Introduction

26, Oct, 2014

Muhammad Saber Ahmadi

- Today's World Wide Web is a **dynamic environment**, and its users set a high bar for both **style** and **function** of sites. To build interesting, interactive sites, developers are turning to **JavaScript libraries** such as **jQuery** to **automate** common tasks and **simplify** complicated ones.

- Most of jQuery concepts are borrowed from the structure of **HTML** and **Cascading Style Sheets (CSS)**. The library's design lends itself to a **quick start** for designers with **little programming experience** since many web developers have more experience with these technologies than they do with JavaScript.

Sites using jQuery

- Google
- Amazon
- IBM
- 2dodeveloper
- Microsoft
- Twitter
- Dell,Inc
- Best Buy
- Trader Bots
- NBC
- Match
- ESPN
- CBS News
- …

What jQuery does

- **Access elements in a document**

  Without a JavaScript library, many lines of code must be written to traverse the **Document Object Model (DOM)** tree, and locate specific portions of an HTML document's structure. A robust and efficient selector mechanism is offered in jQuery for retrieving the exact piece of the document that is to be inspected or manipulated.

- **Modify the appearance of a web page**

  CSS offers a **powerful method** of influencing the way a
  document is **rendered**, but it **falls** short when **web browsers**
  do not all support the same standards. With jQuery,
  developers can **bridge this gap**, relying on the same
  standards support across all browsers.
  In addition, jQuery can **change the classes** or individual style properties
  applied to a portion of the document even **after** the page
  has been **rendered**.

- **Alter the content of a document**

  jQuery can **modify** the content of a document itself with a few keystrokes.

  ➢Text can be changed

  ➢ images can be inserted or swapped

  ➢lists can be reordered

  Or the **entire structure** of the HTML can be **rewritten** and **extended**.

- **Respond to a user's interaction**

  Even the most elaborate and powerful behaviors are not useful if we can't control when they take place.

  The jQuery library offers an elegant way to intercept a wide variety of events.

- **Animate changes being made to a document**

  To effectively implement such interactive behaviors, a designer must also provide visual **feedback** to the user.

  The jQuery library facilitates this by providing an **array of effects** such as **fades** and **wipes**, as well as a toolkit for crafting new ones.

- **Retrieve information from a server without refreshing a page**

  This code pattern has become known as **Asynchronous JavaScript And XML (AJAX)**, and assists web developers in crafting a responsive, feature-rich site. The jQuery library removes the browser-specific complexity from this process, allowing developers to focus on the server-end functionality.

- **Simplify common JavaScript tasks**

  In addition to all of the document-specific features of jQuery, the library provides enhancements to basic JavaScript constructs such as iteration and array manipulation.

Our first jQuery- powered web page

- **Downloading jQuery**

The official jQuery website (http://jquery.com/) is always the most up-to-date resource for code and news related to the library.

- **Adding the jQuery library to your pages**

  The jQuery library is stored a single JavaScript file, containing all the jQuery functions.
  It can be added to a web page with the following mark up:

```
<head>
<script type="text/javascript" src="jquery.js"></script>
</head>
```

- If you don't want to store the jQuery library on your own computer, you can use the **hosted jQuery library** from Google or Microsoft.
- **Google**

    -
      ```
      <script type="text/javascript"
      src="http://ajax.googlepis.com/ajax/libs/jquery/1.4.2/jquery.min.js">
      </script>
      ```
- **Microsfot**

    -
      ```
      <script type="text/javascript"
      src="http://ajax.microsoft.com/ajax/jquery/jquery-1.4.2.min.js"></script>
      ```

- **Setting up the HTML document**

  There are three pieces to most examples of jQuery usage: the HTML document itself, CSS files to style it, and JavaScript files to act on it.

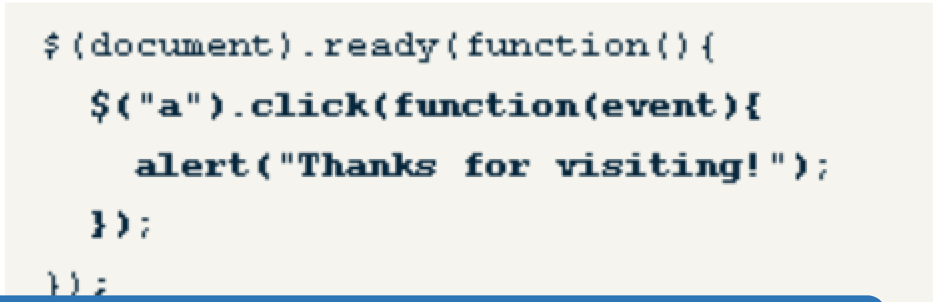
  Demo-1

Launching Code on Document

- window.onload = function(){ alert("welcome"); }

Inside of which is the code that you want to run right when the page is loaded. Problematically, however, the Javascript code isn't run until all **images** are finished downloading (this includes banner ads). The reason for using **window.onload** in the first place is that the HTML 'document' isn't finished loading yet, when you first try to run your code.

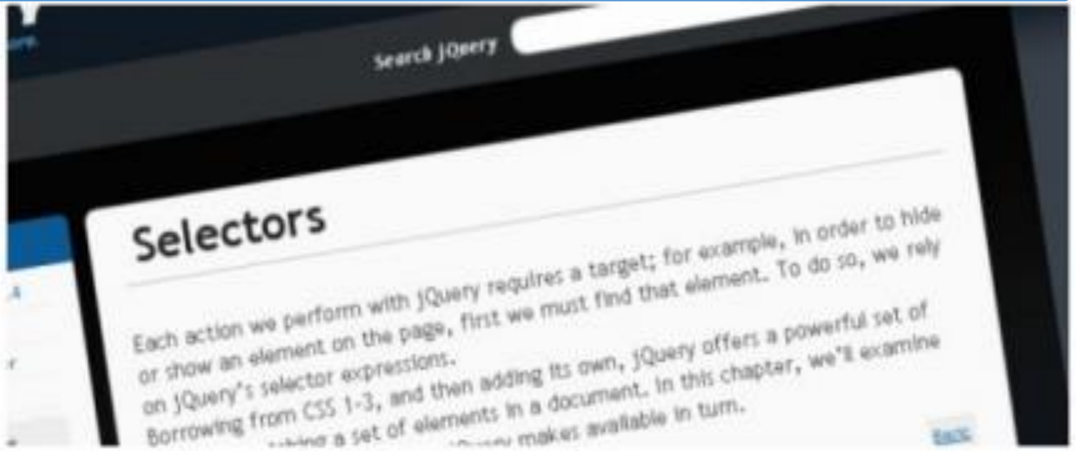
- jQuery has a simple statement that checks the document and waits until it's ready to be manipulated, known as the **ready event**:
  - $(document).ready(function(){
  - // Your code here
  - });



```
$(document).ready(function(){
  $("a").click(function(event){
    alert("Thanks for visiting!");
  });
});
```

Selectors

- **jQuery selectors** are one of the most important aspects of the jQuery library. These selectors use familiar CSS syntax to allow page authors to quickly and easily identify any set of page elements to operate upon with the jQuery library methods. **Understanding jQuery selectors is the key to using the jQuery library most effectively**.



Selectors

Each action we perform with jQuery requires a target; for example, in order to hide or show an element on the page, first we must find that element. To do so, we rely on jQuery's selector expressions.

Borrowing from CSS 1-3, and then adding its own, jQuery offers a powerful set of [illegible] a set of elements in a document. In this chapter, we'll examine [illegible] jQuery makes available in turn.

- A jQuery statement typically follows the syntax pattern: `$(selector).methodName();`
- The selector is a string expression that identifies the set of DOM elements that will be collected into a matched set to be operated upon by the jQuery methods.

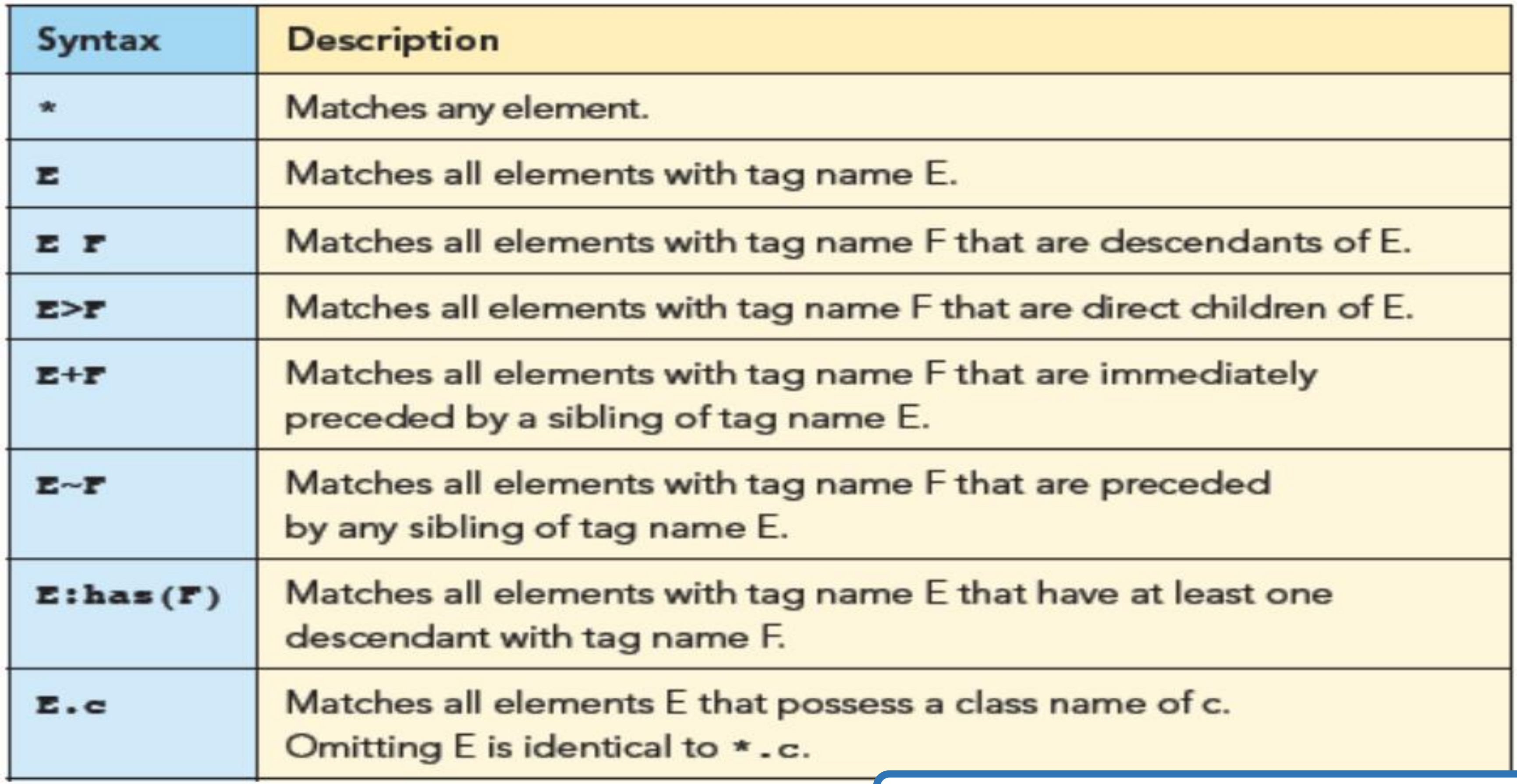
- Many of the jQuery operations can also be chained:

```
$(selector).method1().method2().method3();
```

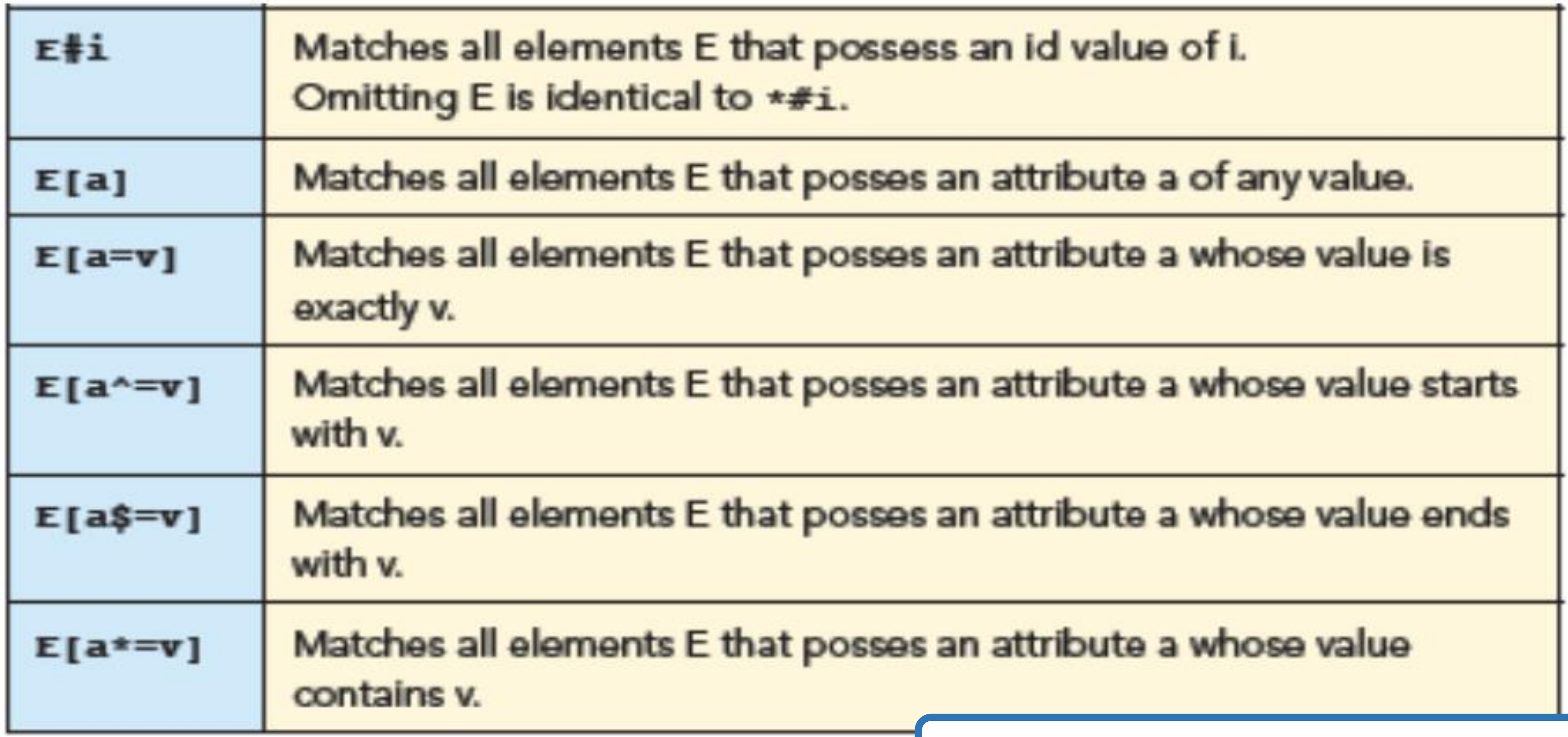
- E.g

```
$('#goAway').hide().addClass('incognito');
```

Basic CSS Selectors



Syntax	Description
`*`	Matches any element.
`E`	Matches all elements with tag name E.
`E F`	Matches all elements with tag name F that are descendants of E.
`E>F`	Matches all elements with tag name F that are direct children of E.
`E+F`	Matches all elements with tag name F that are immediately preceded by a sibling of tag name E.
`E~F`	Matches all elements with tag name F that are preceded by any sibling of tag name E.
`E:has(F)`	Matches all elements with tag name E that have at least one descendant with tag name F.
`E.c`	Matches all elements E that possess a class name of c. Omitting E is identical to `*.c`.



`E#i`	Matches all elements E that possess an id value of i. Omitting E is identical to `*#i`.
`E[a]`	Matches all elements E that posses an attribute a of any value.
`E[a=v]`	Matches all elements E that posses an attribute a whose value is exactly v.
`E[a^=v]`	Matches all elements E that posses an attribute a whose value starts with v.
`E[a$=v]`	Matches all elements E that posses an attribute a whose value ends with v.
`E[a*=v]`	Matches all elements E that posses an attribute a whose value contains v.

Examples

- `$('li>p')` selects all `<p>` elements that are direct childrer of `<li>` elements
- `$('div~p')` selects all `<div>` elements that are preceded by a `<p>` element
- `$('p:has(b)')` selects all `<p>` elements that contain a `<b>` element
- `$('div.someClass')` selects all `<div>` elements with a class name of `someClass`
- `$('.someClass')` selects all elements with class name someClass
- `$('#testButton')` selects the element with the `id` value of `testButton`
- `$('img[alt]')` selects all `<img>` elements that possess an `alt` attribute
- `$('a[href$=.pdf]')` selects all `<a>` elements that possess an `href` attribute that ends in `.pdf`
- `$('button[id*=test]')` selects all buttons whose `id` attributes contain `test`

- `$('div')` selects all `<div>` elements
- `$('fieldset a')` selects all `<a>` elements within `<fieldset>` elements

- `$(‘p:first’)` selects the first `<p>` element on the page
- `$(‘img[src$=.png]:first’)` selects the first `<img>` element on the page that has a `src` attribute ending in `.png`
- `$(‘button.small:last’)` selects the last `<button>` element on the page that has a class name of `small`
- `$(‘li:first-child’)` selects all `<li>` elements that are first children within their lists
- `$(‘a:only-child’)` selects all `<a>` elements that are the only element within their parent
- `$(‘li:nth-child(2)’)` selects all `<li>` elements that are the second item within their lists
- `$(‘tr:nth-child(odd)’)` selects all odd `<tr>` elements within a table
- `$(‘div:nth-child(5n)’)` selects every 5th `<div>` element
- `$(‘div:nth-child(5n+1)’)` selects the element after every 5th `<div>` element
- `$(‘.someClass:eq(1)’)` selects the second element with a class name of `someClass`
- `$(‘.someClass:gt(1)’)` selects all but the first two elements with a class name of `someClass`
- `$(‘.someClass:lt(4)’)` selects the first four elements with a class name of `someClass`

- For further examples and explanations

  please download or see **jquere_selectors.**