**What is same origin policy**  
Browsers allow a web page to make AJAX requests only with in the same domain. Browser security prevents a web page from making AJAX requests to another domain. This is called same origin policy.   
  
  
  
**The following 2 URLs have the same origin**  
http://localhost:1234/api/employees  
http://localhost:1234/Employees.html  
  
**The following 2 URLs have different origins, because they have different port numbers (1234 v/s 5678)**  
http://localhost:1234/api/employees  
http://localhost:5678/Employees.html  
  
**The following 2 URLs have different origins, because they have different domains (.com v/s .net)**  
http://pragimtech.com/api/employees  
http://pragimtech.net/Employees.html  
  
**The following 2 URLs have different origins, because they have different schemes (http v/s https)**  
https://pragimtech.com/api/employees  
http://pragimtech.net/Employees.html  
  
To prove browsers does not allow cross domain ajax requests, let's add a new web forms project to EmployeeService solution. Name it ClientApplication. Add an HTML page. Name it HtmlPage1.html. Copy and paste the following HTML and jQuery code.

<!DOCTYPE html>

<html>

<head>

    <title></title>

    <meta charset="utf-8" />

    <script src="Scripts/jquery-1.10.2.js"></script>

    <script type="text/javascript">

        $(document).ready(function () {

            var ulEmployees = $('#ulEmployees');

            $('#btn').click(function () {

                $.ajax({

                    type: 'GET',

                    // Make sure to change the port number to

                    // where you have the employee service

                    // running on your local machine

                    url: 'http://localhost:23258/api/Employees',

                    dataType: 'json',

                    success: function (data) {

                        ulEmployees.empty();

                        $.each(data, function (index, val) {

                            var fullName = val.FirstName + ' ' + val.LastName;

                            ulEmployees.append('<li>' + fullName + '</li>')

                        });

                    }

                });

            });

            $('#btnClear').click(function () {

                ulEmployees.empty();

            });

        });

    </script>

</head>

<body>

    <input id="btn" type="button" value="Get All Employees" />

    <input id="btnClear" type="button" value="Clear" />

    <ul id="ulEmployees"></ul>

</body>

</html>

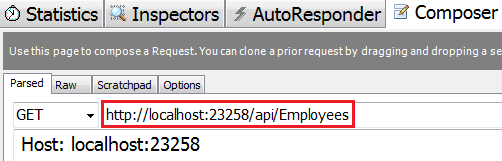
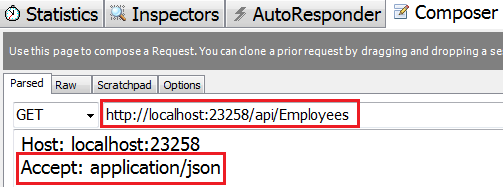
When you click **"Get All Employees"** button on **"HtmlPage1.html"** page, you get the following error. To see the error launch browser tools and click on the console tab.  
XMLHttpRequest cannot load http://localhost:23258/api/Employees. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://localhost:6293' is therefore not allowed access.  
  
On the other hand when you click **"Get All Employees"** button on **"Employees.html"**page that is present in the same project as the ASP.NET Web API Service, that employee data is displayed without any problem.  
  
So this proves, browsers does not allow cross domain ajax requests. There are 2 ways to get around this problem

* Using JSONP (JSON with Padding)
* Enabling CORS (Cross Origin Resource Sharing)

In this video let's use JSONP to overcome the browser cross-domain restriction. In our next video we will discuss enabling CORS.   
  
**So what is JSONP and what does it do?**  
JSONP stands for JSON with Padding. All JSONP does is wraps the data in a function. So for example, if you have the following JSON object  
{  
    "FirstName" : "Mark",  
    "LastName"  : "Hastings",  
    "Gender"    : "Male",  
}  
  
**JSONP will wrap the data in a function as shown below**  
CallbackFunction({  
    "FirstName" : "Mark",  
    "LastName"  : "Hastings",  
    "Gender"    : "Male",  
})  
  
Browsers allow to consume JavaScript that is present in a different domain but not data. Since the data is wrapped in a JavaScript function, this can be consumed by a web page that is present in a different domain.  
  
Steps to make ASP.NET Web API Service to return JSONP formatted data and consume it from a cross domain ajax request  
**Step 1 :** To support JSONP format, execute the following command using NuGet Package Manager Console which installs WebApiContrib.Formatting.Jsonp package.  
Install-Package WebApiContrib.Formatting.Jsonp  
  
**Step 2 :** Include the following 2 lines of code in Register() method of WebApiConfig class in WebApiConfig.cs file in App\_Start folder

var jsonpFormatter = newJsonpMediaTypeFormatter(config.Formatters.JsonFormatter);

config.Formatters.Insert(0, jsonpFormatter);

**Step 3 :** In the ClientApplication, set the dataType option of the jQuery ajax function to jsonp   
dataType: 'jsonp'  
  
**Testing the ASP.NET Web API Service using fiddler**  
Notice in fiddler we have just specified the URL of the Web API service without Accept header and callback function in the URI   
   
  
The above request results in the following error  
A callback parameter was not provided in the request URI  
  
If you want JSON data back, set **Accept** header to **application/json** and there is no need to specify the callback function in the URI. The request completes successfully.   
   
  
If you want JSONP formatted data back, set **Accept** header to **application/javascript**and specify a name for the callback function in the URI. We have set it to **ABC**.   
