## Google CDN

An alternative method for including the jQuery library that's worth considering is via the Google Content Delivery Network (CDN). A CDN is a network of computers that are specifically designed to serve content to users in a fast and scalable manner. These servers are often distributed geographically, with each request being served by the nearest server in the network.

So, instead of hosting the jQuery files on your own web server as we did above, you have the option of letting Google pick up part of your bandwidth bill. You benefit from the speed and reliability of Google's vast infrastructure, with the added bonus of the option to always use the latest version of jQuery.

```
ascript"
.eapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
```

## Uncompressed or compressed?

If you had a poke around on the jQuery download page, you might have also spied a couple of different download format options: compressed (also called **minified**), and uncompressed (also called "development").

Typically, you'll want to use the minified version for your production code, where the jQuery source code is compressed: spaces and line breaks have been removed and variable names are shortened.

The result is exactly the same jQuery library, but contained in a JavaScript file that's much smaller than the original. This is great for reducing bandwidth costs for you, and speeding up page requests for the end user.

## The Document Object Model

One of the most powerful aspects of jQuery is its ability to make selecting elements in the DOM easy. The Document Object Model is a family-tree structure of sorts. HTML, like other markup languages, uses this model to describe the relationships of things on a page. When we refer to these relationships, we use the same terminology that we use when referring to family relationships—parents, children, and so on.

A simple example to understand how the family tree metaphor applies to a document:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
      <title>the title</title>
</head>
   <body>
      <div>
          <p>This is a paragraph.</p>
          <p>This is another paragraph.</p>
          <p>This is yet another paragraph.</p>
      </div>
   </body>
</html>
```

Here, <html> is the **ancestor** of all the other elements; in other words, all the other elements are **descendants** of <html>. The <head> and <body> elements are not only descendants, but **children** of <html>, as well.

Likewise, in addition to being the ancestor of <head> and <body>, <html> is also their **parent**. The <p> elements are children (and descendants) of <div>, descendants of <body> and <html>, and **siblings** of each other.

An important point to note before we begin is that the resulting set of elements from selectors and methods is always wrapped in a jQuery object. These jQuery objects are very easy to work with when we want to actually do something with the things that we find on a page.

We can easily bind **events** to these objects and add slick **effects** to them, as well as **chain** multiple modifications or effects together. Nevertheless, jQuery objects are different from regular DOM elements or node lists, and as such do not necessarily provide the same methods and properties for some tasks.

**The $() factory function**

The fundamental operation in jQuery is selecting a part of the document. This is done with the *$()* construct. Typically, it takes a string as a parameter, which can contain any CSS selector expression.

**Making Sure the Page is Ready: $(document).ready(function)**

While JavaScript provides the *load* event for executing code when a page is rendered, this event does not get triggered until all assets (images, script, frames and other embedded objects) have been completely loaded.

```
$(document).ready(function() {
      alert(‘ready event was triggered’);
});
```

With this code in place, an alert will be displayed when the page is loaded.

**Note:** If your script requires images to be loaded first before executing then you can use the *load()* method instead:

```
$(document).load(function() {
      alert(‘all assets have been loaded’);
});
```

The shortcut version would be:

```
$(function() {
      alert(‘all assets have been loaded’);
});
```