ADVANCED JAVASCRIPT FOR WEB SITES AND WEB APPLICATIONS

SESSION 7 B - AUTOSUGGEST PLUGIN

AUTOSUGGESTION

Let's build an autocomplete plugin to work on a simple search form.

We'll use Google's suggestion api, that can be found here:

http://suggestqueries.google.com/complete/search?client=firefox&q=keyword

This API returns a text file, with a list of keywords that match the query (the "q" parameter).

The results are in an array with two items:

1. the search term
2. the list of matching results, in another array

The application needs to:

1. make an ajax call when the user presses a key, when inside the input field
2. process the results from the ajax call, and print them on the page in a list item
3. when a user selects one of the suggestion, close the suggestion box and add the term to the input field
4. if the user presses escape on their keyboard, while inside the input field, close the suggestion box
5. if input field loses focus, close the suggestion box

1. Start the application

Use the revealing module pattern, with an `init` function. We'll use that `init` function to declare our event handlers.

```
var AUTOSUGGEST = (function () {

  var

    init = function () {
    };

  return {
    init: init
  };

}());

AUTOSUGGEST.init();
```

Select the form and the input element on the page, using the global `var` declaration in the application.

```
searchInput = document.getElementById("input-search"),
form = document.getElementById("form"),
```

2. Make the Ajax call

Using the jQuery Ajax method, create a function to make a call to the Google API on keyup, on the input field.

This function will run everytime a keyup event is triggered in the input field. We'll need to declare an event handler inside the `init` function for this.

Reminder on the Ajax object in jQuery:

```
$.ajax({
  url: url,
  dataType: "jsonp",
  type: "GET",
  success: function (data) {
  }
});
```

For the moment, make the call and leave the success function empty.

```
searchCall = function (event) {

  var

    response,
    html = [],
    searchInputValue = searchInput.value;

    $.ajax({
      url: "http://suggestqueries.google.com/complete/search?client=firef
      dataType: "jsonp",
      type: "GET",
      success: function (data) {
        console.log(data);
      }
    });
  },
```

3. the success function

In the success function, grab the array of suggestion, if this array is not empty, create a `ul` element with a class of `suggestions` and the list of suggestions as `li`'s. And finally insert the `ul` in the DOM, right after the input element.

```
success: function (data) {
  response = data[1];
  if(response.length > 0) {
    html.push("<ul class=\"suggestions\">");
    for (var i = 0; i < response.length; i++) {
      html.push("<li>" + response[i] + "<\/li>");
    }
    html.push("≷/ul>");
    removeElement(".suggestions");
    searchInput.insertAdjacentHTML("afterend", html.join("\n"));
  }
}
```

Make sure you declare the event handler for the `searchCall` function on keyup, on the input element!

```
init = function () {
  searchInput.addEventListener("keyup", searchCall);
};
```

We need to add one more thing to the previous function, make sure we remove the suggestion list before adding a new one.

For this, create a function that will take one argument, a css selector, and remove this element from the DOM, if it exists.

```
removeElement = function (cssSelector) {
  var el = document.querySelectorAll(cssSelector);
  if (el.length > 0) {
    el[0].parentNode.removeChild(el[0]);
  }
},
```

Inside the success function:

```
html.push("");
removeElement(".suggestions");
searchInput.insertAdjacentHTML("afterend", html.join("\n"));
```

4. Selecting a suggestion

When the user selects one of the suggestion, add the term to the input box and close the dropdown list.

Create a new function for this, and attach it to click event handler for one of the li items in the list of suggestion.

Since we don't want to keep adidng event handlers to the li items (they are killed and created on the fly and we're never sure how many we'll have), attach the handler to the form element, and use the `event.target` property to detect if the user has clicked on an `li` item or not.

```
selectSuggestion = function (event) {
  if (event.target.nodeName === "LI") {
    searchInput.value = event.target.textContent;
    removeElement(".suggestions");
  }
},
```

And declare the event handler for a click event on the list of suggestions!

```
form.addEventListener("click", selectSuggestion);
```

5. Improve the Ajax call slightly

We can add two things to make the application slightly more usable:

- before mnaking the Ajax call, check which key was pressed (with the `keyCode` property of the event object), if it is the escape key, don't make a call, but close the list of suggestions.
- before making the Ajax call, make sure the value of the input is not empty, only make a call if it isn't.

```
searchCall = function (event) {
  var
    response,
    html = [],
    searchInputValue = searchInput.value;

    if (event.keyCode === 27) {
      removeElement(".suggestions");
    }
    else {
        if (searchInputValue.length > 0) {

        $.ajax({
          url: "http://suggestqueries.google.com/complete/search?client=f
          dataType: "jsonp",
          type: "GET",
          success: function (data) {
          response = data[1];
          if(response.length > 0) {
            html.push("<ul class=\"suggestions\">");
            for (var i = 0; i < response.length; i++) {
              html.push("<li>" + response[i] + "<\/li>");
            }
            html.push("</ul>");
            removeElement(".suggestions");
            searchInput.insertAdjacentHTML("afterend", html.join("\n"));
          }
        }
```

```
            });
        }
    }
},
```

6. On blur

At the moment, the list of suggestions is still visible when the input loses focus. Make sure that the list of suggestions is removed when the input field loses focus.

The event to use for this is `blur`.

The function to attach to this event handler will need to remove the list of suggestion (using our previous `remove` function), only if the event object's `explicitOriginalTarget` has a granparent with a class of "suggestions".

```
inputOut = function (event) {
if (!event.explicitOriginalTarget.parentNode.parentNode.classList.contain
  removeElement(".suggestions");
};
```

In the `init` function:

```
searchInput.addEventListener("blur", inputOut);
```