

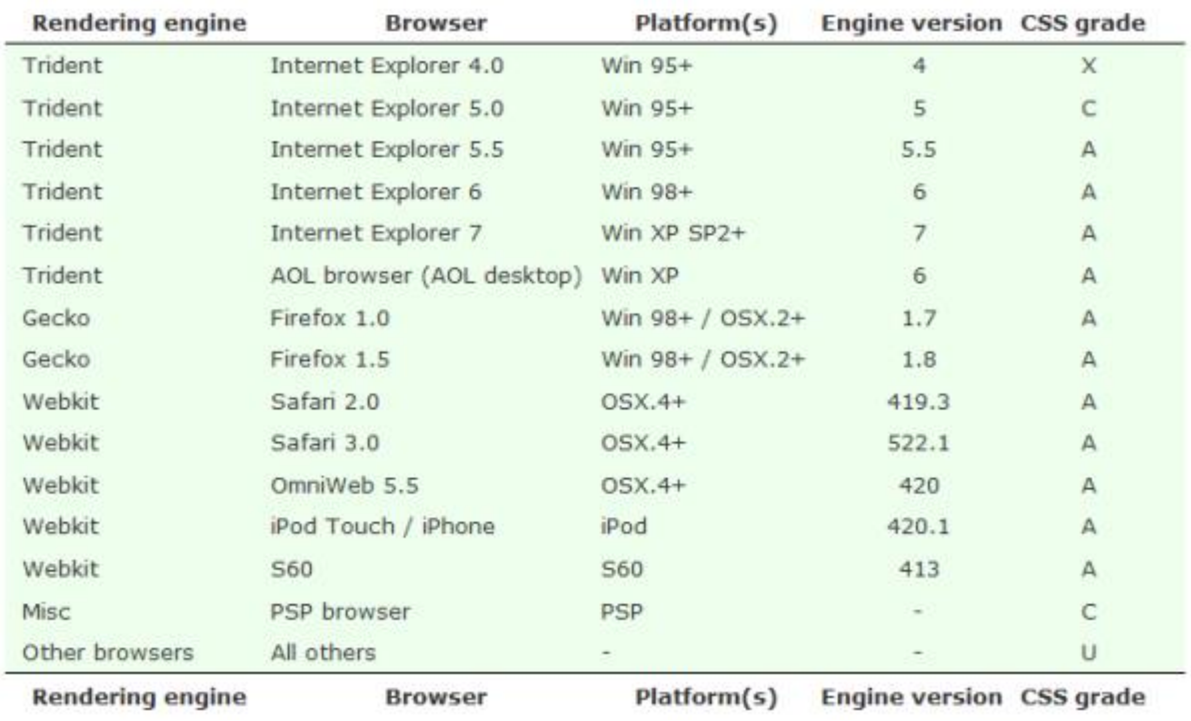
You can use jQuery or/and any of its available plug-ins and libraries to implement a sortable table for your CMS. You are not expected to create your own JavaScript code or develop your own jQuery plug-in. We advise you to use some of the following jQuery plug-ins:

- ✓ DataTables (http://datatables.net),
- ✓ Tablesorter (http://tablesorter.com/docs),
- ✓ TableRowCheckboxToggle (http://pure-essence.net/stuff/webTips/jqueryTableRowCheckboxToggle.html)

- ✓ Ingrid (http://reconstrukt.com/ingrid),
- ✓ Flexigrid (http://flexigrid.info) or,
- ✓ stupidTable (http://joequery.github.io/Stupid-Table-Plugin)

<u>Using the DataTables plug-in</u>

Jquery DataTables is an excellent Open Source jQuery plug-in that enables you to easily enhance your tables. It includes all the features that you have been asked to implement in the coursework. You can download DataTables from http://www. datatables.net/download/



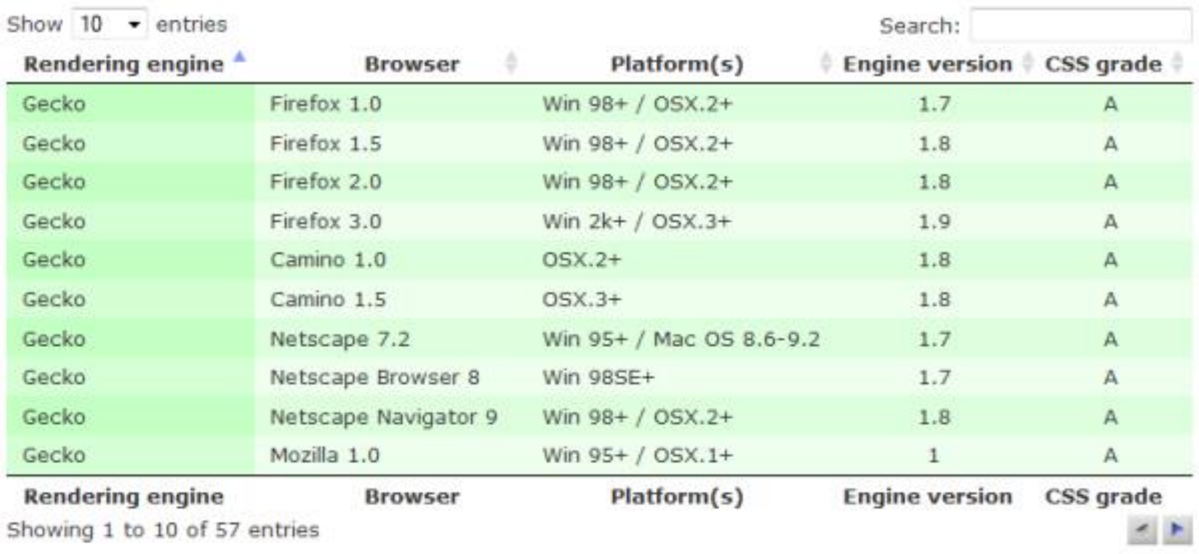
Rendering engine	Browser	Platform(s)	Engine version	CSS grade
Trident	Internet Explorer 4.0	Win 95+	4	X
Trident	Internet Explorer 5.0	Win 95+	5	C
Trident	Internet Explorer 5.5	Win 95+	5.5	A
Trident	Internet Explorer 6	Win 98+	6	A
Trident	Internet Explorer 7	Win XP SP2+	7	A
Trident	AOL browser (AOL desktop)	Win XP	6	A
Gecko	Firefox 1.0	Win 98+ / OSX.2+	1.7	A
Gecko	Firefox 1.5	Win 98+ / OSX.2+	1.8	A
Webkit	Safari 2.0	OSX.4+	419.3	A
Webkit	Safari 3.0	OSX.4+	522.1	A
Webkit	OmniWeb 5.5	OSX.4+	420	A
Webkit	iPod Touch / iPhone	iPod	420.1	A
Webkit	S60	S60	413	A
Misc	PSP browser	PSP	-	C
Other browsers	All others	-	-	U
Rendering engine	**Browser**	**Platform(s)**	**Engine version**	**CSS grade**

Let us assume that you have a plain HTML table in your web page as shown in the image above. The table is a standard valid HTML table. Can you image how much effort will you need to do your coursework with only jQuery?

With jQuery DataTables, the answer is one line of JavaScript code and a few lines of CSS. The required JavaScript code is shown below:

```
$("table#myTableId").dataTable();
```

DataTables will do the job for you. It will add the required features to this table like ordering by columns, searching (filtering) by keyword, pagination, and changing the number of items per page, as shown in the following image.


Show 10 entries

Search:

Rendering engine	Browser	Platform(s)	Engine version	CSS grade
Gecko	Firefox 1.0	Win 98+ / OSX.2+	1.7	A
Gecko	Firefox 1.5	Win 98+ / OSX.2+	1.8	A
Gecko	Firefox 2.0	Win 98+ / OSX.2+	1.8	A
Gecko	Firefox 3.0	Win 2k+ / OSX.3+	1.9	A
Gecko	Camino 1.0	OSX.2+	1.8	A
Gecko	Camino 1.5	OSX.3+	1.8	A
Gecko	Netscape 7.2	Win 95+ / Mac OS 8.6-9.2	1.7	A
Gecko	Netscape Browser 8	Win 98SE+	1.7	A
Gecko	Netscape Navigator 9	Win 98+ / OSX.2+	1.8	A
Gecko	Mozilla 1.0	Win 95+ / OSX.1+	1	A
Rendering engine	Browser	Platform(s)	Engine version	CSS grade

Showing 1 to 10 of 57 entries

You would need to identify your table with an ID (e.g. *#myTableId*), and make sure to use the *<thead>*, *<th>*. *<tbody>* and *<tfoot>* elements in your table. The rest is done by DataTables and your CSS files. DataTables will take the table identified by *#myTableId* and dynamically injects all the elements you see in the figure above and attaches event handlers to them that manage all the interactions with the user.

Configure the default UI features

DataTables can be configured by using its configuration parameters. For example the default functionalities can be easily disabled if some of them are not required. The following listing shows an example of configuration that can be used to disable some default features:

```
$('#myTableId').dataTable( {
    "bPaginate": false,  //To remove pagination
    "bLengthChange": false, //Don’t allow user to select # items per page
    "bFilter": false, // disable multiple words in the search box
    "bSort": false, // disable sorting of column
    "bInfo": false, // disable the table information display
    "bAutoWidth": false, // disable automatic column width calculation
    "asStripClasses": null //To remove "odd"/"event" zebra classes
} );
```

Configure default look&feel

You can also configure the default look of the enhanced table. For example:

1. You can use the LengthMenu array to change the number of items per page shown in the drop-down menu - by default, in the drop-down list are placed 10, 25, 50, and 100 items.
2. You can use the iDisplayLength parameter (default 10) to specify how much rows will be initially displayed per page.
3. You can define text that will be shown when there are no records in the table, using the *sZeroRecords* parameter in the *oLanguage* property. Other labels that can be configured are found at http://datatables.net/extras/tabletools/initialisation and similar.

4. State saving - state of the table (current page, filter condition) can be stored in the browser cookie so it can be restored when the user comes back to the page. You can also set the cookie duration in the iCookieDuration parameter.
5. *bScrollInfinite* (default *false*) disables pagination and loads the next page when the user scrolls down. *sScrollX* and *sScrollY* parameters set the dimensions of the horizontal and vertical scroll bars.
6. You can change the look and the text of the search text box with the *oSearch* parameter. In the property of the object, you also can set whether Regular Expression or smart searching (*bFilter*) should be used.

For the purpose of the coursework you can use the default DataTables configuration. However, if you would like further information on DataTables, you can read the documentation available on the DataTables website and check the relevant tutorials listed in the references at the end of this notes. In addition, you can use the following code within the <*head*> element to include a working version of DataTables in your HTML:

```
<style type="text/css" title="currentStyle">
  @import
"http://www.icbl.hw.ac.uk/santiago/teaching/F28WP/CW/datatables/demo_page.c
ss";
  @import
"http://www.icbl.hw.ac.uk/santiago/teaching/F28WP/CW/datatables/jquery.data
Tables.css";
</style>
<script type="text/javascript"
src="http://www.icbl.hw.ac.uk/santiago/teaching/F28WP/CW/datatables/jquery-
1.4.3.min.js"></script>
<script type="text/javascript"
src="http://www.icbl.hw.ac.uk/santiago/teaching/F28WP/CW/datatables/jquery.
dataTables.js"></script>
<style type="text/css">
.TPtable_w_cnd01fmd {width:260px;height:125px;border:0;}
.TPheader_w_cnd01fmd {font-family:arial,helvetica;font-size:normal;font-
weight:bold;padding-bottom:5px;}
.TPcell_w_cnd01fmd {font-family:arial,helvetica;font-size:smaller;font-
weight:normal;}
a.TPcell_w_cnd01fmd {font-family:arial,helvetica;font-
size:smaller;color:#0000ff;font-weight:bold;}
a.TParrow_w_cnd01fmd {font-family:arial,helvetica;font-
size:smaller;color:#0000ff;font-weight:bold;text-decoration:none;}
</style>
```

Further Reading

jQuery: Novice to Ninja: New Kicks And Tricks by Earle Castledine and Craig Sharkie
jQuery in Action by Bear Bibeault
http://api.jquery.com/category/events/
http://jqfundamentals.com/chapter/events
http://fuelyourcoding.com/jquery-events-stop-misusing-return-false/
http://api.jquery.com/category/manipulation/
http://api.jquery.com/category/effects/
http://jqueryui.com/
http://www.codeproject.com/Articles/194916/Enhancing-HTML-tables-using-a-JQuery-DataTables-pl
http://datatables.net/usage/features
http://datatables.net/usage/i18n
http://fuelyourcoding.com/jquery-events-stop-misusing-return-false/