

# Report

## How did I do?

---

1. I used `<list>`, `<span>` and `<input>` to finish this test.

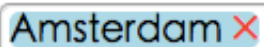


A screenshot of a web form. At the top is a button with the Chinese text "提交" (Submit). Below it is a text input field. The input field contains the text "Harbin" followed by a red "X" icon and the placeholder text "Please type city name".

When type in a letter, `onkeyup` event triggers `query()` handler to query for autocompletion. Each confirmed input will be inserted into a span as an entry of list.

```
<ul>
  <li>
    <span>Harbin</span>
  </li>
</ul>
```

After intertion complete, a new span for next insertion will be created. Each inserted tag has a clickable multiply sign. It allows you remove a tag by clicking the sign.



A screenshot of a text input field. The input field contains the text "Amsterdam" followed by a red "X" icon.

2. Autocomplete was done by calling a function `grep` with a callback function to do pattern matching.

```
set = grep(pattern, function(city, pattern){
  return pattern.test(city);
});
```

What `grep` does is taking each item in list `timezone` for callback. The reason I used callback is that if the number of matching items is large, or if matching remotely, callback can avoid blocking if network is blocked.

A web form with a submit button labeled "提交" (Submit) and a text input field containing "Ha". Below the input field is a dropdown menu with two options: "Harare" (highlighted in yellow) and "Harbin".

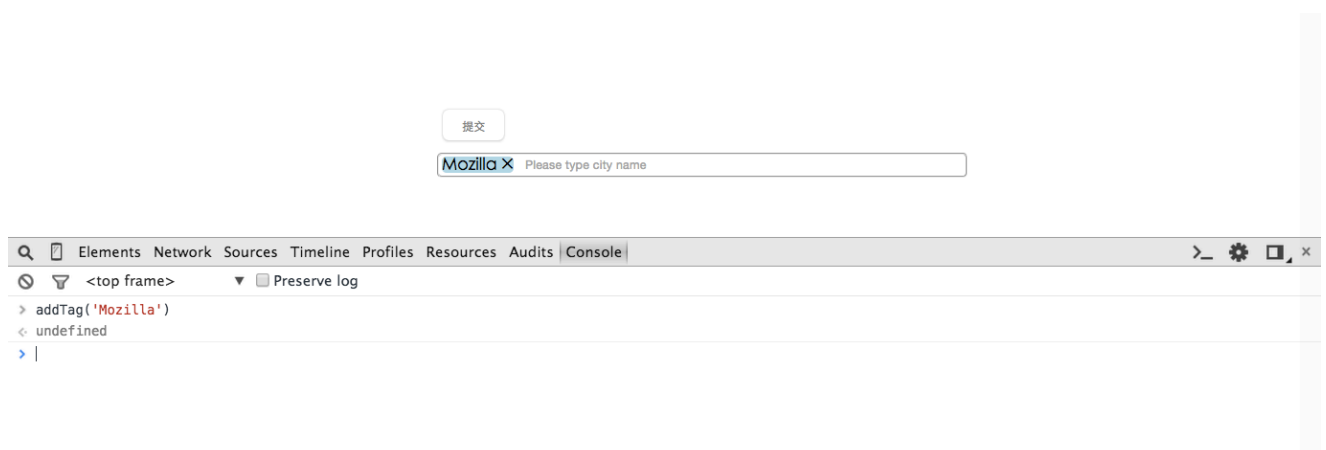
- After remove a tag, we should check if the instertion list still contains item conforming in timezone. If is, keep submit button enable, otherwise, disable submit button. I reached this goal by checking a global variable `tz_tags` instead of traversing all list again. `tz_tags` will be accumulated each time when a tag conforming in timezone list was added, and will be decelerated each removal.

A diagram showing the state of a web form before and after a tag removal. On the left, the form has a submit button labeled "提交" (Submit) and a text input field with the placeholder "Please type city name". On the right, the form has a submit button labeled "提交" (Submit) and a text input field containing "Amsterdam" with a close button "X". An arrow "====>" points from the left state to the right state.

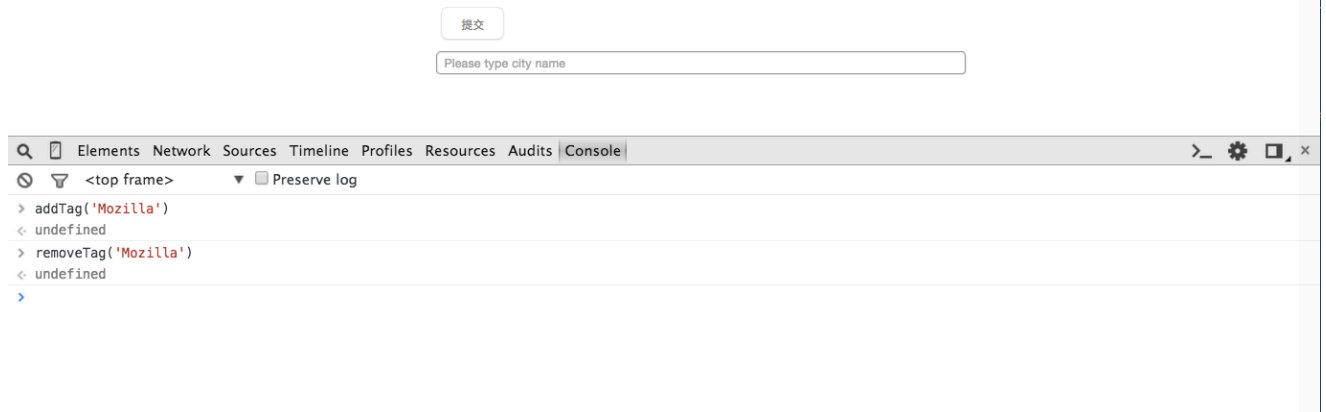
## Unit-testable and reusable

We can use web-based test framework, like karma, to do unit test for each function. The following are test by console in web browser.

- `addTag()`



- `removeTag()`



- `grep()`

```
> var pattern = new RegExp("^A", "i");  
  grep(pattern, function(city, pattern){  
    return pattern.test(city);  
  });  
< ["Amsterdam", "Andorra", "Athens", "Adelaide", "Abidjan", "Accra", "Addis_Ababa", "Algiers", "Asmara", "Aden", "Almaty", "Amman", "Anadyr", "Aqtau", "Aqtobe",  
  "Ashgabat", "Ashkhabad"]  
>
```

I separated pattern matching, add and remove tag into individual function. So that we can test each independently.

Because `query()` function need to distinguish keycode of input, I didn't try it in browser console, but this work can be done by using automatic web test tool.