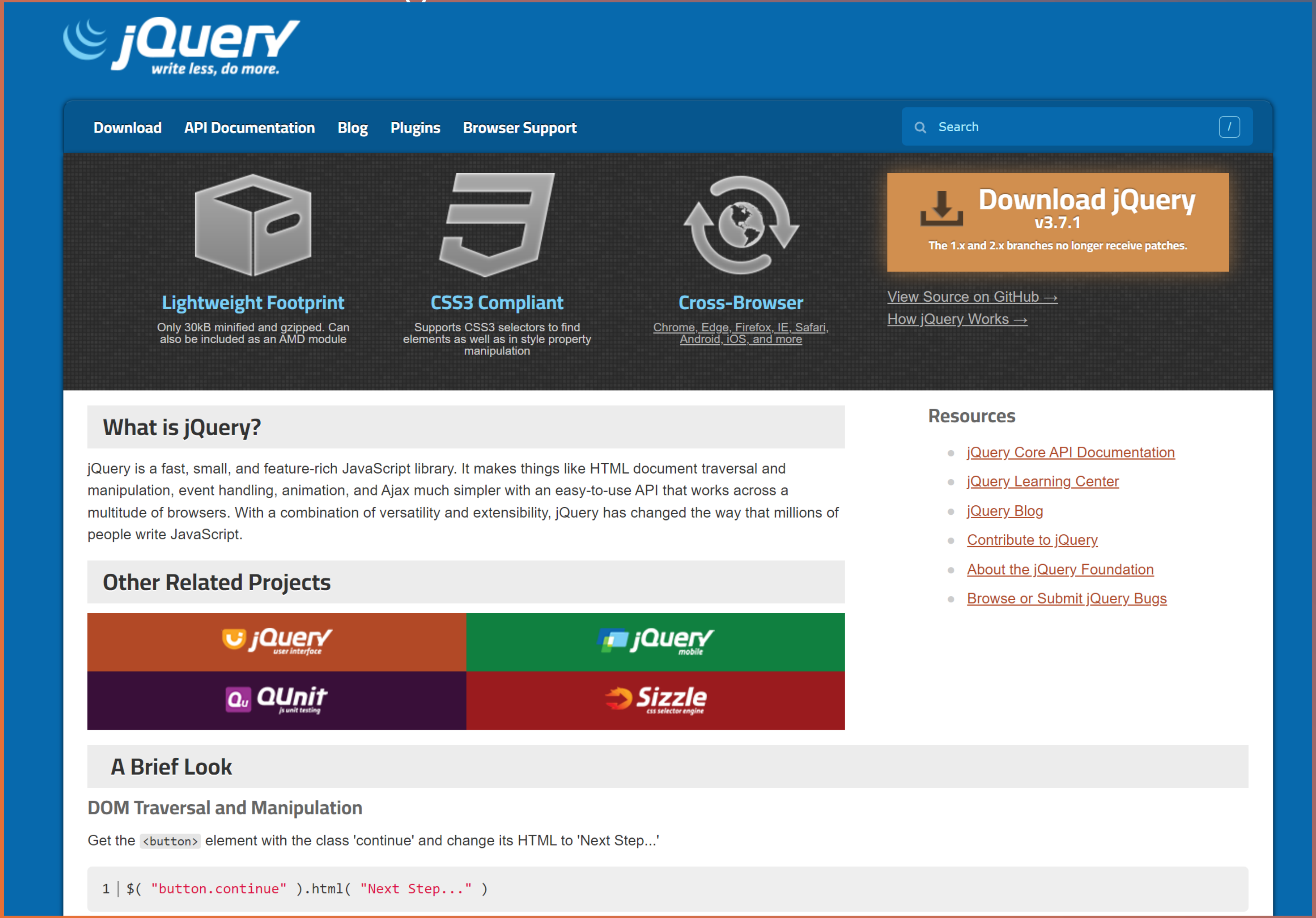
jQuery

What is jQuery



Lightweight Footprint

Only 30kB minified and gzipped. Can also be included as an AMD module

CSS3 Compliant

Supports CSS3 selectors to find elements as well as in style property manipulation

Cross-Browser

Chrome, Edge, Firefox, IE, Safari, Android, iOS, and more

Download jQuery v3.7.1

The 1.x and 2.x branches no longer receive patches.

View Source on GitHub →

How jQuery Works →

What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

Other Related Projects

Resources

- jQuery Core API Documentation
- jQuery Learning Center
- jQuery Blog
- Contribute to jQuery
- About the jQuery Foundation
- Browse or Submit jQuery Bugs

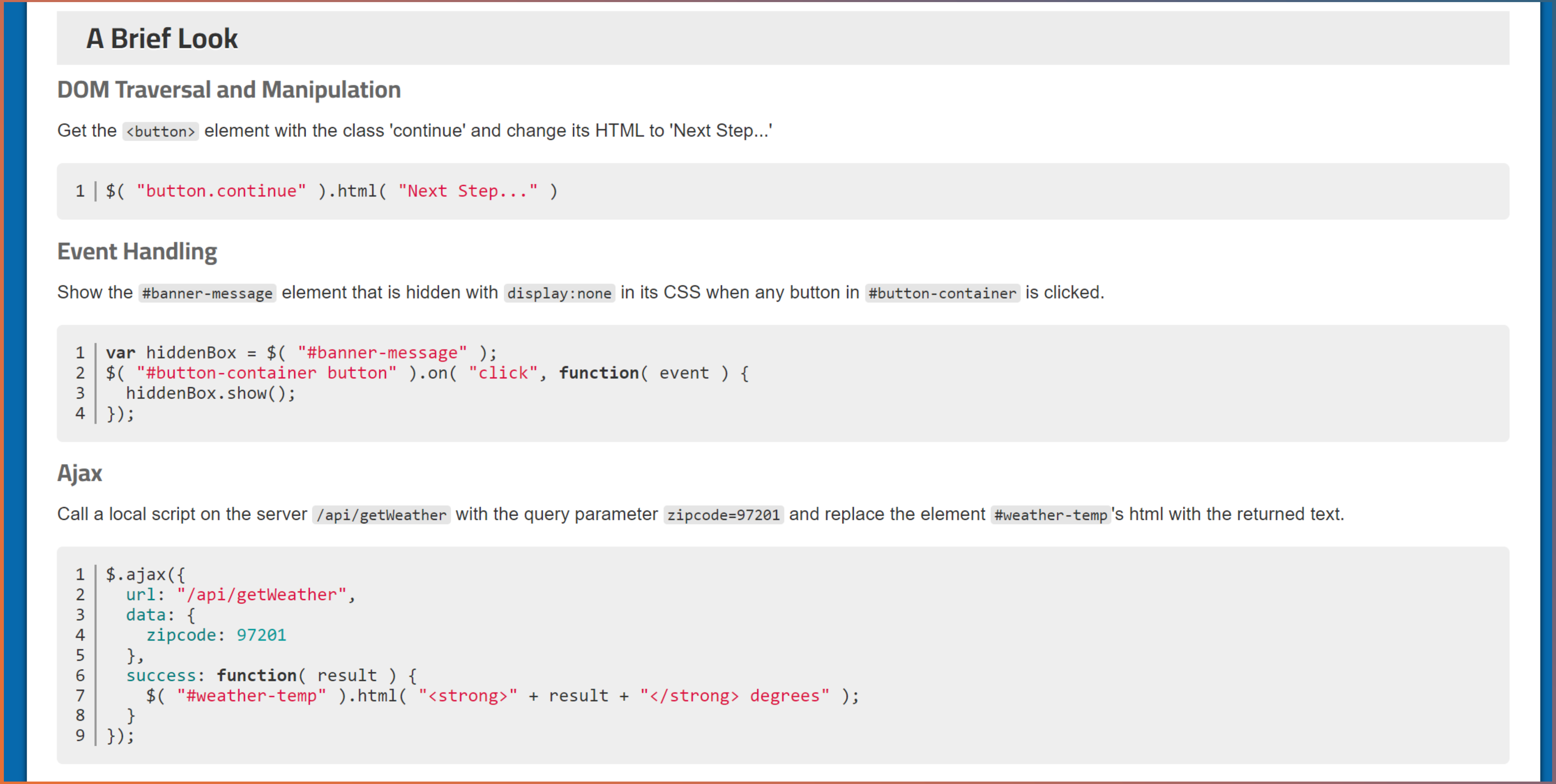
A Brief Look

DOM Traversal and Manipulation

Get the `<button>` element with the class 'continue' and change its HTML to 'Next Step...'

```
$( "button.continue" ).html( "Next Step..." )
```

A brief Look



A Brief Look

DOM Traversal and Manipulation

Get the `<button>` element with the class 'continue' and change its HTML to 'Next Step...'

```
$( "button.continue" ).html( "Next Step..." )
```

Event Handling

Show the `#banner-message` element that is hidden with `display:none` in its CSS when any button in `#button-container` is clicked.

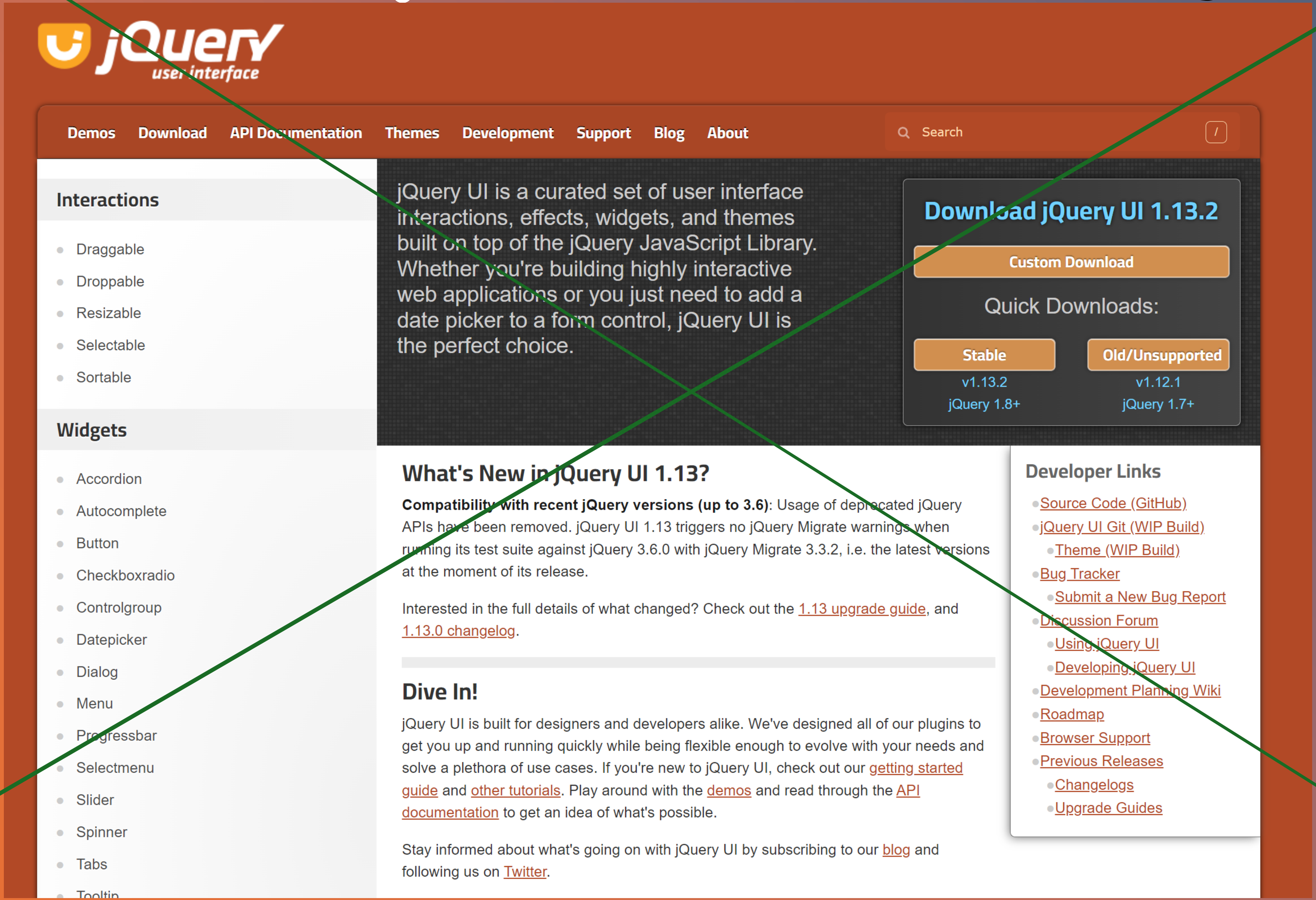
```
var hiddenBox = $( "#banner-message" );
$( "#button-container button" ).on( "click", function( event ) {
  hiddenBox.show();
});
```

Ajax

Call a local script on the server `/api/getWeather` with the query parameter `zipcode=97201` and replace the element `#weather-temp`'s html with the returned text.

```
$.ajax({
  url: "/api/getWeather",
  data: {
    zipcode: 97201
  },
  success: function( result ) {
    $( "#weather-temp" ).html( "<strong>" + result + "</strong> degrees" );
  }
});
```

jQuery UI



Interactions

- Draggable
- Droppable
- Resizable
- Selectable
- Sortable

Widgets

- Accordion
- Autocomplete
- Button
- Checkboxradio
- Controlgroup
- Datepicker
- Dialog
- Menu
- Progressbar
- Selectmenu
- Slider
- Spinner
- Tabs

jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice.

Download jQuery UI 1.13.2

Quick Downloads:

Stable	Old/Unsupported
v1.13.2	v1.12.1
jQuery 1.8+	jQuery 1.7+

What's New in jQuery UI 1.13?

Compatibility with recent jQuery versions (up to 3.6): Usage of deprecated jQuery APIs have been removed. jQuery UI 1.13 triggers no jQuery Migrate warnings when running its test suite against jQuery 3.6.0 with jQuery Migrate 3.3.2, i.e. the latest versions at the moment of its release.

Interested in the full details of what changed? Check out the 1.13 upgrade guide, and 1.13.0 changelog.

Dive In!

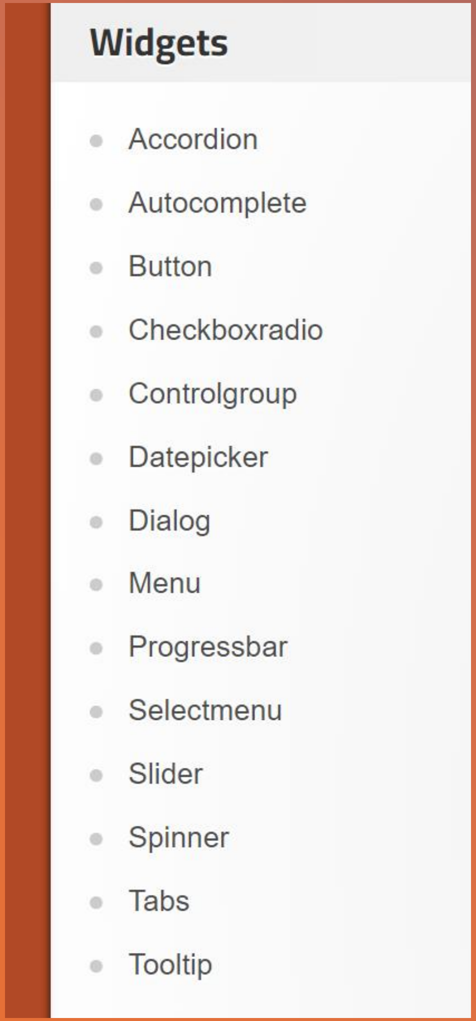
jQuery UI is built for designers and developers alike. We've designed all of our plugins to get you up and running quickly while being flexible enough to evolve with your needs and solve a plethora of use cases. If you're new to jQuery UI, check out our getting started guide and other tutorials. Play around with the demos and read through the API documentation to get an idea of what's possible.

Stay informed about what's going on with jQuery UI by subscribing to our blog and following us on Twitter.

Developer Links

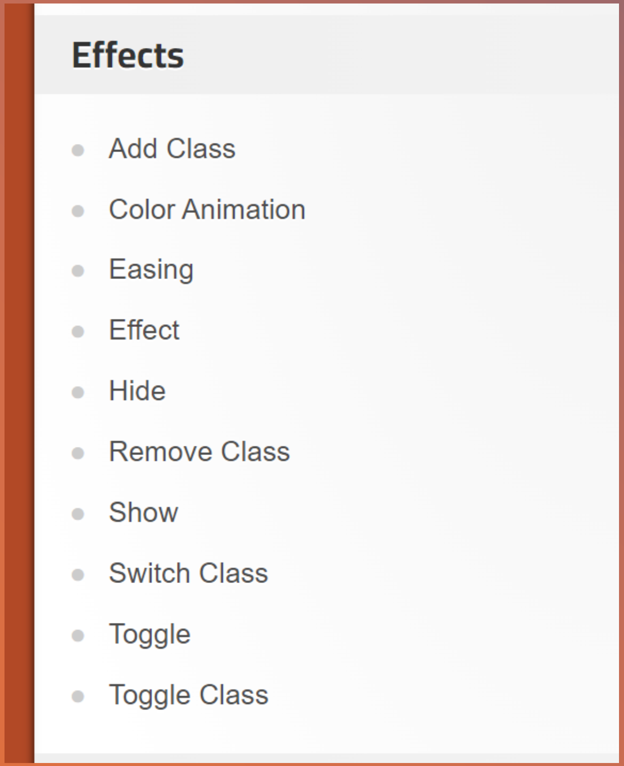
- Source Code (GitHub)
- jQuery UI Git (WIP Build)
  - Theme (WIP Build)
- Bug Tracker
  - Submit a New Bug Report
- Discussion Forum
  - Using jQuery UI
  - Developing jQuery UI
- Development Planning Wiki
- Roadmap
- Browser Support
- Previous Releases
  - Changelogs
  - Upgrade Guides

Legacy – Should be replaced by other libs (Bootstrap).



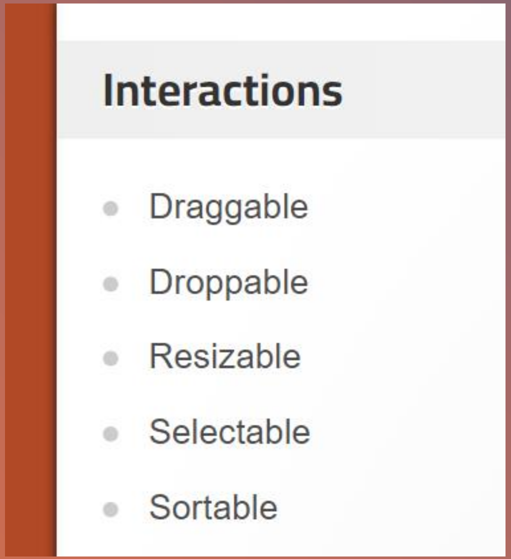
Widgets

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



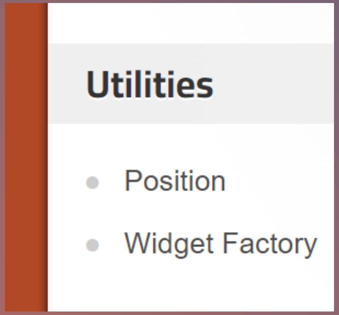
Effects

- Add Class
- Color Animation
- Easing
- Effect
- Hide
- Remove Class
- Show
- Switch Class
- Toggle
- Toggle Class



Interactions

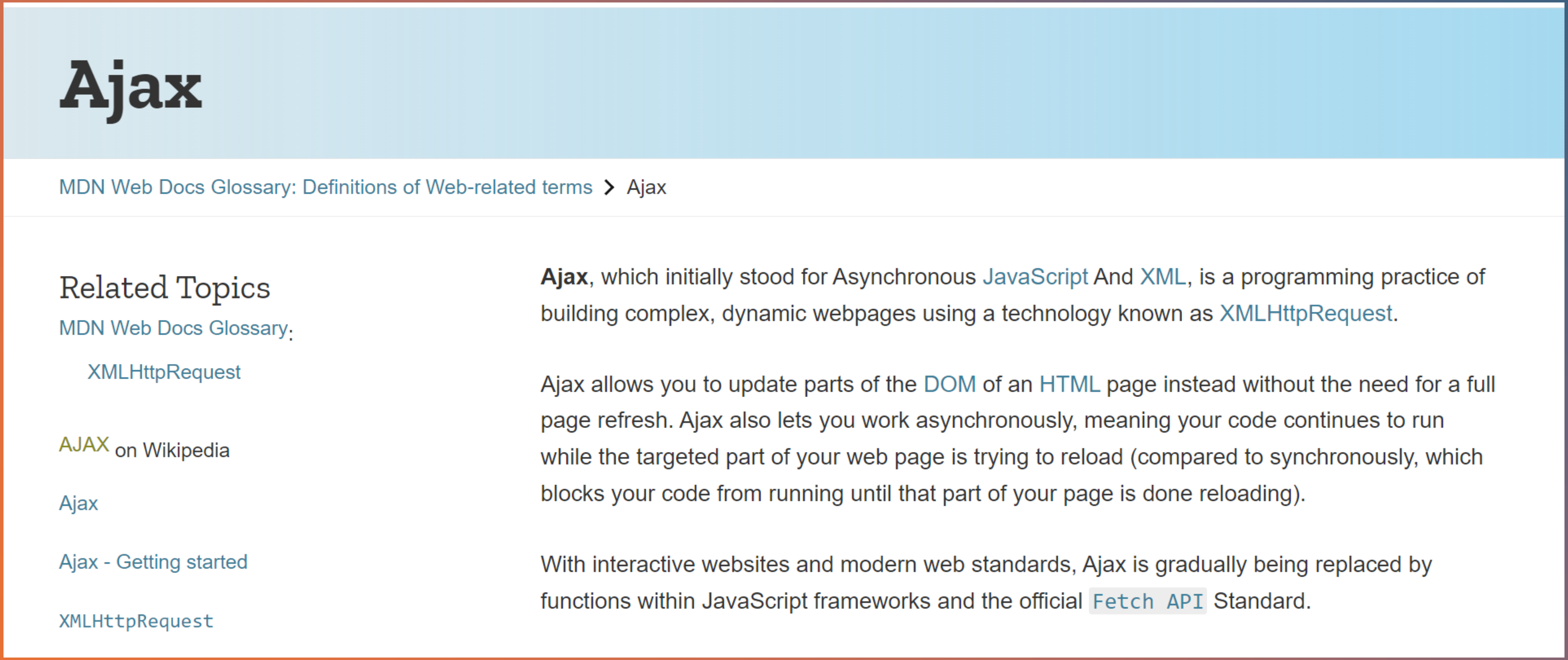
- Draggable
- Droppable
- Resizable
- Selectable
- Sortable



Utilities

- Position
- Widget Factory

Ajax –Asynchronous JavaScript and XML



Ajax

MDN Web Docs Glossary: Definitions of Web-related terms > Ajax

Related Topics

MDN Web Docs Glossary:

XMLHttpRequest

AJAX on Wikipedia

Ajax

Ajax - Getting started

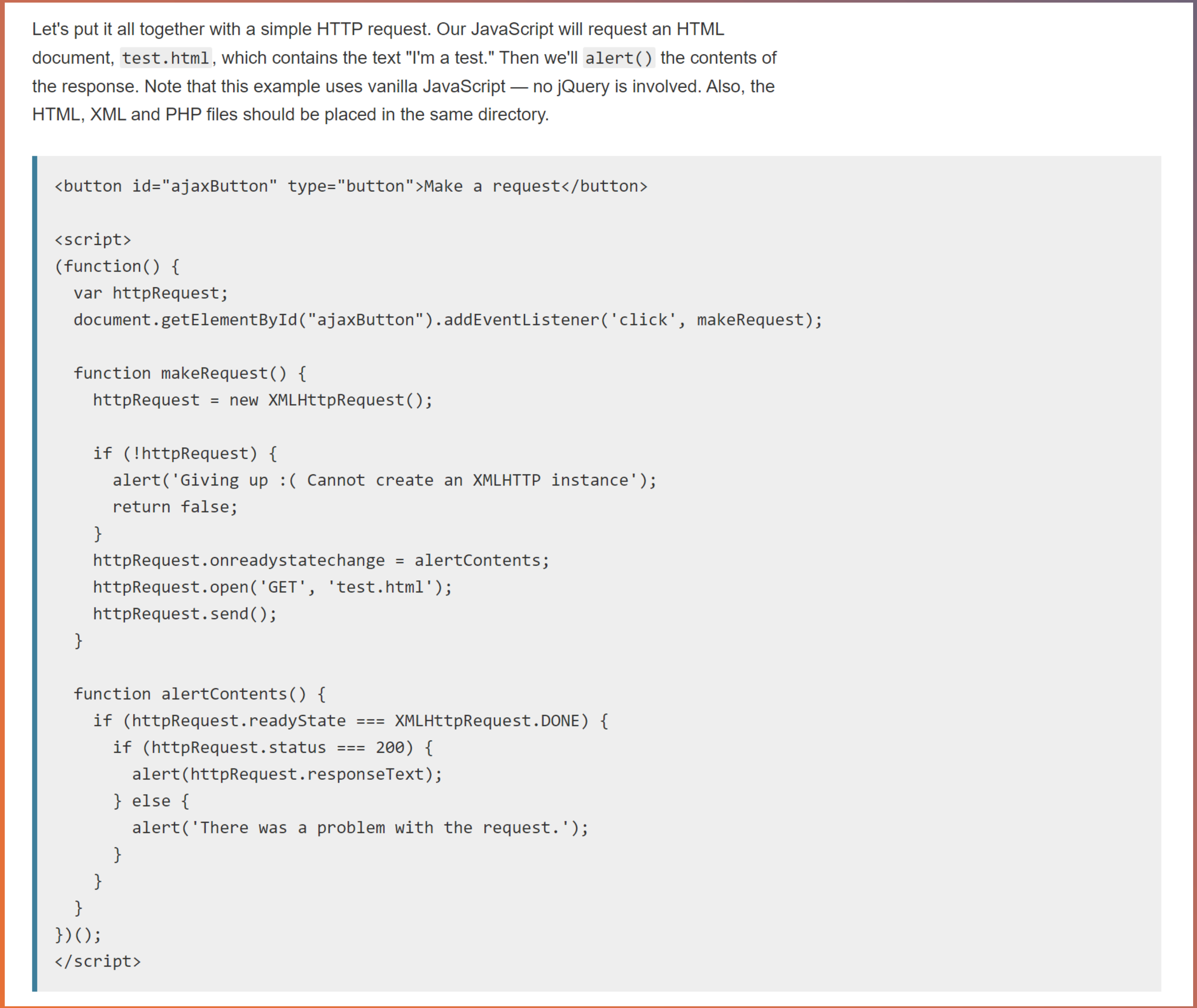
XMLHttpRequest

Ajax, which initially stood for Asynchronous JavaScript And XML, is a programming practice of building complex, dynamic webpages using a technology known as XMLHttpRequest.

Ajax allows you to update parts of the DOM of an HTML page instead without the need for a full page refresh. Ajax also lets you work asynchronously, meaning your code continues to run while the targeted part of your web page is trying to reload (compared to synchronously, which blocks your code from running until that part of your page is done reloading).

With interactive websites and modern web standards, Ajax is gradually being replaced by functions within JavaScript frameworks and the official `Fetch API` Standard.

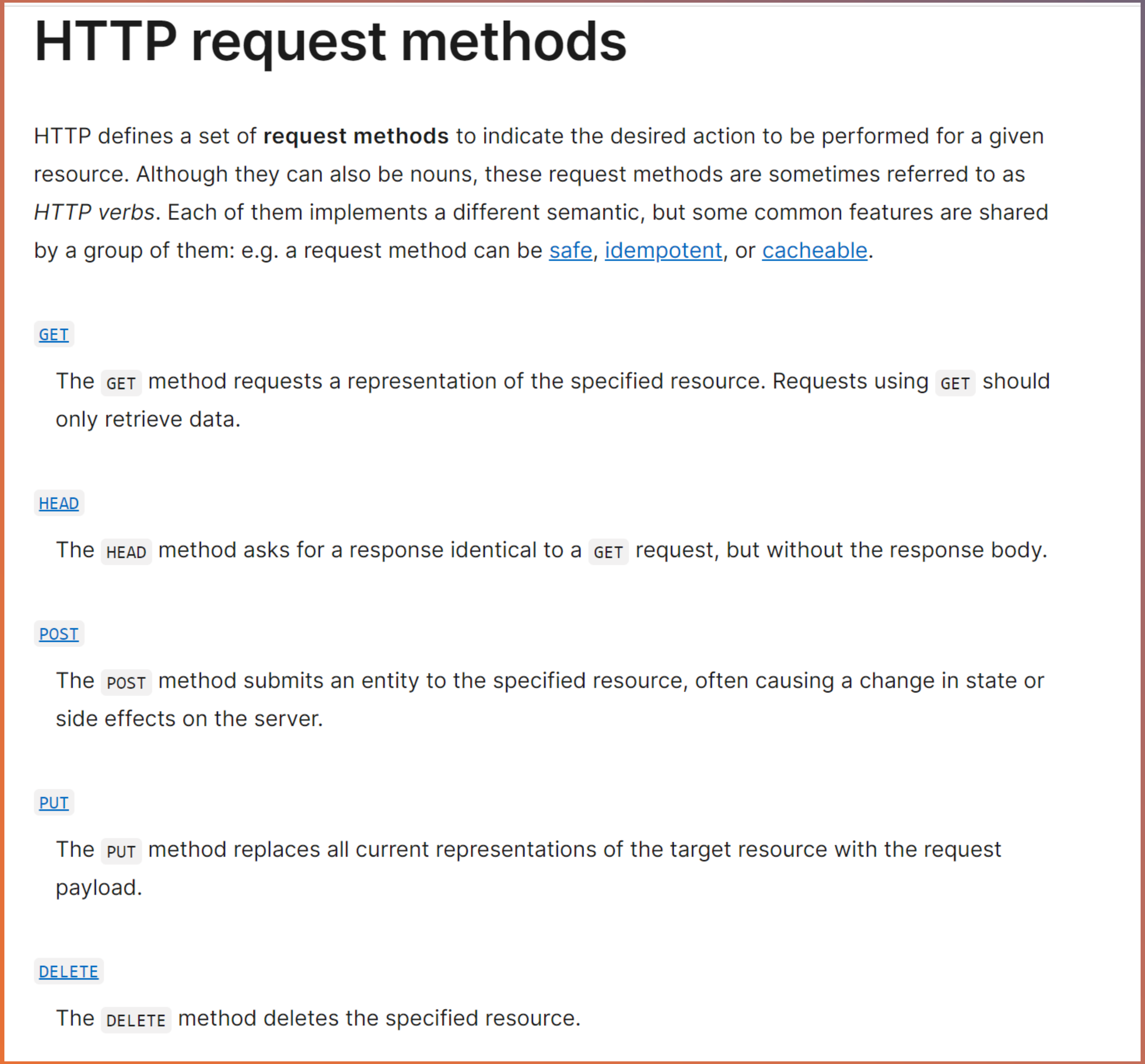
A Simple Example



Let's put it all together with a simple HTTP request. Our JavaScript will request an HTML document, `test.html`, which contains the text "I'm a test." Then we'll `alert()` the contents of the response. Note that this example uses vanilla JavaScript — no jQuery is involved. Also, the HTML, XML and PHP files should be placed in the same directory.

```
<button id="ajaxButton" type="button">Make a request</button>

<script>
(function() {
  var httpRequest;
  document.getElementById("ajaxButton").addEventListener('click', makeRequest);

  function makeRequest() {
    httpRequest = new XMLHttpRequest();

    if (!httpRequest) {
      alert('Giving up :( Cannot create an XMLHTTP instance');
      return false;
    }
    httpRequest.onreadystatechange = alertContents;
    httpRequest.open('GET', 'test.html');
    httpRequest.send();
  }

  function alertContents() {
    if (httpRequest.readyState === XMLHttpRequest.DONE) {
      if (httpRequest.status === 200) {
        alert(httpRequest.responseText);
      } else {
        alert('There was a problem with the request.');
      }
    }
  }
})();
</script>
```

HTTP Verbs



HTTP request methods

HTTP defines a set of **request methods** to indicate the desired action to be performed for a given resource. Although they can also be nouns, these request methods are sometimes referred to as *HTTP verbs*. Each of them implements a different semantic, but some common features are shared by a group of them: e.g. a request method can be safe, idempotent, or cacheable.

`GET`

The `GET` method requests a representation of the specified resource. Requests using `GET` should only retrieve data.

`HEAD`

The `HEAD` method asks for a response identical to a `GET` request, but without the response body.

`POST`

The `POST` method submits an entity to the specified resource, often causing a change in state or side effects on the server.

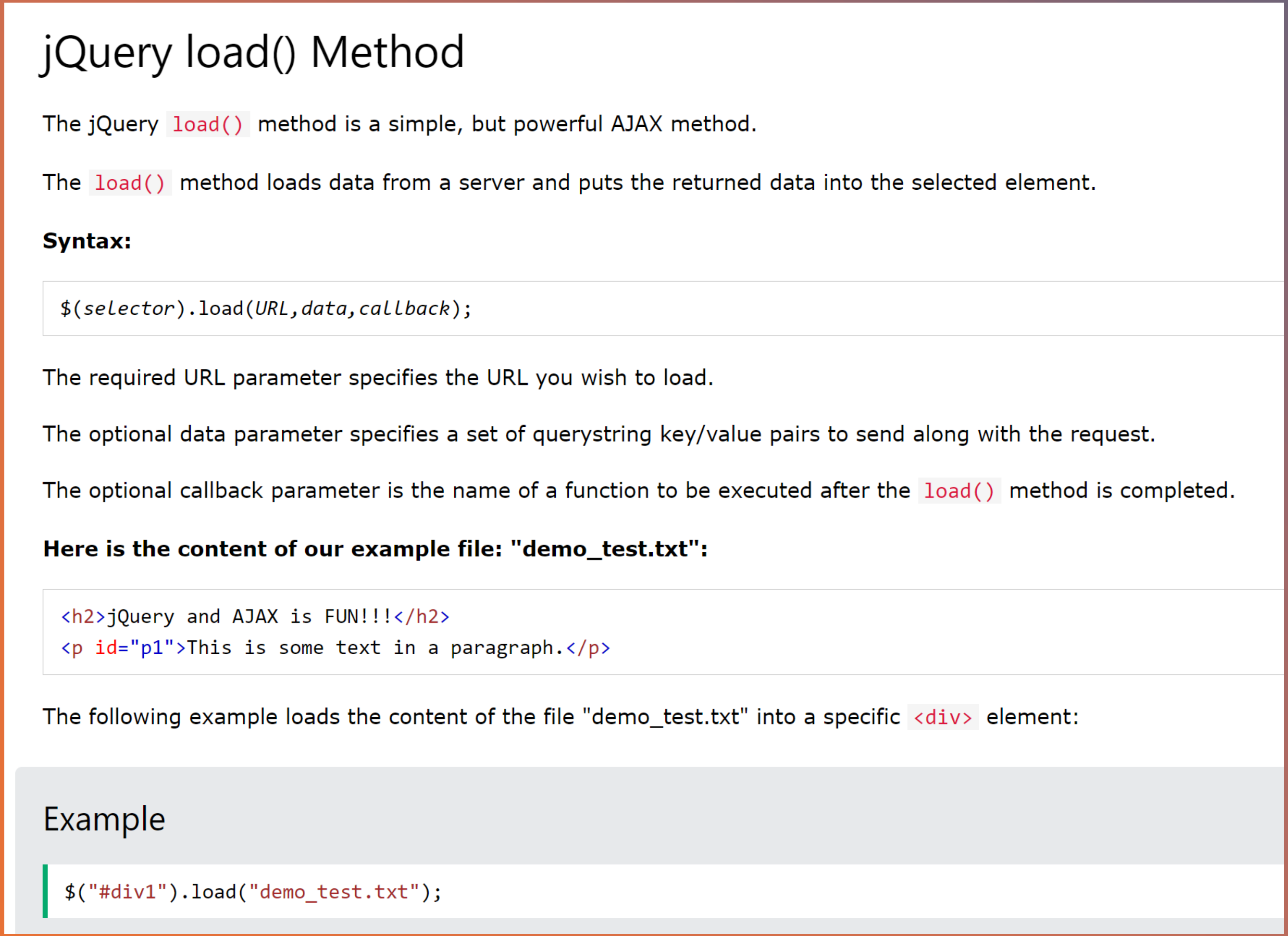
`PUT`

The `PUT` method replaces all current representations of the target resource with the request payload.

`DELETE`

The `DELETE` method deletes the specified resource.

Ajax with jQuery load() method



jQuery load() Method

The jQuery `load()` method is a simple, but powerful AJAX method.

The `load()` method loads data from a server and puts the returned data into the selected element.

Syntax:

```
$(selector).load(URL,data,callback);
```

The required URL parameter specifies the URL you wish to load.

The optional data parameter specifies a set of querystring key/value pairs to send along with the request.

The optional callback parameter is the name of a function to be executed after the `load()` method is completed.

Here is the content of our example file: "demo_test.txt":

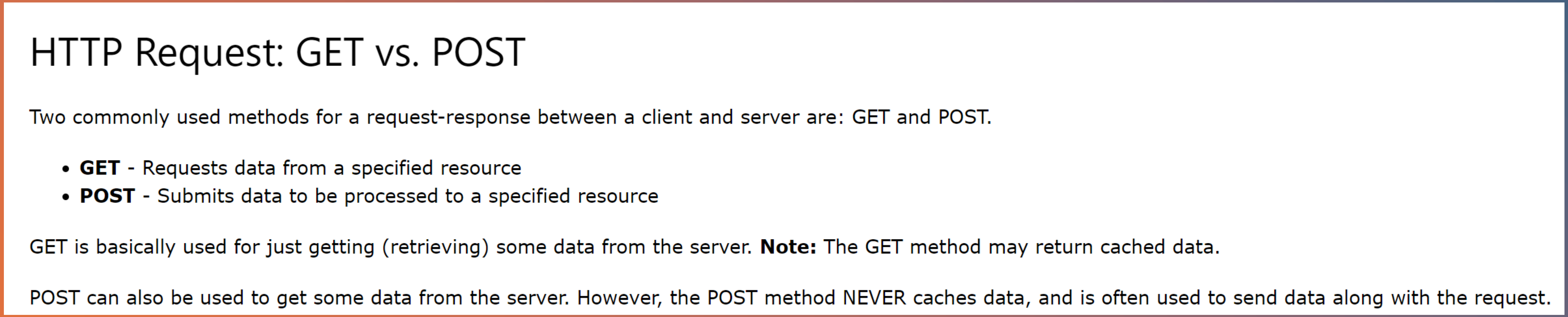
```
<h2>jQuery and AJAX is FUN!!!</h2>
<p id="p1">This is some text in a paragraph.</p>
```

The following example loads the content of the file "demo_test.txt" into a specific `<div>` element:

Example

```
$("#div1").load("demo_test.txt");
```

Ajax with jQuery AJAX get() and post() methods



HTTP Request: GET vs. POST

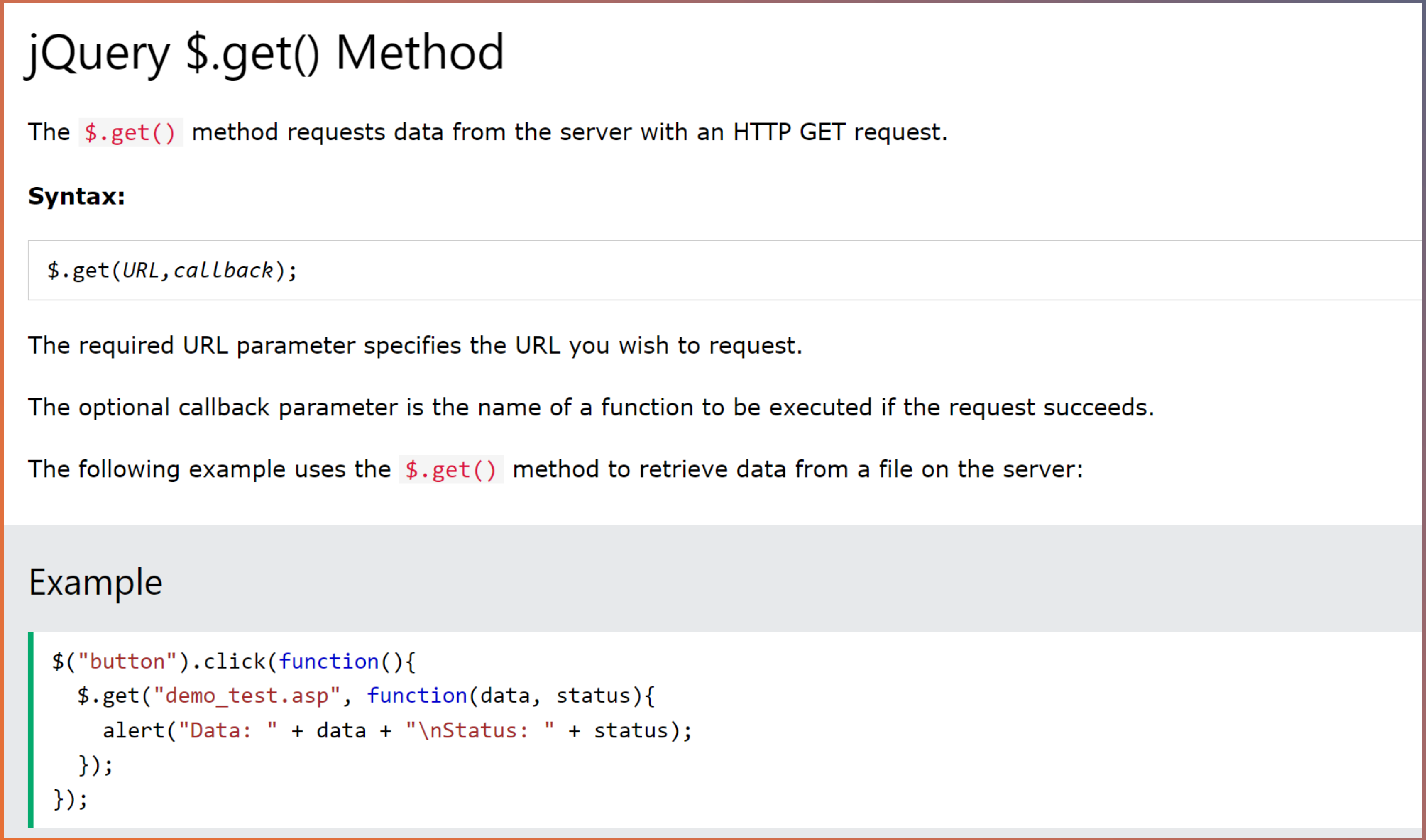
Two commonly used methods for a request-response between a client and server are: GET and POST.

- **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.

Ajax with jQuery AJAX get() method



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.

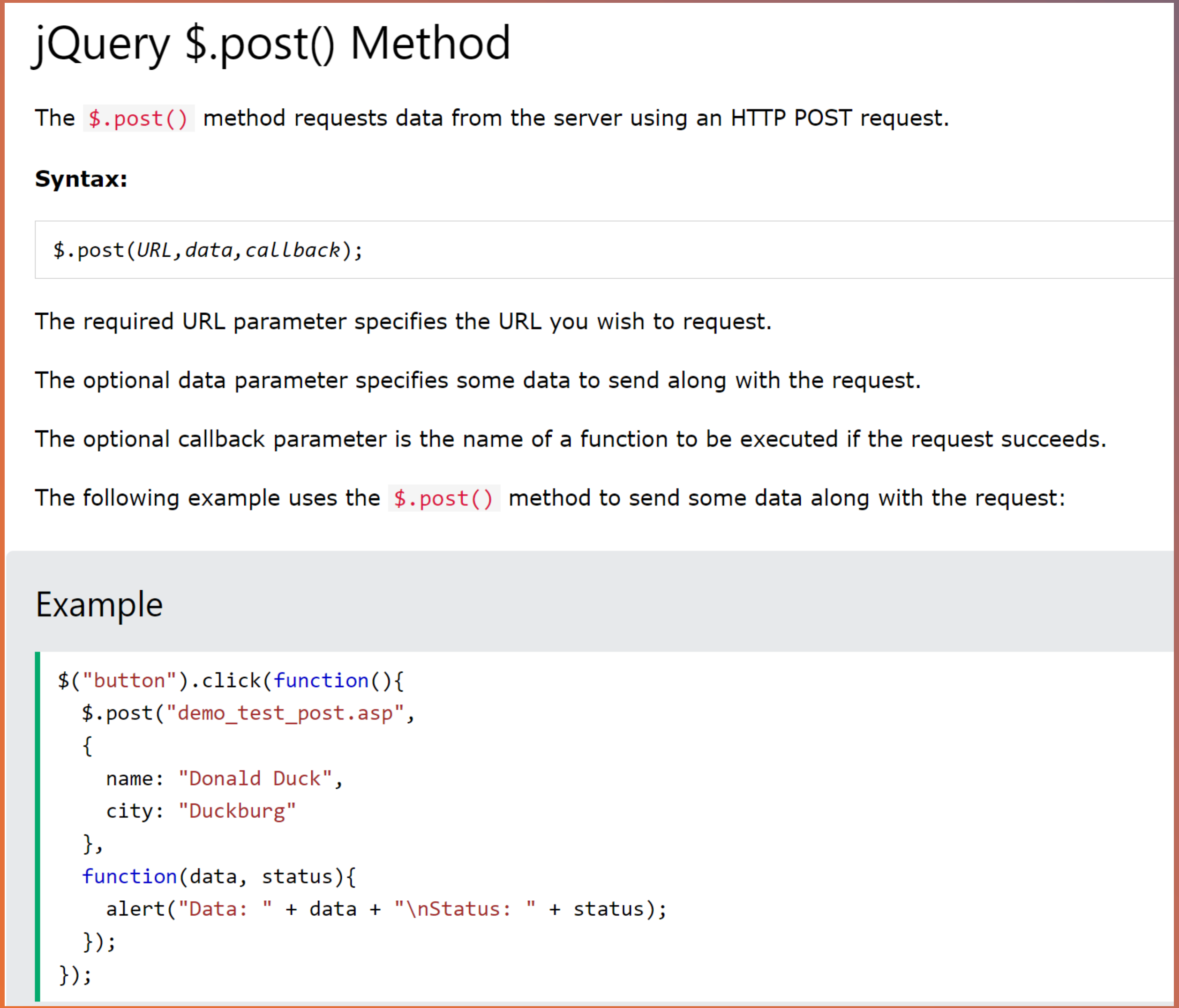
The optional callback parameter is the name of a function to be executed if the request succeeds.

The following example uses the `$.get()` method to retrieve data from a file on the server:

Example

```
$("button").click(function(){
  $.get("demo_test.asp", function(data, status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

Ajax with jQuery AJAX post() method



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

```
$("button").click(function(){
  $.post("demo_test_post.asp",
  {
    name: "Donald Duck",
    city: "Duckburg"
  },
  function(data, status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

Thank you