In this video we will discuss implementing logout functionality for ASP.NET Web API. This is continuation to [Part 25](http://csharp-video-tutorials.blogspot.com/2016/12/web-api-bearer-token-example.html). Please watch [Part 25](http://csharp-video-tutorials.blogspot.com/2016/12/web-api-bearer-token-example.html) from [ASP.NET Web API tutorial](https://www.youtube.com/playlist?list=PL6n9fhu94yhW7yoUOGNOfHurUE6bpOO2b)before proceeding.   
  
To log out the user from the application all we have to do is remove the Access token from the client browser session storage. Here is what we want to do.  
1. Include a **Log Off** button on the Data.html page  
2. When the **Log Off** button is clicked remove the access token from client browser session storage and redirect the user to the login page.   
   
  
  
  
HTML for the **Log Off** button. Include the following HTML on the Data.html page immediately below the Load Employees button.

<input id="btnLogoff" class="btn btn-success pull-right"

       type="button" value="Log Off" />

In the script section include the following jQuery click event handler for the Log Off button as shown below.

$('#btnLogoff').click(function () {

    sessionStorage.removeItem('accessToken');

    window.location.href = "Login.html";

});

**There are 2 ways for the user to Log Off**  
1. By closing the browser window. Since we are storing the access token in browser session storage, the access token will be lost when we close the browser window.   
2. By clicking the **"Log Off"** button, which explicitly removes the access token from the browser session storage.  
  
If you do not want to loose the access token, when the browser is closed store the access token in browser local storage instead of session storage. The way you store, retrieve and remove items from local storage is exactly the same as storing, retrieving and removing items from session storage, except that you use localStorage object instead of sessionStorage object.  
  
At this point, you may have the following questions.  
  
We are only deleting the access token on the client. We are not invalidating or deleting the access token from the server side. If someone can intercept the access token, will they not be able to use that access token and gain access to the system.  
The straight answer to the question is YES. If someone is able to intercept the access token, they will be able to impersonate and gain access to the system. However, most of the systems that use access tokens, work over SSL (Secure Socket Layer), which inhibits intercepting access tokens.  
  
Should we invalidate or delete access tokens from the server  
No, there is no need to invalidate or delete access tokens from the server. Access token lives on the client, and it is enough if we remove it from the client. Another good practise is to set the expiry of the access token to as short time as practically possible depending on the nature of your application.