# Autocomplete Tutorial

By: Chad Clites

For: extant.digital

Date: 11/08/215

Email: chad@extant.digital

## Real World Use Case

As a user, when I enter some text in a field, I want to be presented with a list of suggestions based on the text entered. I want the list to be selectable, so that when I select an option from the list, it auto-populates the input field for me.

## Steps:

1. Create HTML
2. Create CSS
3. Build the JavaScript

## Introduction:

Technically this is not an 'autocomplete' feature, but rather a 'suggestions' feature, which is a modern component of any modern search engine on and modern web page. The user enters some text in an input element, an AJAX call is used to query the server for matching text. In this example, we will forgo the AJAX call and just use a hardcoded model instead.

There are multiple ways to accomplish the task at hand. This is simply one approach. If you are not familiar with JQuery or JavaScript in general, this may be a bit challenging. The expectation is that one has some experience with both.

## Create HTML:

We start out with a simple HTML5 skeleton.

```
<!DOCTYPE html>
<html>
   <head>
       <title>Basic Template</title>
       <style></style>
       <script></script>
   </head>
   <body>
      <input id="myinput" value="" />
      <div id="myselection"></div>
      <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
   </body>
</html>
```

Included are placeholders for our styles and JavaScript, an input field, and a container to hold the list of suggestions.

## Create CSS:

The CSS is nothing special.

```
<style>
   #myselection{
      display: none;
      width: 100px;
      height: auto;
   }

   #myselection div{
      cursor: pointer;
    }
</style>
```

We want to hide the selection element initially. Other than that, nothing special here.

## Build The JavaScript:

The JavaScript as a whole is a bit complex, but if we break it down into small parts, it isn't all that bad. It is worth looking at the each of the steps.

- Wait for the page to load.
- Once the page loads, add a listener to the input field.
- When text is entered, search through the menu item list.
- If the list has matches, add them to 'myselection'.
- Add listeners to each of the elements.
- When selection is clicked, fill in the input field

The first thing we want to do is create a JavaScript array that contains available suggestions.

```
<script>

    //An array of strings to test our dropdown.
    var menuItems = ["Alpha", "Bravo", "Charlie", "Delta", "Echo", "Foxtrot", "Golf",
                     "Hotel", "India", "Juliet", "Kilo", "Lima", "Mike", "November",
                     "Oscar", "Papa", "Quebec", "Romeo", "Sierra", "Tango",
                     "Uniform", "Victor", "Whiskey", "Xray", "Yankee", "Zulu"];

</script>
```

In our <script> block, we want to add an onready function to do some things once the page is loaded. In this instance, the 'some things' is to attach a keyup function to the input field that is triggered when we enter text.

```
$(function(){
    $("#myinput").keyup(function(){
            getMatches();
    });
});
```

Now we add the getMatches function.

```
function getMatches(){

            var searchStr = $("#myinput").val(),  //get input
                minLength = 0, //Set a minimum length for the search string
                tempArray = [];

            if( searchStr.length > minLength){

                for(var i =0; i<menuItems.length; i += 1){
                        var str = menuItems[i];

                        if( str.indexOf(searchStr) > -1 ){
                            //Found a match. Add it to new temporary array
                            tempArray.push(str);
                    }
                }

                if(tempArray.length > 0){

                    //re-populate the list
                   $("#myselection"). html(tempArray);
                   $("#myselection").show();

                   //add a click listener to the new menu items
                   $("#myselection div").click(function(){

                          //when a selection is clicked, we want to add its value
                          //to the input field.
                          $("#myinput").val( $(this).text() );

                         //now hide the selection window
                         $("#myselection").hide();

                    });
            }

        }else{
            $("#myselection").hide();
        }
}
```

```
function appendEntry(tempArray){

   var html = "";
   for (var i=0; i< tempArray.length; i +=1 ){

      html += "    <div class='ir_gbl_entry'>\n" +
                 "        <span class='ir_gbl_label'>" + gbls[i] + "</span>\n" +
                 "    </div>\n";
    }

    return html;
}
```

There are a lot of things going on in here. The first thing is to set a couple of fields. The minLength field is used to determine when to trigger the search. In this example, the minLength is set to zero, but in the real world, one would probably not want to trigger the search until 2 or 3 letters have been entered in the input field.

If the minLength threshold has been met, then we loop through each of the entries in the menuItems array and see if our text is contained anywhere in any of the menuItems. If it is, add that item to the tempArray.

The tempArray is then fed to the appendEntry function, which in turn generates the html necessary to populate the dropdown.

Our function also adds a click listener to the suggestion list. When one of the entries in the list are selected, the value is transferred to the input field, and the list is removed from the view.

## Conclusion:

Full disclosure. I adjusted some of the code on the fly and didn't test it, so although this worked previously, my additions may have broken the code.