



#jQuery Notes

| Ajax

\$.ajax()

- \$.ajax() method allows you to send asynchronous http requests to submit or retrieve data from the server without reloading the whole page.
- \$.ajax() can be used to send http GET, POST, PUT, DELETE etc. request. It can retrieve any type of response from the server.



jqXHR (jQuery XMLHttpRequest) vs. XHR (XMLHttpRequest)

- jQuery 1.8 has brought a major change in how ajax are mode through jQuery. This change is the return type of \$.ajax() method.
- Previously till version 1.7, return type was XHR i.e. XMLHttpRequest, but from version 1.8 it's jqXHR i.e. jQuery XMLHttpRequest.
- In jQuery 1.8, library wraps the browser native XMLHttpRequest object with a superset API and return jqXHR object. jqXHR object simulates native XHR functionality as well as provides some more features e.g.
- It handles the HTTP request headers (Last-Modified, etag, Content-Type, MIME types etc...)
- It handles the callbacks of the request (including promise callbacks .done(), .fail() etc...)
- It handles any pre-filters set for the request
- It handles any timeouts set for the request
- It handles any cross domain calls (including jsonp)



jQuery AJAX Example (below v1.8)

```
$.ajax({  
  url: "/app-url/relative-url",  
  data: {  
    name : "The name",  
    desc : "The description"  
  },  
  success: function(data, textStatus, jqXHR)  
  {  
    alert("Success: " + response);  
  },  
  error: function(jqXHR, textStatus, errorThrown)  
  {  
    alert("Error");  
  }  
});
```



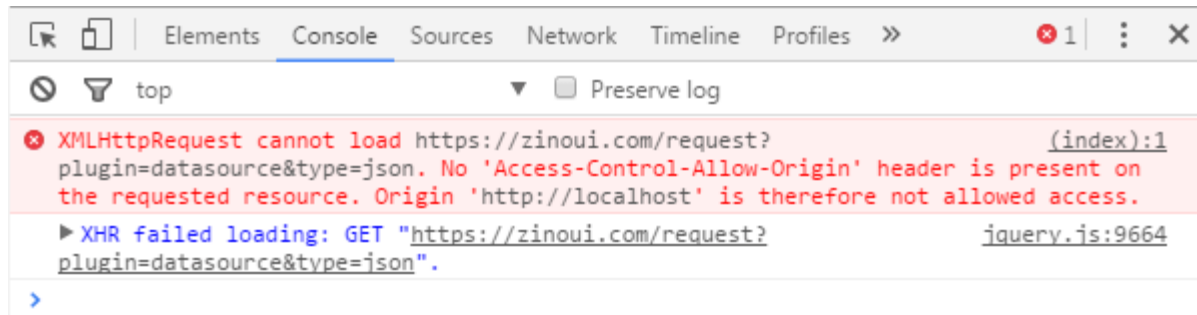
jQuery AJAX Example (since v1.8)

```
$.ajax({
  url: "/app-url/relative-url",
  data: {
    name : "The name",
    desc : "The description"
  }
})
.done (function(data, textStatus, jqXHR) {
  alert("Success: " + response);
})
.fail (function(jqXHR, textStatus, errorThrown) {
  alert("Error");
})
.always (function(jqXHROrData, textStatus,
jqXHROrErrorThrown) {
  alert("complete");
});
```



Cross Domain AJAX Request

- A common problem for developers is a browser to refuse access to a remote resource. Usually, this happens when you execute AJAX cross domain request using jQuery or plain XMLHttpRequest. As result is that the AJAX request is not performed and data are not retrieved.



XMLHttpRequest cannot load http://remote-domain/url. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'https://localhost:5433' is therefore not allowed access



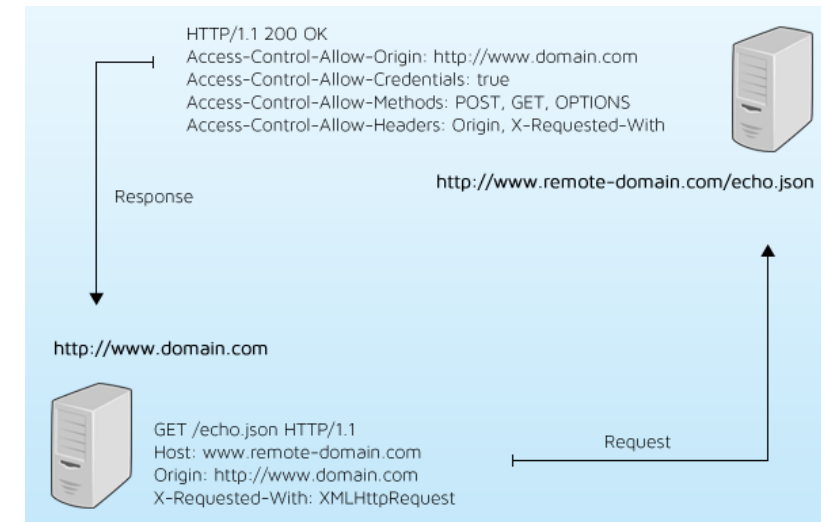
SAME-ORIGIN POLICY

- This is a security policy who defines the rules of how a web page can access an external resource (e.g. fonts, AJAX requests).
- Under the same-origin policy, web browsers do not permit a web page to access resources who origin differ than that of the current page.
- The origin is considered to be different when the scheme, hostname or port of the resource do not match that of the page.
- Overcoming the limitations of same-origin security policy is possible using a technique called Cross-origin resource sharing or simply CORS.



CROSS-ORIGIN RESOURCE SHARING

- CORS is a mechanism that defines a procedure in which the browser and the web server interact to determine whether to allow a web page to access a resource from different origin.
- When you do a cross-origin request, the browser sends Origin header with the current domain value.
 - Access-Control-Allow-Origin: http://abc.com
- If you want to allow access for all, use a wildcard '*'
 - Access-Control-Allow-Origin: *



ALLOW ACCESS FROM ALL DOMAINS

// Raw header

Access-Control-Allow-Origin: *

// How to send the response header with PHP

header("Access-Control-Allow-Origin: *");

// How to send the response header with Apache (.htaccess)

Header set Access-Control-Allow-Origin "*"

// How to send the response header with Nginx

add_header 'Access-Control-Allow-Origin' '*';

// How to send the response header with Express.js

```
app.use(function(req, res, next) {  
  res.header("Access-Control-Allow-Origin", "*");  
  next();  
});
```



Solution

- <https://chrome-allow-file-access-from-file.com/windows.html>
- Get the url of your Chrome Installation **path to your chrome installation** e.g C:\Users\your-user-name\AppData\Local\Google\Chrome\Application>
- Launch the Google Chrome browser from the command line window with the additional argument '**--allow-file-access-from-files**'. E.g '**path to your chrome installation\chrome.exe --allow-file-access-from-files**'
- Use LiveServer Extension in VS Code



jQuery ajax() Method

- The jQuery ajax() method provides core functionality of Ajax in jQuery. It sends asynchronous HTTP requests to the server.
- Using the **.ajax() function** you can fetch data from Text, HTML, JSON, XML or any external file with no page reloading. Once this data is fetched, it can be easily shown in a div, textbox or any other HTML control.

- `$.ajax(url);`

- `$.ajax(url,[options]);`

- url: A string URL to which you want to submit or retrieve the data
- options: Configuration options for Ajax request. An options parameter can be specified using JSON format. This parameter is optional.



The following table list all the options available for configuring Ajax request.

Options	Description
accepts	The content type sent in the request header that tells the server what kind of response it will accept in return.
async	By default, all requests are sent asynchronously. Set it false to make it synchronous.
beforeSend	A callback function to be executed before Ajax request is sent.
cache	A boolean indicating browser cache. Default is true.
complete	A callback function to be executed when request finishes.
contentType	A string containing a type of content when sending MIME content to the server.Default is "application/x-www-form-urlencoded; charset=UTF-8"
crossDomain	A boolean value indicating whether a request is a cross-domain.
data	A data to be sent to the server. It can be JSON object, string or array.
dataType	The type of data that you're expecting back from the server.
error	A callback function to be executed when the request fails.
global	A Boolean indicating whether to trigger a global Ajax request handler or not. Default is true.
headers	An object of additional header key/value pairs to send along with request.
ifModified	Allow the request to be successful only if the response has changed since the last request. This is done by checking the Last-Modified header. Default value is false.
isLocal	Allow the current environment to be recognized as local.



jsonp	Override the callback function name in a JSONP request. This value will be used instead of 'callback' in the 'callback=?' part of the query string in the url.
jsonpCallback	String containing the callback function name for a JSONP request.
mimeType	String containing a mime type to override the XMLHttpRequest mime type.
password	A password to be used with XMLHttpRequest in response to an HTTP access authentication request.
processData	A Boolean indicating whether data assigned to data option will be converted to a query string. Default is true.
statusCode	A JSON object containing numeric HTTP codes and functions to be called when the response has the corresponding code.
success	A callback function to be executed when Ajax request succeeds.
timeout	A number value in milliseconds for the request timeout.
type	A type of http request e.g. POST, PUT and GET. Default is GET.
url	A string containing the URL to which the request is sent.
username	A username to be used with XMLHttpRequest in response to an HTTP access authentication request.
xhr	A callback for creating the XMLHttpRequest object.
xhrFields	An object of fieldName-fieldValue pairs to set on the native XMLHttpRequest object.



Important Key Value

Name	Value
url	The URL to where AJAX request is made.
Type	The HTTP method used – POST or GET.
Datatype	The type of data returned from the AJAX request. Can be xml, json, script, or html.
success(result,status,xhr)	The function to call when the AJAX request is successful. All 3 parameters are optional. The result parameter will get the return value from the AJAX request.
error(xhr,status,error)	The function to call when AJAX request fails. All 3 parameters are optional. Helps in debugging during error.
data: '{"k1":"v1","k2":"v2",...}'	The key-value to be passed with the AJAX request.



Load Content

```
<html>
<head>
  <script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.js"></script>
  <script>
    $(document).ready(function () {
      $("#loadTextFile").click(function (e) {
        $.ajax({
          url: "file.txt",
          success: function (result,status,xhr) {
            $("#textData").html(result);
          }
        });
      });
    });
  </script>
</head>
<body>
  <button id="loadTextFile">Load Data</button>
  <div id="textData"></div>
</body>
</html>
```

Load Data

Load Data

I am Content



jQuery AJAX Method to Fetch Contents of a Text File

```
<button id="loadTextFile">Try</button>
```

```
<div id="textData"></div>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
```

```
<script>
```

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

```
    $("#loadTextFile").click(function (e) {
```

```
        $.ajax({
```

```
            url: "file.txt",
```

```
            success: function (result,status,xhr) {
```

```
                $("#textData").html(result);
```

```
            }
```

```
        });
```

```
    });
```

```
});
```

```
</script>
```



jQuery AJAX Method to Fetch Contents of a HTML File

```
<html>
<head>
  <script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.js"></script>
  <script>
    $(document).ready(function () {
      $("#LoadData").click(function (e) {
        $.ajax({
          url: "demo.html",
          success: function (result,status,xhr) {
            $("#myData").html(result);
          }
        });
      });
    });
  </script>
</head>
<body>
  <button id="LoadData">Load Data</button>
  <div id="myData"></div>
</body>
</html>
```

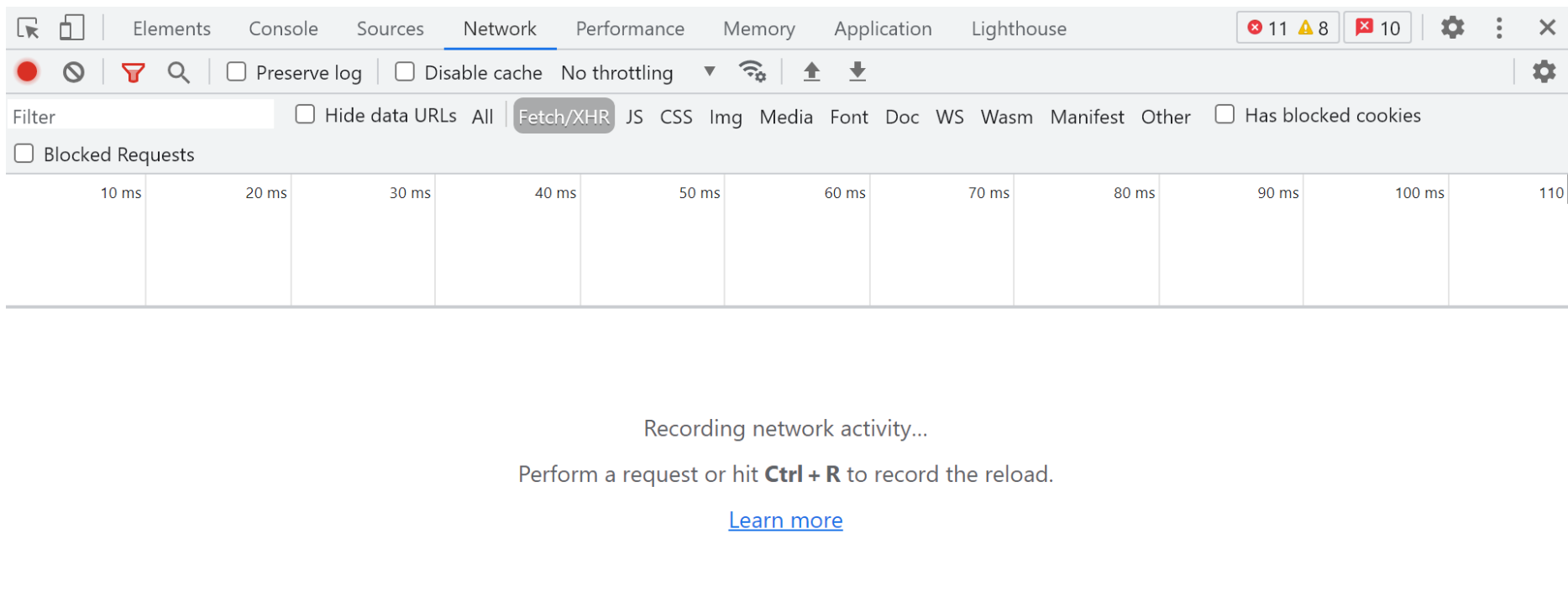


```
<html>
<head>
  <script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.js"></script>
  <script>
    $(document).ready(function () {
      $("#LoadData").click(function (e) {
        $.ajax({
          url: "demo.html",
          success: function (result,status,xhr) {
            $("#myData").html(result);
          }
        });
      });
    });
  </script>
</head>
<body>
  <button id="LoadData">Load Data</button>
  <div id="myData"></div>
</body>
</html>
```



How do I see my requests?

- you can access with the shortcut Alt+Command+Q or Control+Alt+Q. HTTP requests appear under the Network tab.

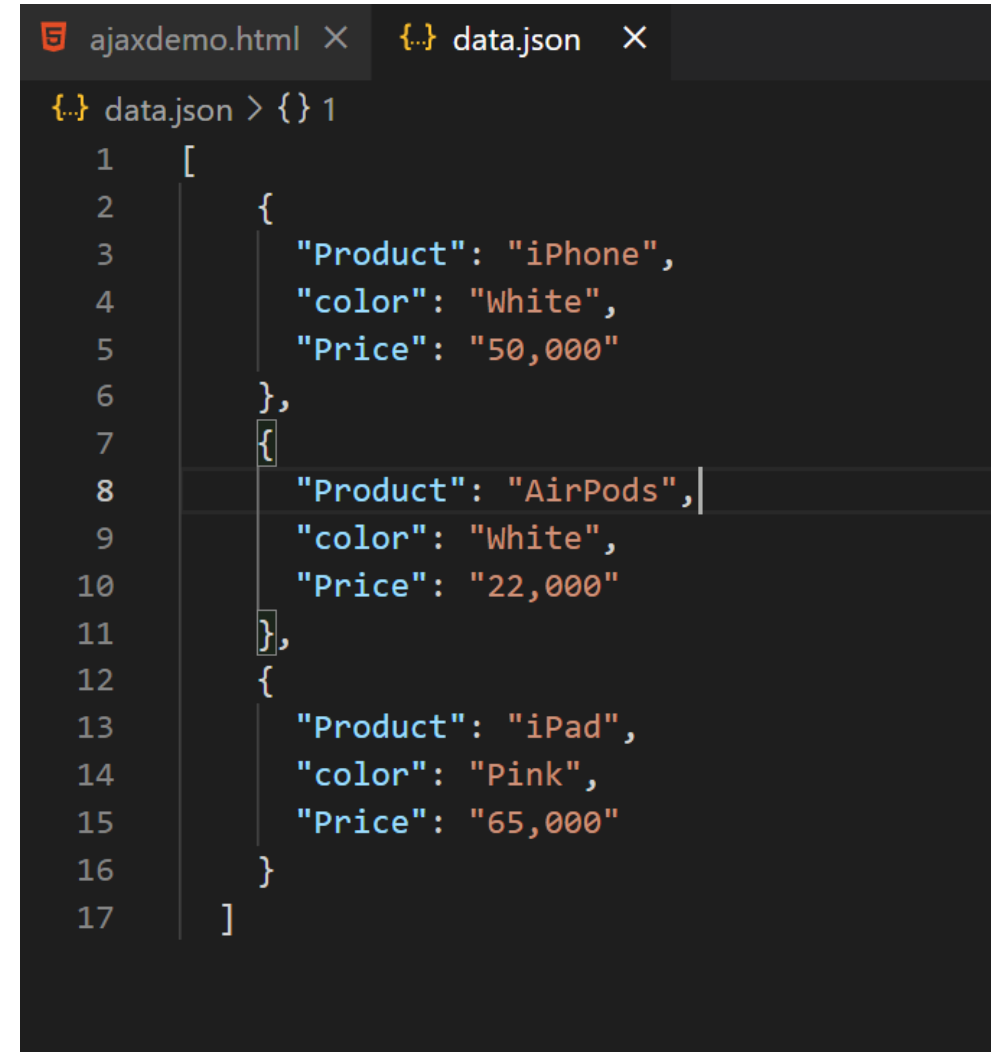


Fetch JSON from jQuery AJAX



Data.json

```
[
  {
    "Product": "iPhone",
    "color": "White",
    "Price": "50,000"
  },
  {
    "Product": "AirPods",
    "color": "White",
    "Price": "22,000"
  },
  {
    "Product": "iPad",
    "color": "Pink",
    "Price": "65,000"
  }
]
```

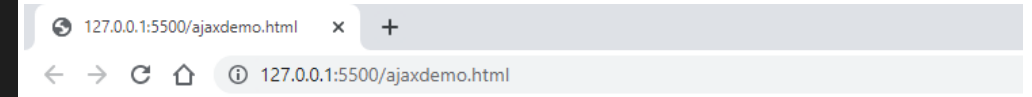


```
ajaxdemo.html X {..} data.json X
{..} data.json > {} 1
1  [
2      {
3          "Product": "iPhone",
4          "color": "White",
5          "Price": "50,000"
6      },
7      {
8          "Product": "AirPods",
9          "color": "White",
10         "Price": "22,000"
11     },
12     {
13         "Product": "iPad",
14         "color": "Pink",
15         "Price": "65,000"
16     }
17 ]
```



Load Data in Table

```
ajaxdemo.html > html > head > script > ready() callback
1 <html>
2 <head>
3   <script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.js"></script>
4   <script>
5     $(document).ready(function () {
6       $.ajax({
7         url: "data.json",
8         success: function (result, status, xhr) {
9           var table = $("<table><tr><th>Product</th><th>Color</th><th>Price</th></tr>");
10          var tr;
11          for (var i = 0; i < result.length; i++) {
12            tr = $("<tr>");
13            tr.append("<td>" + result[i].Product + "</td>");
14            tr.append("<td>" + result[i].color + "</td>");
15            tr.append("<td>" + result[i].Price + "</td>");
16            tr.append("</tr>");
17            table.append(tr);
18          }
19          table.append("</table>");
20          $("#myData").html(table);
21        }
22      });
23    });
24  </script>
25 </head>
26 <body>
27   <div id="myData"></div>
28 </body>
29 </html>
```



Product	Color	Price
iPhone	White	50,000
AirPods	White	22,000
iPad	Pink	65,000



Example

```
<html>
<head>
  <script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.js"></script>
  <script>
    $(document).ready(function () {
      $.ajax({
        url: "data.json",
        success: function (result, status, xhr) {
          var table = $("<table><tr><th>Product</th><th>Color</th><th>Price</th></tr>");
          var tr;
          for (var i = 0; i < result.length; i++) {
            tr = $("<tr>");
            tr.append("<td>" + result[i].Product + "</td>");
            tr.append("<td>" + result[i].color + "</td>");
            tr.append("<td>" + result[i].Price + "</td>");
            tr.append("</tr>");
            table.append(tr);
          }
          table.append("</table>");
          $("#myData").html(table);
        }
      });
    });
  </script>
</head>
<body>
  <div id="myData"></div>
</body>
</html>
```



LoadData UsingXML

```
<?xml version="1.0" encoding="utf-8" ?>
<products>
  <product>
    <name>Mini Skirt size 38</name>
    <color>Brown</color>
    <price>$40.77</price>
  </product>
  <product>
    <name>Women Pant size 39</name>
    <color>Black</color>
    <price>$21.45</price>
  </product>
  <product>
    <name>Men Coat size 47</name>
    <color>Pink</color>
    <price>$101.50</price>
  </product>
</products>
```



jQuery AJAX Error

- You can use the error parameter to call a function in case of receiving any **jQuery AJAX Error**. To understand it,
- let us fetch a non-existing file using **jQuery AJAX method** and show the error message inside a div.
-



Load Using Error

```
<html>
<head>
  <script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.js"></script>
  <script>
    $(document).ready(function () {
      $("#LoadData").click(function (e) {
        $.ajax({
          url: "no-file.txt",
          success: function (result, status, xhr) {
            $("#myData").html(result)
          },
          error: function (xhr, status, error) {
            $("#myData").html("Result: " + status + " " + error + " " + xhr.status + " " + xhr.statusText)
          }
        });
      });
    });
  </script>
</head>
<body>
  <button id="LoadData">Load</button>
  <div id="myData"></div>
</body>
</html>
```

Load

Result: error Not Found 404 Not Found

GET http://127.0.0.1:5500/no-file.txt 404 (Not Found)
XHR failed loading: GET "http://127.0.0.1:5500/no-file.txt".



.load()

- This method can be used to load data on a website element. Just indicates the URL of the file that we want and the method update the HTML of the element.
- This method doesn't accept any parameter and just makes the function of load, nothing else.
- `<script type="text/javascript"> $("#element").load("mypage.html"); </script>`
- Also we can send data by GET method, just indicates the data inside the url.
- `<script type="text/javascript"> $("#element").load("mypage.php?data=12qw"); </script>`



\$.get() and \$.post()

- Two brother methods that do the same task: an AJAX connection sending data to the server. The difference is that \$.get() send data via GET (in the url) and \$.post() send data via POST (in HTTP headers).



\$.getScript()

- This method loads a JavaScript file in a specific address

```
$.getScript("js/myscript.js");
```



\$.getJSON

- Here we specify the response type , in case of expect jsonp you must add 'callback=?' to the url.

```
<script type="text/javascript">
```

```
$.getJSON("https://jsonplaceholder.typicode.com/posts",
```

```
function(){
```

```
    alert("success");
```

```
});
```

```
</script>
```

```
<script type="text/javascript">
```

```
$.ajax({ url: "https://jsonplaceholder.typicode.com/posts",
```

```
type: 'GET',
```

```
dataType: "script",
```

```
success: function(){
```

```
    alert("Success");
```

```
}
```

```
});
```

```
</script>
```



Global Event Handlers

- jQuery has a set of global AJAX functions which you can use to listen for AJAX events across all AJAX requests sent via jQuery



\$.ajaxSend()

- The callback function registered with the ajaxSend() function is always called just before an AJAX request is sent via jQuery.

```
$(document).ajaxSend(function() {  
    console.log("called before each send");  
});
```



\$.ajaxStart()

- Whenever an Ajax request is about to be sent, jQuery checks whether there are any other outstanding Ajax requests. If none are in progress, jQuery triggers the ajaxStart event.
- If \$.ajax() or \$.ajaxSetup() is called with the global option set to false, the ajaxStart() method will not fire.

```
$( document ).ajaxStart(function() {  
    $( "#loading" ).show();  
});
```



\$.ajaxStop()

- Whenever an Ajax request completes, jQuery checks whether there are any other outstanding Ajax requests. If none remain, jQuery triggers the ajaxStop event.
- If \$.ajax() or \$.ajaxSetup() is called with the global option set to false, the .ajaxStop() method will not fire.

```
$( document ).ajaxStop(function() {  
    $( "#loading" ).hide();  
});
```



\$.ajaxSuccess()

- Whenever an Ajax request completes successfully, jQuery triggers the ajaxSuccess event.

```
$( document ).ajaxSuccess(function( event, xhr,  
settings ) {  
    $( "#msg" ).append( "<li>Successful Request!</li>"  
);  
});
```



\$.ajaxError()

- Whenever an Ajax request completes with an error, jQuery triggers the ajaxError event.

```
$( document ).ajaxError(function( event, xhr, settings ) {  
    $( "#msg" ).append( "<li>Failed Request!</li>" );  
});
```



\$.ajaxComplete()

- Whenever an Ajax request completes, jQuery triggers the ajaxComplete event.

```
$( document ).ajaxComplete(function( event, xhr, settings ) {  
    $( "#msg" ).append( "<li>Request Completed !!</li>" );  
});
```



Task

- Perform all Task Given in PPT
- Load JSON Api and Print Data in HTML Table
 - <https://jsonplaceholder.typicode.com/posts>
- Take 2 Dropdown
 - State (Feed 2-3 State in HTML Data)
 - City (On Basis of State Selection Load City Name)
- Create 2-3 Ajax Programs with Proper Coding Idea.
- Ref :
 - <https://www.webslesson.info/2017/05/json-dynamic-dependent-select-box-using-jquery-and-ajax.html>

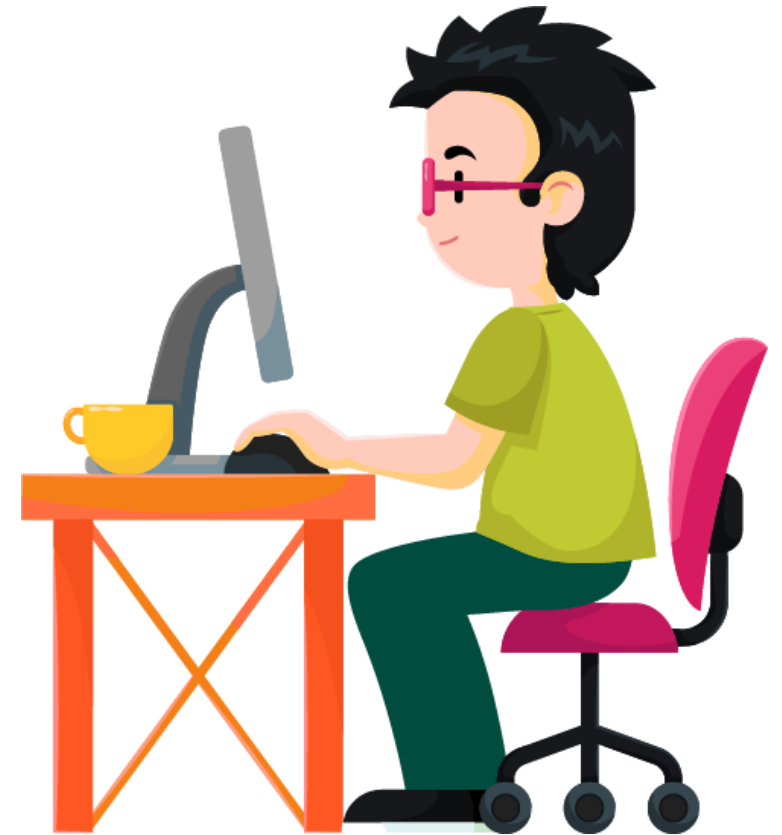


Get Exclusive Video Tutorials



www.apptutorials.com

<https://www.youtube.com/user/Akashtips>





Get More Details

www.akashsir.com



If You Liked It !

Rating Us Now



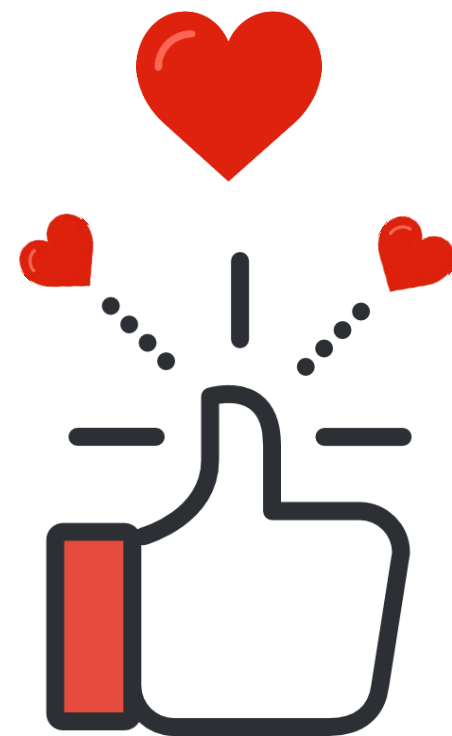
Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET



Sulekha

<https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad>



Connect With Me



Akash Padhiyar
#AkashSir

www.akashsir.com

www.akashtechlabs.com

www.akashpadhiyar.com

www.apptutorials.com

Social Info



Akash.padhiyar



Akashpadhiyar



Akash_padhiyar



+91 99786-21654



#Akashpadhiyar
#apptutorials