JQUERY

INDEX

Best coding practices

Software quality

As listed below, there are many attributes associated with good software. Some of these can be mutually contradictory (e.g. very fast versus full error checking), and different customers and participants may have different priorities. Weinberg provides an example of how different goals can have a dramatic effect on both effort required and efficiency.Furthermore, he notes that programmers will generally aim to achieve any explicit goals which may be set, probably at the expense of any other quality attributes.

Sommerville has identified four generalised attributes which are not concerned with what a program does, but how well the program does it.

- Maintainability
- Dependability
- Efficiency
- Usability

Weinberg has identified four targets which a good program should meet.

- Does a program meet its specification; "correct output for each possible input"?
- Is the program produced on schedule (and within budget)?
- How adaptable is the program to cope with changing requirements?
- Is the program efficient enough for the environment in which it is used?

Hoare has identified seventeen objectives related to software quality, including.

- Clear definition of purpose.
- Simplicity of use.
- Ruggedness (difficult to misuse, kind to errors).
- Early availability (delivered on time when needed).
- Reliability.
- Extensibility in the light of experience.
- Brevity.
- Efficiency (fast enough for the purpose to which it is put).
- Minimum cost to develop.
- Conformity to any relevant standards.
- Clear, accurate, and precise user documents.

Prerequisites

Before coding starts, it is important to ensure that all necessary prerequisites have been completed (or have at least progressed far enough to provide a solid foundation for coding). If the various prerequisites are not satisfied then the software is likely to be unsatisfactory, even if it is completed.

From Meek & Heath: "What happens before one gets to the coding stage is often of crucial importance to the success of the project.

The prerequisites outlined below cover such matters as:

- how is development structured? (life cycle)
- what is the software meant to do? (requirements)
- the overall structure of the software system (architecture)
- more detailed design of individual components (design)
- choice of programming language(s)

For small simple projects involving only one person, it may be feasible to combine architecture with design and adopt a very simple life cycle.

Life cycle

A software development methodology is a framework that is used to structure, plan, and control the life cycle of a software product. Common methodologies include waterfall, prototyping, iterative and incremental development, spiral development, agile software development, rapid application development, and extreme programming.

The waterfall model is a sequential development approach; in particular, it assumes that the requirements can be completely defined at the start of a project. However, McConnell quotes three studies which indicate that, on average, requirements change by around 25% during a project. The other methodologies mentioned above all attempt to reduce the impact of such requirement changes, often by some form of step-wise, incremental, or iterative approach. Different methodologies may be appropriate for different development environments.

Requirements

McConnell states: "The first prerequisite you need to fulfill before beginning construction is a clear statement of the problem the system is supposed to solve.

Meek and Heath emphasise that a clear, complete, precise, and unambiguous written specification is the target to aim for. Note that it may not be possible to achieve this target, and the target is likely to change anyway (as mentioned in the previous section).

Sommerville distinguishes between less detailed user requirements and more detailed system requirements. He also distinguishes between functional requirements (e.g. update a record) and non-functional requirements (e.g. response time must be less than 1 second).

Architecture

Hoare points out: "there are two ways of constructing a software design: one way is to make it so simple that there are *obviously* no deficiencies; the other way is to make it so complicated that there are no *obvious* deficiencies. The first method is far more difficult.

Software architecture is concerned with deciding what has to be done, and which program component is going to do it (how something is done is left to the detailed design phase, below). This is particularly important when a software system contains more than one program since it effectively defines the interface between these various programs. It should include some consideration of any user interfaces as well, without going into excessive detail.

Any non-functional system requirements (response time, reliability, maintainability, etc.) need to be considered at this stage.

The software architecture is also of interest to various stakeholders (sponsors, end-users, etc.) since it gives them a chance to check that their requirements can be met.

Design

The main purpose of design is to fill in the details which have been glossed over in the architectural design. The intention is that the design should be detailed enough to provide a good guide for actual coding, including details of any particular algorithms to be used. For example, at the architectural level, it may have been noted that some data has to be sorted, while at the design level it is necessary to decide which sorting algorithm is to be used. As a further example, if an object-oriented approach is being used, then the details of the objects must be determined (attributes and methods).

Choice of programming language(s)

Mayer states: "No programming language is perfect. There is not even a single best language; there are only languages well suited or perhaps poorly suited for particular purposes. Understanding the problem and associated programming requirements is necessary for choosing the language best suited for the solution.

From Meek & Heath: "The essence of the art of choosing a language is to start with the problem, decide what its requirements are, and their relative importance since it will probably be impossible to satisfy them all equally well. The available languages should then be measured against the list of requirements, and the most suitable (or least unsatisfactory) chosen."

It is possible that different programming languages may be appropriate for different aspects of the problem. If the languages or their compilers permit, it may be feasible to mix routines written in different languages within the same program.

Even if there is no choice as to which programming language is to be used, McConnell provides some advice: "Every programming language has strengths and weaknesses. Be aware of the specific strengths and weaknesses of the language you're using.

Coding standards

This section is also really a prerequisite to coding, as McConnell points out: "Establish programming conventions before you begin programming. It's nearly impossible to change code to match them later.

As listed near the end of Coding conventions, there are different conventions for different programming languages, so it may be counterproductive to apply the same conventions across different languages.

The use of coding conventions is particularly important when a project involves more than one programmer (there have been projects with thousands of programmers). It is much easier for a programmer to read code written by someone else if all code follows the same conventions.

For some examples of bad coding conventions, Roedy Green provides a lengthy (tongue-in-cheek) article on how to produce unmaintainable code.

Commenting

Due to time restrictions or enthusiastic programmers who want immediate results for their code, commenting of code often takes a back seat. Programmers working as a team have found it better to leave comments behind since coding usually follows cycles, or more than one person may work on a particular module. However, some commenting can decrease the cost of knowledge transfer between developers working on the same module.

In the early days of computing, one commenting practice was to leave a brief description of the following:

1. Name of the module

2. Purpose of the Module
3. Description of the Module
4. Original Author
5. Modifications
6. Authors who modified code with a description on why it was modified.

The "description of the module" should be as brief as possible but without sacrificing clarity and comprehensiveness.

However, the last two items have largely been obsoleted by the advent of revision control systems. Modifications and their authorship can be reliably tracked by using such tools rather than by using comments.

Also, if complicated logic is being used, it is a good practice to leave a comment "block" near that part so that another programmer can understand what is exactly happening.

Unit testing can be another way to show how code is intended to be used.

Naming conventions

Use of proper naming conventions is considered good practice. Sometimes programmers tend to use X1, Y1, etc. as variables and forget to replace them with meaningful ones, causing confusion.

In order to prevent this waste of time, it is usually considered good practice to use descriptive names in the code since it's about real data.

Example: A variable for taking in weight as a parameter for a truck can be named TrkWeight or TruckWeightKilograms, with TruckWeightKilograms being the more preferable one, since it is instantly recognisable.

Keep the code simple

The code that a programmer writes should be simple. Complicated logic for achieving a simple thing should be kept to a minimum since the code might be modified by another programmer in the future. The logic one programmer implemented may not make perfect sense to another. So, always keep the code as simple as possible.

For example, consider these equivalent lines of C code:

```
if (hours < 24 && minutes < 60 && seconds < 60)
{
    return true;
}
else
{
    return false;
}
```

and

```
if (hours < 24 && minutes < 60 && seconds < 60)
   return true;
else
    return false;
```

And

```
return hours < 24 && minutes < 60 && seconds < 60;
```

The 1st approach, which is much more commonly used, is considerably larger than the 3rd. In particular, it consumes 5 times more screen vertical space (lines), and 97 characters versus 52 (though editing tools may reduce the difference in actual typing). It is arguable, however, which is "simpler". The first has an explicit if/then else, with an explicit return value obviously connected with each; even a novice programmer should have no difficulty understanding it. The 2nd merely discards the braces, cutting the "vertical" size in half with little change in conceptual complexity. In most languages the "return" statements could also be appended to the prior lines, bringing the "vertical" size to only one more line that the 3rd form.

The third form obviously minimizes the size, but may increase the complexity: It leaves the "true" and "false" values implicit, and intermixes the notions of "condition" and "return value". It is likely obvious to most programmers, but a novice might not immediately understand that the result of evaluating a condition is actually a value (of type Boolean, or its equivalent in whatever language), and thus can be manipulated or returned. In more realistic examples, the 3rd form could have problems due to operator precedence, perhaps returning an unexpected type, where the prior forms would in some languages report an error. Thus, "simplicity" is not merely a matter of length, but of logical and conceptual structure; making code shorter may make it less or more complex.

For large, long lived programs using verbose alternatives could contribute to bloat.

Compactness can allow coders to view more code per page, reducing scrolling gestures and keystrokes. Given how many times code might be viewed in the process of writing and maintaining, it might amount to a significant savings in programmer keystrokes in the life of the code. This might not seem significant to a student first learning to program. But when producing and maintaining large programs which often reach thousands or even millions of lines, it becomes apparent how much a minor code simplification might speed work, and lessen finger, wrist and eye strain, which are common medical issues suffered by production coders and information workers.

Terser coding speeds compilation very slightly, as fewer symbols need to be processed. Furthermore, the 3rd approach may allow similar lines of code to be more easily compared, particularly when many such constructs can appear on one screen at the same time.

Finally, very terses layouts might better utilize modern wide-screen computer displays. In the past screens were limited to 40 or 80 characters (such limits originated far earlier: manuscripts, printed books, and even scrolls, have for millennia used quite short lines. Modern screens can easily display 200 or more characters, allowing extremely long lines. Most modern coding styles and standards do not take up that entire width. Thus, if using one window as wide as the screen, a great deal of available space is wasted. On the other hand, with multiple windows, or using an IDE or other tool with various information in side panes, the available width for code is in the range familiar from earlier systems.

It is also worth noting that the human visual system is greatly affected by line length; very long lines slightly increase reading speed, but reduce comprehension and add to eye-tracking errors. Some studies suggest that longer lines fare better online than in print, but this still only goes up to about 10 inches, and mainly for raw speed of reading prose.

Portability

Program code should not contain "hard-coded" (literal) values referring to environmental parameters, such as absolute file paths, file names, user names, host names, IP addresses, URLs, UDP/TCP ports. Otherwise the application will not run on a host that has a different design than anticipated. A careful programmer can parametrize such variables and configure them for the hosting environment outside of the application proper (for example in property files, on an application server, or even in a database). Compare the mantra of a "single point of definition.

As an extension, resources such as XML files should also contain variables rather than literal values, otherwise the application will not be portable to another environment without editing the XML files. For example, with J2EE applications running in an application server, such environmental parameters can be defined in the scope of the JVM and the application should get the values from there.

Construction guidelines in brief

A general overview of all of the above:

1. Know what the code block must perform
2. Maintain naming conventions which are uniform throughout.
3. Indicate a brief description of what a variable is for (reference to commenting)
4. Correct errors as they occur.
5. Keep your code simple

Code development

Code building

A best practice for building code involves daily builds and testing, or better still continuous integration, or even continuous delivery.

Testing

Testing is an integral part of software development that needs to be planned. It is also important that testing is done proactively; meaning that test cases are planned before coding starts, and test cases are developed while the application is being designed and coded.

Debugging the code and correcting errors

Programmers tend to write the complete code and then begin debugging and checking for errors. Though this approach can save time in smaller projects, bigger and complex ones tend to have too many variables and functions that need attention. Therefore, it is good to debug every module once you are done and not the entire program. This saves time in the long run so that one does not end up wasting a lot of time on figuring out what is wrong. Unit tests for individual modules, and/or functional tests for web services and web applications, can help with this.

Deployment

Deployment is the final stage of releasing an application for users. Some best practices are.

1. Keep the installation structure simple: Files and directories should be kept to a minimum. Don’t install anything that’s never going to be used.

2. Keep only what is needed: The software configuration management activities must make sure this is enforced. Unused resources (old or failed versions of files, source code, interfaces, etc.) must be archived somewhere else to keep newer builds lean.
3. Keep everything updated: The software configuration management activities must make sure this is enforced. For delta-based deployments, make sure the versions of the resources that are already deployed are the latest before deploying the deltas. If not sure, perform a deployment from scratch (delete everything first and then re-deploy).
4. Adopt a multi-stage strategy: Depending on the size of the project, sometimes more deployments are needed.
5. Have a roll back strategy: There must be a way to roll-back to a previous (working) version.
6. Rely on automation for repeatable processes: There's far too much room for human error, deployments should not be manual. Use a tool that is native to each operating system or, use a scripting language for cross-platform deployments.
7. Re-create the real deployment environment: Consider everything (routers, firewalls, web servers, web browsers, file systems, etc.)
8. Do not change deployment procedures and scripts on-the-fly and, document such changes: Wait for a new iteration and record such changes appropriately.
9. Customize deployment: Newer software products such as APIs, micro-services, etc. require specific considerations for successful deployment.
10. Reduce risk from other development phases: If other activities such as testing and configuration management are wrong, deployment surely will fail.
11. Consider the influence each stakeholder has: Organizational, social, governmental considerations.

jQuery

jQuery is a JavaScript Library.
jQuery greatly simplifies JavaScript programming.
jQuery is easy to learn.

jQuery Introduction

The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery is a framework built using JavaScript capabilities. So, you can use all the functions and other capabilities available in JavaScript.

What is jQuery?

jQuery is a fast and concise JavaScript Library created by John Resig in 2006 with a nice motto: **Write less, do more**. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by jQuery:

· **HTML / DOM manipulation:** The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called **Sizzle**.

· **Event handling:** The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.

· **AJAX Support:** The jQuery helps you a lot to develop a responsive and feature-rich site using AJAX technology.

· **Effects and Animations:** The jQuery comes with plenty of built-in animation effects which you can use in your websites.

· **Lightweight:** The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).

· **Cross Browser Support:** The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+

Latest Technology: The jQuery supports CSS3 selectors and basic XPath syntax.

Why jQuery?

There are lots of other JavaScript frameworks out there, but jQuery seems to be the most popular, and also the most extendable.
Many of the biggest companies on the Web use jQuery, such as:

- Google
- Microsoft
- IBM

- Netflix

Will jQuery work in all browsers?

The jQuery team knows all about cross-browser issues, and they have written this knowledge into the jQuery library. jQuery will run exactly the same in all major browsers, including Internet Explorer 6!

jQuery Get Started

Adding jQuery to Your Web Pages

There are several ways to start using jQuery on your web site. You can:

- Download the jQuery library from jQuery.com
- Include jQuery from a CDN(content delivery network), like Google

Downloading jQuery

There are two versions of jQuery available for downloading:

- Production version - this is for your live website because it has been minified and compressed
- Development version - this is for testing and development (uncompressed and readable code)

Both versions can be downloaded from *jQuery.com.*
The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag (notice that the <script> tag should be inside the <head> section):

```
<head>
<script src="jquery-1.12.4.min.js"></script>
</head>
```

Tip: Place the downloaded file in the same directory as the pages where you wish to use it.
Do you wonder why we do not have type="text/javascript" inside the <script> tag?

This is not required in HTML5. JavaScript is the default scripting language in HTML5 and in all modern browsers!

jQuery CDN

If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).
Both Google and Microsoft host jQuery.
To use jQuery from Google or Microsoft, use one of the following:

Google CDN:

```
<head>
<scriptsrc="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
</head>
```

Microsoft CDN:

```
<head>
```

```
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.12.4.min.js"></script>
</head>
```

One big advantage of using the hosted jQuery from Google or Microsoft:

Many users already have downloaded jQuery from Google or Microsoft when visiting another site. As a result, it will be loaded from cache when they visit your site, which leads to faster loading time. Also, most CDN's will make sure that once a user requests a file from it, it will be served from the server closest to them, which also leads to faster loading time.

jQuery Syntax

The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).
Basic syntax is: **$(*selector*).*action*()**

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

Examples:
$(this).hide() - hides the current element.
$("p").hide() - hides all <p> elements.
$(".test").hide() - hides all elements with class="test".
$("#test").hide() - hides the element with id="test".

The Document Ready Event

You might have noticed that all jQuery methods in our examples, are inside a document ready event:

```
$(document).ready(function(){

// jQuery methods go here...

});
```

This is to prevent any jQuery code from running before the document is finished loading (is ready).
Before you can safely use jQuery to do anything to your page, you need to ensure that the page is in a state where it's ready to be manipulated. With jQuery, we accomplish this by putting our code in a function, and then passing that function to $(document).ready(). As you can see here, the function we pass can just be an anonymous function.

It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.
Here are some examples of actions that can fail if methods are run before the document is fully loaded:

- Trying to hide an element that is not created yet
- Trying to get the size of an image that is not loaded yet

Tip: The jQuery team has also created an even shorter method for the document ready event:

```
$(function(){
// jQuery methods go here...


});
```

Use the syntax you prefer. We think that the document ready event is easier to understand when reading the code.

jQuery Selectors

jQuery selectors are one of the most important parts of the jQuery library.
jQuery selectors allow you to select and manipulate HTML element(s).
jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.
All selectors in jQuery start with the dollar sign and parentheses: $().

The element Selector

The jQuery element selector selects elements based on the element name.
You can select all <p> elements on a page like this:

```
$("p")
```

Example :
When a user clicks on a button, all <p> elements will be hidden:

```
$(document).ready(function(){
        $("button").click(function(){
    $("p").hide();
        });
});
```

The#id Selector

The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.
An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.
To find an element with a specific id, write a hash character, followed by the id of the HTML element:

```
$("#test")
```

Example :
When a user clicks on a button, the element with id="test" will be hidden:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("button").click(function(){
```

```
        $("#test").hide();
        });
});
</script>
</head>
<body>

<h2>This is a heading</h2>

<p>This is a paragraph.</p>
<p id="test">This is another paragraph.</p>

<button>Click me</button>

</body>
</html>
```

The .class Selector

The jQuery class selector finds elements with a specific class.
To find elements with a specific class, write a period character, followed by the name of the class:
$(".test")

Example :
When a user clicks on a button, the elements with class="test" will be hidden:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js">
</script>
<script>
$(document).ready(function(){
        $("button").click(function(){
        $(".test").hide();
        });
});
</script>
</head>
<body>

<h2 class="test">This is a heading</h2>

<p class="test">This is a paragraph.</p>
<p>This is another paragraph.</p>

<button>Click me</button>

</body>
</html>
```

More Examples of jQuery Selectors

Syntax	Description	Example
*	$("*")	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>, 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 element
element + next	$("div + p")	The <p> element that are next to each elements
element ~ siblings	$("div ~ p")	All <p> elements that are siblings of a 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
: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 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
[attribute]	$("[href]")	All elements with a href attribute
[attribute=value]	$("[href='default.html']")	All elements with a href attribute value equal to "default.html"
[attribute!=value]	$("[href!='default.html']")	All elements with a href attribute value not equal to "default.html"
[attribute$=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 specific word "hello"
[attribute*=value]	$("[title*='hello']")	All elements with a title attribute value containing the word "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

jQuery Event Methods

jQuery is tailor-made to respond to events in an HTML page.

What are Events?

All the different visitor's actions that a web page can respond to are called events.
An event represents the precise moment when something happens.
Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element

· The term **"fires/fired"** is often used with events. Example: "The keypress event is fired, the moment you press a key".

· Here are some common DOM events:

Mouse Events	**Keyboard Events**	**Form Events**	**Document/Window Events**
click dblclick mouseenter mouseleave	keypress keydown keyup	submit change focus blur	load resize scroll unload

jQuery Syntax For Event Methods

In jQuery, most DOM events have an equivalent jQuery method.
To assign a click event to all paragraphs on a page, you can do this:

```
$("p").click();
```

The next step is to define what should happen when the event fires. You must pass a function to the event:

```
$("p").click(function(){
  // action goes here!!
});
```

Commonly Used jQuery Event Methods

$(document).ready()

The $(document).ready() method allows us to execute a function when the document is fully loaded.

click()

The click() method attaches an event handler function to an HTML element.
The function is executed when the user clicks on the HTML element.
The following example says: When a click event fires on a <p> element; hide the current <p> element:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("p").click(function(){
        $(this).hide();
        });
```

```
});
</script>
</head>
<body>

<p>If you click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>

</body>
</html>
```

dblclick()

The dblclick() method attaches an event handler function to an HTML element.
The function is executed when the user double-clicks on the HTML element:

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("p").dblclick(function(){
        $(this).hide();
        });
});
</script>
</head>
<body>

<p>If you double-click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>

</body>
</html>
```

mouseenter()

The mouseenter() method attaches an event handler function to an HTML element.
The function is executed when the mouse pointer enters the HTML element:

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
```

```
        $("#p1").mouseenter(function(){
alert("You entered p1!");
        });
});
</script>
</head>
<body>

<p id="p1">Enter this paragraph.</p>

</body>
</html>
```

mouseleave()

The mouseleave() method attaches an event handler function to an HTML element.
The function is executed when the mouse pointer leaves the HTML element:

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("#p1").mouseleave(function(){
alert("Bye! You now leave p1!");
        });
});
</script>
</head>
<body>

<p id="p1">This is a paragraph.</p>

</body>
</html>
```

hover()

The hover() method takes two functions and is a combination of the mouseenter() and mouseleave() methods.
The first function is executed when the mouse enters the HTML element, and the second function is executed when the mouse leaves the HTML element:

Example :

```
<!DOCTYPE html>
<html>
<head>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("#p1").hover(function(){
alert("You entered p1!");
        },
function(){
alert("Bye! You now leave p1!");
        });
});
</script>
</head>
<body>

<p id="p1">This is a paragraph.</p>

</body>
</html>
```

mousedown()

The mousedown() method attaches an event handler function to an HTML element.
The function is executed, when the left, middle or right mouse button is pressed down, while the mouse is over the HTML element:

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("#p1").mousedown(function(){
alert("Mouse down over p1!");
        });
});
</script>
</head>
<body>

<p id="p1">This is a paragraph.</p>

</body>
</html>
```

mouseup()

The mouseup() method attaches an event handler function to an HTML element.
The function is executed, when the left, middle or right mouse button is released, while the mouse is over the HTML element:

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("#p1").mouseup(function(){
alert("Mouse up over p1!");
        });
});
</script>
</head>
<body>

<p id="p1">This is a paragraph.</p>

</body>

</html>
```

focus() & blur()

The focus() & blur() methods attaches an event handler function to an HTML form field.
The focus() function is executed when the form field gets focus.
The blur() function is executed when the form field loses focus.

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("input").focus(function(){
    $(this).css("background-color", "#cccccc");
        });
        $("input").blur(function(){
    $(this).css("background-color", "#ffffff");
        });
});
</script>
</head>
<body>

Name: <input type="text" name="fullname"><br>
Email: <input type="text" name="email">

</body>
</html>
```

The on() Method

The on() method attaches one or more event handlers for the selected elements.

Example 1 :

Attach a click event to a <p> element:

```
$("p").on("click", function(){
        $(this).hide();
});
```

Example 2 :

Attach multiple event handlers to a <p> element:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("p").on({
mouseenter: function(){
        $(this).css("background-color", "lightgray");
        },
mouseleave: function(){
        $(this).css("background-color", "lightblue");
        },
click: function(){
        $(this).css("background-color", "yellow");
        }
        });
});
</script>
</head>
<body>

<p>Click or move the mouse pointer over this paragraph.</p>

</body>
</html>
```

jQuery Effects

The following table lists all the jQuery methods for creating animation effects.

Method	Description
hide()	Hides the selected elements
show()	Shows the selected elements
toggle()	Toggles between the hide() and show() methods
animate()	Runs a custom animation on the selected elements
delay()	Sets a delay for all queued functions on the selected elements
fadeIn()	Fades in the selected elements
fadeOut()	Fades out the selected elements
fadeTo()	Fades in/out the selected elements to a given opacity
fadeToggle()	Toggles between the fadeIn() and fadeOut() methods

slideDown()	Slides-down (shows) the selected elements
slideUp()	Slides-up (hides) the selected elements
slideToggle()	Toggles between the slideUp() and slideDown() methods

jQuery hide() and show()

With jQuery, you can hide and show HTML elements with the hide() and show() methods:

Example 1 :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("#hide").click(function(){
        $("p").hide();
        });
        $("#show").click(function(){
        $("p").show();
        });
});
</script>
</head>
<body>

<p>If you click on the "Hide" button, I will disappear.</p>

<button id="hide">Hide</button>
<button id="show">Show</button>

</body>
</html>
```

Syntax:

$(*selector*).hide(*speed,callback*);

$(*selector*).show(*speed,callback*);

The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds.
The optional callback parameter is a function to be executed after the hide() or show() method completes (you will learn more about callback functions in a later chapter).
The following example demonstrates the speed parameter with hide():

Example 2 :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("button").click(function(){
        $("p").hide(1000);
        });
});
</script>
</head>
<body>

<button>Hide</button>

<p>This is a paragraph with little content.</p>
<p>This is another small paragraph.</p>

</body>
</html>
```

toggle()

With jQuery, you can toggle between the hide() and show() methods with the toggle() method.
Shown elements are hidden and hidden elements are shown.

Syntax:

$(*selector*).toggle(*speed,easing,callback*);

The optional easing parameter Specifies the speed of the element in different points of the animation.
Default value is "swing"

- "swing" - moves slower at the beginning/end, but faster in the middle
- "linear" - moves in a constant speed

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
```

```
$(document).ready(function(){
        $("button").click(function(){
        $("p").toggle();
        });
});
</script>
</head>
<body>

<button>Toggle between hiding and showing the paragraphs</button>

<p>This is a paragraph with little content.</p>
<p>This is another small paragraph.</p>

</body>

</html>
```

jQuery Fading Methods

With jQuery you can fade an element in and out of visibility.
jQuery has the following fade methods:

- fadeIn()
- fadeOut()
- fadeToggle()
- fadeTo()

fadeIn() Method

The jQuery fadeIn() method is used to fade in a hidden element.

Syntax:

$(*selector*).fadeIn(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
The optional callback parameter is a function to be executed after the fading completes.

Example :
The following example demonstrates the fadeIn() method with different parameters:

```
$("button").click(function(){
   $("#div1").fadeIn();
   $("#div2").fadeIn("slow");
   $("#div3").fadeIn(3000);
});
```

fadeOut() Method

The jQuery fadeOut() method is used to fade out a visible element.

Syntax:

$(*selector*).fadeOut(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
The optional callback parameter is a function to be executed after the fading completes.

Example :

The following example demonstrates the fadeout() method with different parameters:

```
$("button").click(function(){
   $("#div1").fadeOut();
         $("#div2").fadeOut("slow");
   $("#div3").fadeOut(3000);
});
```

fadeToggle() Method

The jQuery fadeToggle() method toggles between the fadeIn() and fadeOut() methods.
If the elements are faded out, fadeToggle() will fade them in.
If the elements are faded in, fadeToggle() will fade them out.

Syntax:

$(*selector*).fadeToggle(*speed,callback*);
The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
The optional callback parameter is a function to be executed after the fading completes.

Example :

The following example demonstrates the fadeToggle() method with different parameters:

```
$("button").click(function(){
   $("#div1").fadeToggle();
   $("#div2").fadeToggle("slow");
         $("#div3").fadeToggle(3000);
});
```

fadeTo() Method

The jQuery fadeTo() method allows fading to a given opacity (value between 0 and 1).

Syntax:

$(*selector*).fadeTo(*speed,opacity,callback*);

The required speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
The required opacity parameter in the fadeTo() method specifies fading to a given opacity (value between 0 and 1).
The optional callback parameter is a function to be executed after the function completes.
The following example demonstrates the fadeTo() method with different parameters:

Example :

```
$("button").click(function(){
        $("#div1").fadeTo("slow", 0.15);
        $("#div2").fadeTo("slow", 0.4);
        $("#div3").fadeTo("slow", 0.7);
});
```

jQuery Sliding Methods

With jQuery you can create a sliding effect on elements.
jQuery has the following slide methods:

- slideDown()
- slideUp()
- slideToggle()

SlideDown() Method

The jQuery slideDown() method is used to slide down an element.

Syntax:

$(*selector*).slideDown(*speed,callback*);
The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
The optional callback parameter is a function to be executed after the sliding completes.
The following example demonstrates the slideDown() method:

Example :

```
$("#flip").click(function(){
        $("#panel").slideDown();
});
```

SlideUp() Method

The jQuery slideUp() method is used to slide up an element.

Syntax:

$(*selector*).slideUp(*speed,callback*);

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
The optional callback parameter is a function to be executed after the sliding completes.
The following example demonstrates the slideUp() method:

Example :

```
$("#flip").click(function(){
```

```
        $("#panel").slideUp();
});
```

SlideToggle() Method

The jQuery slideToggle() method toggles between the slideDown() and slideUp() methods.
If the elements have been slid down, slideToggle() will slide them up.
If the elements have been slid up, slideToggle() will slide them down.

Syntax:

$(*selector*).slideToggle(*speed,callback*);

The optional speed parameter can take the following values: "slow", "fast", milliseconds.
The optional callback parameter is a function to be executed after the sliding completes.
The following example demonstrates the slideToggle() method:

Example :

```
$("#flip").click(function(){
        $("#panel").slideToggle();
});
```

animate() Method

The jQuery animate() method is used to create custom animations.

Syntax:

$(*selector*).animate({*params*},*speed,callback*);

The required params parameter defines the CSS properties to be animated.
The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
The optional callback parameter is a function to be executed after the animation completes.

The following example demonstrates a simple use of the animate() method; it moves a element to the right, until it has reached a left property of 250px:

Example :

```
$("button").click(function(){
        $("div").animate({left: '250px'});
});
```

Note: By default, all HTML elements have a static position, and cannot be moved.
To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!

Self Evaluation 1

1. Hide all headings on a page when they are clicked.

2. Count the number of specific elements.

3. Scroll to the top of the page with jQuery.

4. How to detect whether the user has pressed 'Enter key' using jQuery.

5. Blink text using jQuery.

Solutions

1. HTML Code:

```
<!DOCTYPE html>
<html>
<head>
<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>
        <title>Hide all headings  on a page when they are clicked</title>
</head>
<body>
<h1>Heading-1</h1>
<h2>Heading-2</h2>
<h3>Heading-3</h3>
</body>
</html>
```

JavaScript Code:

```
$("h1,h2,h3" ).click(function() {
  $( this ).slideUp();
});
```

2. HTML Code:

```
<!DOCTYPE html>
<html>
<head>
<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>

<title>Count the number of specific elements</title>
</head>
<body>
<ul>
<li>List - 1</li>
<li>List - 2</li>
<li>List - 3</li>
</ul>
<button>Display the number of li elements in console</button>
</body>
</html>
```

JavaScript Code:

```
$("button").click(function(){
        console.log($("li").length);
        })
```

3. HTML Code:

```
<!DOCTYPE html>
<html>
```

```
<head>
<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>
<title>Scroll to the top of the page with jQuery</title>
</head>
<body>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<p>jquery</p>
<a href='#top'>Go Top</a>
</body>
</html>
```

JavaScript Code:

```
$("a[href='#top']").click(function() {
  $("html, body").animate({ scrollTop: 0 }, "slow");
return false;
});
```

4. HTML Code:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-git.js"></script>
<title>Detect pressing Enter key on keyboard using jQuery.</title>
</head>
<body>
<p>Input a value within the textbox and press Enter</p>
<form>
<input type="text" id="hello"></p>
</form>
</body>
</html>
```

JavaScript Code:

```
$(document).keypress(function(e) {
        if(e.which == 13) {
        console.log('You pressed enter!');
        }
});
```

5. HTML Code:

```
<!DOCTYPE html>

<html>

<head>


<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>

<title>Blink text using jQuery</title>

</head>

<body>

<p>jQuery <span class="blink">Exercises</span> and Solution</p>

</body>

</html>
```

JavaScript Code:

```
function blink_text() {
        $('.blink').fadeOut(500);

        $('.blink').fadeIn(500);

}

setInterval(blink_text, 1000);
```

jQuery Callback Functions

A callback function is executed after the current effect is 100% finished.
JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors.
To prevent this, you can create a callback function.
A callback function is executed after the current effect is finished.

Syntax:

```
$(selector).hide(speed,callback);
```

Examples
The example below has a callback parameter that is a function that will be executed after the hide effect is completed:

Example with Callback

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>

$(document).ready(function(){
$("button").click(function(){
        $("p").hide("slow", function(){
                alert("The paragraph is now hidden");
        });
});
});
</script>
</head>
<body>

<button>Hide</button>

<p>This is a paragraph with little content.</p>

</body>
</html>
```

The example below has no callback parameter, and the alert box will be displayed before the hide effect is completed:

Example without Callback

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("button").click(function(){
        $("p").hide(1000);
                alert("The paragraph is now hidden");
        });
});
</script>
</head>
<body>

<button>Hide</button>

<p>This is a paragraph with little content.</p>

</body>
</html>
```

jQuery HTML

jQuery – Set and Get Content and Attributes

jQuery contains powerful methods for changing and manipulating HTML elements and attributes.

jQuery DOM Manipulation

One very important part of jQuery is the possibility to manipulate the DOM.
jQuery comes with a bunch of DOM related methods that make it easy to access and manipulate elements and attributes.

DOM = Document Object Model

The DOM defines a standard for accessing HTML and XML documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

Get Content - text(), html(), and val()

Three simple, but useful, jQuery methods for DOM manipulation are:

- text() - Sets or returns the text content of selected elements
- html() - Sets or returns the content of selected elements (including HTML markup)
- val() - Sets or returns the value of form fields

The following example demonstrates how to get content with the jQuerytext() and html() methods:

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
	$("#btn1").click(function(){
		alert("Text: " + $("#test").text());
	});
	$("#btn2").click(function(){
		alert("HTML: " + $("#test").html());
	});
});
</script>
</head>
<body>
```

```
<p id="test">This is some <b>bold</b> text in a paragraph.</p>

<button id="btn1">Show Text</button>
<button id="btn2">Show HTML</button>

</body>
</html>
```

The following example demonstrates how to get the value of an input field with the jQuery val() method:

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("button").click(function(){
                alert("Value: " + $("#test").val());
        });
});
</script>
</head>
<body>

<p>Name: <input type="text" id="test" value="Mickey Mouse"></p>

<button>Show Value</button>

</body>
</html>
```

Set Content - text(), html(), and val()

We will use the same three methods from the previous page to **set content**:

- text() - Sets or returns the text content of selected elements
- html() - Sets or returns the content of selected elements (including HTML markup)
- val() - Sets or returns the value of form fields

The following example demonstrates how to set content with the jQuerytext(), html(), and val() methods:

Example ;

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
```

```
$("#btn1").click(function(){
                $("#test1").text("Hello world!");
        });
        $("#btn2").click(function(){
                $("#test2").html("<b>Hello world!</b>");
        });
        $("#btn3").click(function(){
                $("#test3").val("Dolly Duck");
        });
});
</script>
</head>
<body>

<p id="test1">This is a paragraph.</p>
<p id="test2">This is another paragraph.</p>


<p>Input field: <input type="text" id="test3" value="Mickey Mouse"></p>

<button id="btn1">Set Text</button>
<button id="btn2">Set HTML</button>
<button id="btn3">Set Value</button>

</body>
</html>
```

jQuery - Add Elements

With jQuery, it is easy to add new elements/content.

We will look at four jQuery methods that are used to add new content:

- append() - Inserts content at the end of the selected elements
- prepend() - Inserts content at the beginning of the selected elements
- after() - Inserts content after the selected elements
- before() - Inserts content before the selected elements

append()

The jQuery append() method inserts content AT THE END of the selected HTML elements.

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
```

```
<script>

$(document).ready(function(){
        $("#btn1").click(function(){
                $("p").append(" <b>Appended text</b>.");
        });

        $("#btn2").click(function(){
                 $("ol").append("<li>Appended item</li>");
        });
});
</script>
</head>
<body>

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

<ol>
<li>List item 1</li>
<li>List item 2</li>
<li>List item 3</li>
</ol>

<button id="btn1">Append text</button>
<button id="btn2">Append list items</button>
</body>
</html>
```

prepend()

The jQuery prepend() method inserts content AT THE BEGINNING of the selected HTML elements.

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("#btn1").click(function(){
      $("p").prepend("<b>Prepended text</b>. ");
        });
        $("#btn2").click(function(){
      $("ol").prepend("<li>Prepended item</li>");
        });
});
</script>
</head>
<body>
```

```
<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

<ol>
<li>List item 1</li>
<li>List item 2</li>
<li>List item 3</li>
</ol>

<button id="btn1">Prepend text</button>
<button id="btn2">Prepend list item</button>

</body>
</html>
```

after() and before()

The jQuery after() method inserts content AFTER the selected HTML elements.
The jQuery before() method inserts content BEFORE the selected HTML elements.

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("#btn1").click(function(){
    $("img").before("<b>Before</b>");
        });

        $("#btn2").click(function(){
    $("img").after("<i>After</i>");
        });
});
</script>
</head>
<body>

<imgsrc="/images/w3jquery.gif" alt="jQuery" width="100" height="140"><br><br>

<button id="btn1">Insert before</button>
<button id="btn2">Insert after</button>

</body>
</html>
```

jQuery - Remove Elements

With jQuery, it is easy to remove existing HTML elements

To remove elements and content, there are mainly two jQuery methods:

- remove() - Removes the selected element (and its child elements)
- empty() - Removes the child elements from the selected element

remove()

The jQuery remove() method removes the selected element(s) and its child elements.

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("button").click(function(){
        $("#div1").remove();
        });
});
</script>
</head>
<body>

<div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;">

This is some text in the div.
<p>This is a paragraph in the div.</p>
<p>This is another paragraph in the div.</p>

</div>
<br>

<button>Remove div element</button>

</body>
</html>
```

empty()

The jQuery empty() method removes the child elements of the selected element(s).

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
```

```
$("button").click(function(){
        $("#div1").empty();
        });
});
</script>
</head>
<body>

<div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;">

This is some text in the div.
<p>This is a paragraph in the div.</p>
<p>This is another paragraph in the div.</p>

</div>
<br>

<button>Empty the div element</button>

</body>
</html>
```

Filter the Elements to be Removed

The jQuery remove() method also accepts one parameter, which allows you to filter the elements to be removed.
The parameter can be any of the jQuery selector syntaxes.
The following example removes all <p> elements with class="test":

Example

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("button").click(function(){
                $("p").remove(".test");
        });
});
</script>
<style>
.test {
color: red;
font-size: 20px;
}
</style>
</head>
<body>

<p>This is a paragraph.</p>
<p class="test">This is another paragraph.</p>
```

```
<p class="test">This is another paragraph.</p>

<button>Remove all p elements with class="test"</button>

</body>
</html>
```

jQuery - Get and Set CSS Classes

jQuery has several methods for CSS manipulation. We will look at the following methods:

- addClass() - Adds one or more classes to the selected elements
- removeClass() - Removes one or more classes from the selected elements
- toggleClass() - Toggles between adding/removing classes from the selected elements
- css() - Sets or returns the style attribute

Example Stylesheet

The following stylesheet will be used for all the examples on this page:

```
.important {
        font-weight: bold;
        font-size: xx-large;
}

.blue {
        color: blue;
}
```

jQuery addClass() Method

The following example shows how to add class attributes to different elements. Of course you can select multiple elements, when adding classes:

Example 1 :

```
$("button").click(function(){
        $("h1, h2, p").addClass("blue");
        $("div").addClass("important");
});
```

You can also specify multiple classes within the addClass() method:

Example 2 :

```
$("button").click(function(){
        $("#div1").addClass("important blue");
});
```

jQuery removeClass() Method

The following example shows how to remove a specific class attribute from different elements:

Example :

```
$("button").click(function(){
        $("h1, h2, p").removeClass("blue");
});
```

jQuery toggleClass() Method

The following example will show how to use the jQuery toggleClass() method. This method toggles between adding/removing classes from the selected elements:

Example :

```
$("button").click(function(){
        $("h1, h2, p").toggleClass("blue");
});
```

jQuery css() Method

The css() method sets or returns one or more style properties for the selected elements.
To return the value of a specified CSS property, use the following syntax:

css("*propertyname*");

The following example will return the background-color value of the FIRST matched element:

Example :

```
$("p").css("background-color");
```

Set a CSS Property

To set a specified CSS property, use the following syntax:
css("*propertyname*","*value*");
The following example will set the background-color value for ALL matched elements:

Example :

```
$("p").css("background-color", "yellow");
```

Set Multiple CSS Properties

To set multiple CSS properties, use the following syntax:

css({"*propertyname*":"*value*","*propertyname*":"*value*",...});

The following example will set a background-color and a font-size for ALL matched elements:

Example :

```
$("p").css({"background-color": "yellow", "font-size": "200%"});
```

Self Evaluation 2

1. How to get the value of a textbox using jQuery.

2. Finds all elements that are empty.

3.Double click to toggle background color of a paragraph.

4. Disable the submit button until the visitor has clicked a check box.

Solutions

1. HTML Code:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-git.js"></script>
<title>How to get the value of a textbox using jQuery</title>
</head>
<body>
<input type="text" value="Input text here">
</body>
</html>
```

JavaScript Code:

```
$( "input" )
  .keyup(function() {

var tvalue = $( this ).val();
```

```
console.log(tvalue);
 })
.keyup();
```

2. HTML Code:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-git.js"></script>
<title>Finds all elements that are empty.</title>
</head>
<body>
<table border="1">
<tr><td>R1C1</td><td></td><td>R1C3</td></tr>
<tr><td>R2C1</td><td>R2C2</td><td></td></tr>
<tr><td></td><td>R3C2</td><td>R3C3</td></tr>
</table>
</body>
</html>
```

JavaScript Code:

```
$( "td:empty" )
.text( "Empty!" )
.css( "background", "rgb(255,230,210)" );
```

3. HTML Code:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-git.js"></script>
<title>JS Bin</title>
</head>
<body>
<p>Double click to toggle background color</p>
</body>
</html>
```

CSS Code:

```
p {
background: blue;
color: white;
 }
 p.dbl {
background: red;
color: white;
 }
```

JavaScript Code:

```
var pdbl = $( "p:first" );
pdbl.dblclick(function() {
pdbl.toggleClass( "dbl" );
```

```
});
```

4. HTML Code:

```
<!DOCTYPE html>
<html>
<head>
<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>
        <title>Disable/enable the form submit button</title>
</head>
<body>
<input id="accept" name="accept" type="checkbox" value="y"/>I accept<br>
<input id="submitbtn" disabled="disabled" name="Submit" type="submit" value="Submit" />
</body>
</html>
```

JavaScript Code:

```
$('#accept').click(function() {
        if ($('#submitbtn').is(':disabled')) {
                $('#submitbtn').removeAttr('disabled');
        } else {
                $('#submitbtn').attr('disabled', 'disabled');
        }
});
```

jQuery - AJAX Introduction

AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page.

What is AJAX?

AJAX = Asynchronous JavaScript and XML.
In short; AJAX is about loading data in the background and display it on the webpage, without reloading the whole page.
Examples of applications using AJAX: Gmail, Google Maps, Youtube, and Facebook tabs.

jQuery get() and post() Methods

The jQuery get() and post() methods are used to request data from the server with an HTTP GET or POST request.

HTTP Request: GET vs. POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

- **GET** - Requests data from a specified resource
- **POST** - Submits data to be processed to a specified resource

GET is basically used for just getting (retrieving) some data from the server. **Note:** The GET method may return cached data.
POST can also be used to get some data from the server. However, the POST method NEVER caches data, and is often used to send data along with the request.

jQuery $.get() Method

The $.get() method requests data from the server with an HTTP GET request.

Syntax:
$.get(*URL,callback*);

The required URL parameter specifies the URL you wish to request.
The optional callback parameter is the name of a function to be executed if the request succeeds.
The following example uses the $.get() method to retrieve data from a file on the server:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("button").click(function(){
        $.get("demo_test.asp", function(data, status){
                alert("Data: " + data + "\nStatus: " + status);
        });
        });

});

</script>
</head>
<body>

<button>Send an HTTP GET request to a page and get the result back</button>
```

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

The first parameter of $.get() is the URL we wish to request ("demo_test.asp").
The second parameter is a callback function. The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.

jQuery $.post() Method

The $.post() method requests data from the server using an HTTP POST request.

Syntax:
$.post(*URL,data,callback*);

The required URL parameter specifies the URL you wish to request.
The optional data parameter specifies some data to send along with the request.
The optional callback parameter is the name of a function to be executed if the request succeeds.
The following example uses the $.post() method to send some data along with the request:

Example :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script>
$(document).ready(function(){
        $("button").click(function(){
        $.post("demo_test_post.asp",
        {
        name: "Donald Duck",
        city: "Duckburg"
        },
        function(data,status){
        alert("Data: " + data + "\nStatus: " + status);
        });
        });
});
</script>
</head>
<body>

<button>Send an HTTP POST request to a page and get the result back</button>

</body>
</html>
```

jQuery - The noConflict() Method

What if you wish to use other frameworks on your pages, while still using jQuery?

As you already know; jQuery uses the **$** sign as a shortcut for jQuery.

There are many other popular JavaScript frameworks like: Angular, Backbone, Ember, Knockout, and more.

What if other JavaScript frameworks also use the $ sign as a shortcut?
If two different frameworks are using the same shortcut, one of them might stop working.
The jQuery team have already thought about this, and implemented the noConflict() method.
The noConflict() method releases the hold on the $ shortcut identifier, so that other scripts can use it.

Example 1 :

You can of course still use jQuery, simply by writing the full name instead of the shortcut:

```
$.noConflict();
jQuery(document).ready(function(){
        jQuery("button").click(function(){
jQuery("p").text("jQuery is still working!");
        });
});
```

Example 2 :

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
var jq = $.noConflict();
jq(document).ready(function(){
        jq("button").click(function(){
        jq("p").text("jQuery is still working!");
        });
});
</script>
</head>
<body>

<p>This is a paragraph.</p>
<button>Test jQuery</button>

</body>
</html>
```

Self Evaluation 3

1.Limit character input in the textarea including count.

2.Disable a link using jQuery.

3.Finds the checked radio input.

4.Write a jQuery Code to get a single element from a selection.

5.Set background-image using jQuery CSS property.

6.Create a Zebra Stripes table effect.

7.Move one DIV element inside another using jQuery.

8.Distinguish between left and right mouse click with jQuery

Solutions

1. HTML Code:

```
<!DOCTYPE html>
<html>
<head>
<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>
<title>Limit character input in the textarea including count</title>
</head>
<body>
<form>
<label>Maximum 15 characters</label>
<textarea style=”display:block;” maxlength="15"></textarea>
<span id="rchars">15</span> Character(s) Remaining
</form>
</body>
</html>
```

JavaScript Code:

```
var maxLength = 15;

$('textarea').keyup(function() {

var textlen = maxLength - $(this).val().length;

  $('#rchars').text(textlen);

});
```

2. HTML Code:

```
<a href="https://www.w3resource.com/jquery-exercises/">jQuery Exercises, Practice, Solution</a>
```

JavaScript Code:

```
$(document).ready(function(){
$('#button1').click(function(){
$("a").removeAttr('href');
});
});
```

3. HTML Code:

```
<!DOCTYPE html>

<html>

<head>

<script src="https://code.jquery.com/jquery-git.js"></script>

<title>Finds the checked radio input.</title>

</head>

<body>

<form>

<input type="radio" name="color" value="Red">Red

<input type="radio" name="color" value="Green">Green
```

```
<input type="radio" name="color" value="Blue">Blue

<input type="radio" name="color" value="Black">Black

</form>

<p></p>

</body>

</html>
```

JavaScript Code:

```
$( "input" ).on( "click", function() {

$( "p" ).html( $( "input:checked" ).val() + " is checked!" );

});
```

4. HTML Code:

```
<!DOCTYPE html>

<html>

<head>

<script src="//code.jquery.com/jquery-1.11.1.min.js"></script>

<meta charset="utf-8">

<title>Write a jQuery Code to get a single element from a selection</title>

</head>

<body>

<p> Click this paragraph to see it fade.</p>

</body>

</html>
```

JavaScript Code:

```
$( "p:first" ).click(function() {
```

```
  $( this ).fadeTo( 500, 0.40 );

});
```

Self Evaluation 4

1. What is jQuery?
2. Why do we use jQuery?
3. How JavaScript and jQuery are different?
4. Is jQuery replacement of Java Script?
5. Is jQuery a library for client scripting or server scripting?
6. Does jQuery follow W3C recommendations?
7. What is the basic need to start with jQuery?
8. Which is the starting point of code execution in jQuery?
9. What does dollar sign ($) means in jQuery?
10. Can we have multiple document.ready() function on the same page?
11. Can we use our own specific character in the place of $ sign in jQuery?
12. Is it possible to use other client side libraries like MooTools, Prototype along with jQuery?
13. What is jQuery.noConflict?
14. Is there any difference between body onload() and document.ready() function?
15. What is the difference between .js and .min.js?
16. Why there are two different version of jQuery library?
17. What is a CDN?
18. Which are the popular jQuery CDN? And what is the advantage of using CDN?
19. What are selectors in jQuery and how many types of selectors are there?
20. How do you select element by ID in jQuery?
21. What does $("div") will select?
22. How to select element having a particular class (".selected")?
23. What does $("div.parent") will select?
24. What are the slow selectors in jQuery?
25. How jQuery selectors are executed?
26. Which is fast document.getElementByID('txtName') or $('#txtName')?
27. Difference between $(this) and 'this' in jQuery?
28. How do you check if an element is empty?
29. How do you check if an element exists or not in jQuery?
30. What is the difference between jquery.size() and jquery.length?
31. What is the difference between $('div') and $(' ') in jQuery?
32. What is the difference between parent() and parents() methods in jQuery?
33. How do you implement animation functionality?
34. How do you stop the currently-running animation?
35. What is the difference between .empty(), .remove() and .detach() methods in jQuery?
36. What is wrong with this code line "$('#myid.3').text('blah blah!!!');"
37. How to create clone of any object using jQuery?
38. How to check data type of any variable in jQuery?
39. Can you include multiple version of jQuery? If yes then how they are executed?
40. What is chaining in jQuery?
41. You get "jquery is not defined" or "$ is not defined" error. What could be the reason?
42. Can we use jQuery to make ajax request?
43. Is there any advantage of using $.ajax() for ajax call against $.get() or $.post()?
44. Can we execute/run multiple Ajax request simultaneously in jQuery? If yes, then how?
45. Which is the latest version of jQuery library?

46. Is it possible to get value of multiple CSS properties in single statement?
47. Consider a scenario where things can be done easily with javascript, would you still prefer jQuery?
48. What is jQuery plugin and what is the advantage of using plugin?
49. What is jQuery UI?
50. What are the fastest selectors in jQuery?

Solutions

1. jQuery is fast, lightweight and feature-rich client side JavaScript Library/Framework which helps in to traverse HTML DOM, make animations, add Ajax interaction, manipulate the page content, change the style and provide cool UI effect. It is one of the most popular client side libraries and as per a survey it runs on every second website.

2. Due to following advantages.
· Easy to use and learn.
· Easily expandable.
· Cross-browser support (IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+)
· Easy to use for DOM manipulation and traversal.
· Large pool of built in methods.
· AJAX Capabilities.
· Methods for changing or applying CSS, creating animations.
· Event detection and handling.
· Tons of plug-ins for all kind of needs.

3. JavaScript is a language while jQuery is a library built in the JavaScript language that helps to use the JavaScript language.

4. No. jQuery is not a replacement of JavaScript. jQuery is a different library which is written on top of JavaScript. jQuery is a lightweight JavaScript library that emphasizes interaction between JavaScript and HTML.

5. Client side scripting.

6. No.

7. To start with jQuery, one needs to make reference of its library. The latest version of jQuery can be downloaded from jQuery.com.

8. The starting point of jQuery code execution is $(document).ready() function which is executed when DOM is loaded.

9. Dollar Sign is nothing but it's an alias for JQuery. Take a look at below jQuery code.
$(document).ready(function(){ }); Over here $ sign can be replaced with "jQuery" keyword.
jQuery(document).ready(function(){ });

10. YES. We can have any number of document.ready() function on the same page.

11. Yes. It is possible using jQuery.noConflict().

12. Yes.

13. As other client side libraries like MooTools, Prototype can be used with jQuery and they also use $() as their global function and to define variables. This situation creates conflict as $() is used by jQuery and other library as their global function. To overcome from such situations, jQuery has introduced jQuery.noConflict().

```
jQuery.noConflict(); // Use jQuery via jQuery(...)
jQuery(document).ready(function(){
jQuery("div").hide();
});
```

You can also use your own specific character in the place of $ sign in jQuery.

```
var $j = jQuery.noConflict(); // Use jQuery via jQuery(...)
$j(document).ready(function(){
$j("div").hide();
});
```

14. document.ready() function is different from body onload() function for 2 reasons.
1. We can have more than one document.ready() function in a page where we can have only one body onload function.
2. document.ready() function is called as soon as DOM is loaded where body.onload() function is called when everything gets loaded on the page that includes DOM, images and all associated resources of the page.

15. jQuery library comes in 2 different versions Production and Deployment. The deployment version is also known as minified version. So .min.js is basically the minified version of jQuery library file. Both the files are same as far as functionality is concerned. But .min.js is quite small in size so it loads quickly and saves bandwidth.

16. jQuery library comes in 2 different versions. 1. Production 2. Deployment The production version is quite useful at development time as jQuery is open source and if you want to change something then you can make those changes in production version. But the deployment version is minified version or compressed version so it is impossible to make changes in it. Because it is compressed, so its size is very less than the production version which affects the page load time.

17. A content delivery network or content distribution network (CDN) is a large distributed system of servers deployed in multiple data centers across the Internet. The goal of a CDN is to serve content to end-users with high availability and high performance.

18. There are 3 popular jQuery CDNs. 1. Google. 2. Microsoft 3. jQuery.
Advantage of using CDN.

- It reduces the load from your server.
- It saves bandwidth. jQuery framework will load faster from these CDN.
- The most important benefit is it will be cached, if the user has visited any site
- which is using jQuery framework from any of these CDN

19. To work with an element on the web page, first we need to find them. To find the html element in jQuery we use selectors. There are many types of selectors but basic selectors are:

- Name: Selects all elements which match with the given element Name.
- #ID: Selects a single element which matches with the given ID

.Class: Selects all elements which match with the given Class.

- Universal (*): Selects all elements available in a DOM.

20. To select element use ID selector. We need to prefix the id with "#" (hash symbol). For example, to select element with ID "txtName", then syntax would be, $('#txtName')

21. This will select all the div elements on page.

22. $('.selected'). This selector is known as class selector. We need to prefix the class name with "." (dot).

23. All the div element with parent class.

24. class selectors are the slow compare to ID and element.

25. Your last selector is always executed first. For example, in below jQuery code, jQuery will first find all the elements with class ".myCssClass" and after that it will reject all the other elements which are not in "p#elmID".

```
$("p#elmID .myCssClass");
```

26. Native JavaScript is always fast. jQuery method to select txtName "$('#txtName')" will internally makes a call to document.getElementByID('txtName'). As jQuery is written on top of JavaScript and it internally uses JavaScript only So JavaScript is always fast.

27. this and $(this) refers to the same element. The only difference is the way they are used. 'this' is used in traditional sense, when 'this' is wrapped in $() then it becomes a jQuery object and you are able to use the power of jQuery.

```
$(document).ready(function(){
        $('#spnValue').mouseover(function(){
                alert($(this).text());
        });
});
```

In below example, this is an object but since it is not wrapped in $(), we can't use jQuery method and use the native JavaScript to get the value of span element.

```
$(document).ready(function(){
        $('#spnValue').mouseover(function(){
                alert(this.innerText);
        });
 });
```

28. There are 2 ways to check if element is empty or not. We can check using ":empty" selector.

```
$(document).ready(function(){
        if ($('#element').is(':empty')){
                //Element is empty
                });
});
```

29. Using jQuery length property, we can ensure whether element exists or not.

```
$(document).ready(function(){
        if ($('#element').length > 0){
                //Element exists
        });

});
```

30. jQuery .size() method returns number of element in the object. But it is not preferred to use the size() method as jQuery provide .length property and which does the same thing. But the .length property is preferred because it does not have the overhead of a function call.

31. $(' ') : This creates a new div element. However this is not added to DOM tree unless you don't append it to any DOM element.
$('div') : This selects all div elements present on the page.

32. The basic difference is the parent() function travels only one level in the DOM tree, where parents() function search through the whole DOM tree.

33. The .animate() method allows us to create animation effects on any numeric CSS property. This method changes an element from one state to another with CSS styles. The CSS property value is changed gradually, to create an animated effect.
Syntax is : (selector).animate({styles},speed,easing,callback)

34. Using jQuery ".stop()" method.

35. All these methods .empty(), .remove() and .detach() are used for removing elements from DOM but they all are different. .empty(): This method removes all the child element of the matched element where remove() method removes set of matched elements from DOM. .remove(): Use .remove() when you want to remove the element itself, as well as everything inside it. In addition to the elements themselves, all bound events and jQuery data associated with the elements are removed. .detach(): This method is the same as .remove(), except that .detach() keeps all jQuery data associated with the removed elements. This method is useful when removed elements are to be reinserted into the DOM at a later time. Find out more here

36. The problem with above statement is that the selectors is having meta characters and to use any of the meta-characters (such as !"#$%&'()*+,./:;<=>?@[\]^`{|}~) as a literal part of a name, it must be escaped with with two backslashes: \\. For example, an element with id="foo.bar", can use the selector $("#foo\\.bar"). So the correct syntax is, $('#myid\\.3').text('blah blah!!!');

37. jQuery provides clone() method which performs a deep copy of the set of matched elements, meaning that it copies the matched elements as well as all of their descendant elements and text nodes.

```
$(document).ready(function(){
        $('#btnClone').click(function(){
                $('#dvText').clone().appendTo('body'); return false;
        });
});
```

38. Using $.type(Object) which returns the built-in JavaScript type for the object.

39. Yes. Multiple versions of jQuery can be included in same page.

40. Chaining is one of the most powerful features of jQuery. In jQuery, Chaining means to connect multiple functions, events on selectors. It makes your code short and easy to manage and it gives better performance. The chain starts from left to right. So left most will be called first and so on.

```
$(document).ready(function(){


        $('#dvContent').addClass('dummy');
        $('#dvContent').css('color', 'red');
```

```
        $('#dvContent').fadeIn('slow');
});
```

The above jQuery code sample can be re-written using chaining. See below.

```
$(document).ready(function(){
        $('#dvContent').addClass('dummy') .css('color', 'red') .fadeIn('slow');
});
```

Not only functions or methods, chaining also works with events in jQuery.

41. There could be many reasons for this. You have forgotten to include the reference of jQuery library and trying to access jQuery. You have included the reference of the jQuery file, but it is after your jQuery code. The order of the scripts is not correct. For example, if you are using any jQuery plugin and you have placed the reference of the plugin js before the jQuery library then you will face this error.

42. Yes. jQuery can be used for making ajax request.

43 . By using jQuery post()/ jQuery get(), you always trust the response from the server and you believe it is going to be successful all the time. Well, it is certainly not a good idea to trust the response. There may be n number of reason which may lead to failure of response. Where jQuery.ajax() is jQuery's low-level AJAX implementation. $.get and $.post are higher-level abstractions that are often easier to understand and use, but don't offer as much functionality (such as error callbacks). Find out more here.

44. Yes, it is possible to execute multiple Ajax request simultaneously or in parallel. Instead of waiting for first Ajax request to complete and then issue the second request is time consuming. The better approach to speed up things would be to execute multiple Ajax request simultaneously. Using jQuery .when() method which provides a way to execute callback functions based on one or more objects, usually Deferred objects that represent asynchronous events. Find out more here.

45. The latest version (when this book is written) of jQuery is 3.3.1.

46. Well, before jQuery 1.9 release it was not possible but one of the new feature of jQuery 1.9 was .css() multi-property getter.

```
var propCollection = $("#dvBox").css([ "width", "height", "backgroundColor" ]);
```

In this case, the propCollection will be an array and it will look something like this.

```
{
width: "100px",
height: "200px",
backgroundColor: "#FF00FF"
}
```

47. No. If things can be done easily via CSS or JavaScript then You should not think about jQuery. Remember, jQuery library always comes with xx kilobyte size and there is no point of wasting bandwidth.

48. A plug-in is piece of code written in a standard JavaScript file. These files provide useful jQuery methods which can be used along with jQuery library methods. jQuery plugins are quite useful as its piece of code which is already written by someone and re-usable, which saves your development time.

49. jQuery UI is a set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library that can be used to build interactive web applications.

50. ID and element selectors are the fastest selectors in jQuery.

Reference

https://www.w3schools.com/jquery
http://jqfundamentals.com/chapter/jquery-basics
https://www.tutorialspoint.com/jquery
https://www.w3resource.com/jquery-exercises