



# **COMP 4021**

# **Internet Computing**

## **More on Autocomplete**

Dr. Kenneth LEUNG

Slides created by Prof. Dik Lun LEE

# “select(event, ui)” Event

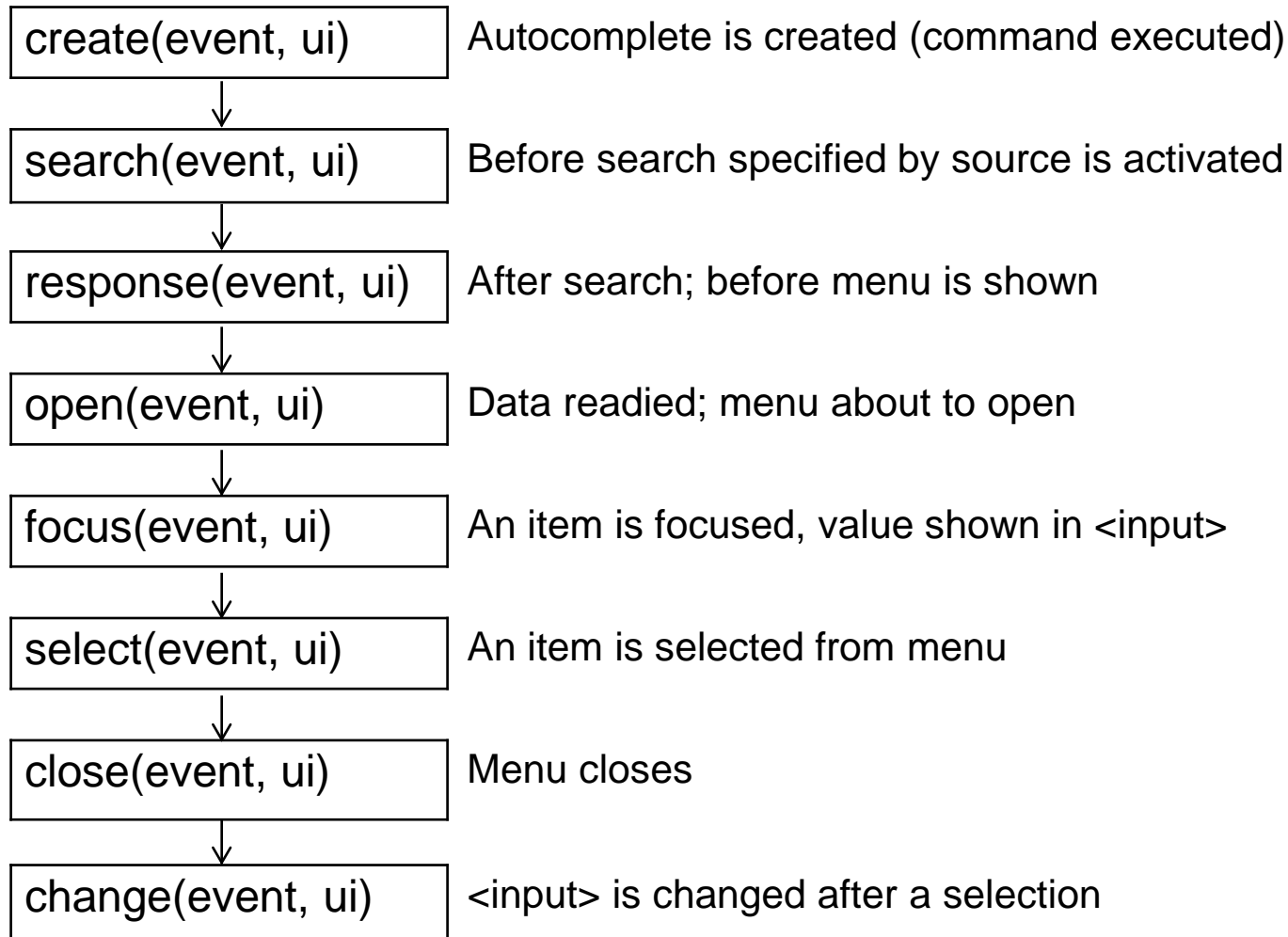
```
var items = [ { label: "Bianche Classic", value: " Bianche Classical  
Bicycle",  
                price: "HKD 1000" }, ... ]  
$( "#searchBox" ).autocomplete( {  
    source: items;  
    select: function( event, ui ) {  
        var itemDesc = ui.item.value + ui.item.price;  
        $("#searchBox").val(itemDesc);  
        return false;     }    } })
```

- function( event, ui ) defines the select event handler, which is executed when an item in menu is selected
- Note: In addition to label and value, we add other fields too
- focus( event, ui ) is triggered when mouse hovers over an item in menu but before clicking

# Autocomplete() Options For Reference Only

appendTo	Append an element to the menu
delay	Delay (msec) before updating suggestions
minLength	Number of characters entered before updating suggestions
position	Position of menu relative to associated input element.
source	Methods to obtain suggestions from input and dictionary

# Event Methods and Flow



# “\_renderItem(ul, item)” Event

```
var items = [ { label: "Bianche Classic", value: " Bianche Classical Bicycle",  
               price: "HKD 1000" }, ... ]  
$( "#searchBox" ).autocomplete({  
  source: items,  
}).data("ui-autocomplete")._renderItem = function( ul, item ) {  
  return $( "<li>" ).append( "<a>" + item.label + "<br>" + item.desc + "</a>" )  
    .appendTo( ul );  
};
```

- Customize how items are displayed in menu window
- Menu window is a `<ul><li>...</li> ...<li>...</li></ul>` element
- The code selects all `<li>` and appends a string to each item and finally append them to `<ul>`

## Other Extensions For reference only

<code>_renderItem( ul, item )</code>	Creation of each item in the menu. Creates a new <code>&lt;li&gt;</code> element, appends it to the menu and return it
<code>_renderMenu( ul, items )</code>	Building the menu
<code>_resizeMenu()</code>	Size of the menu before it is displayed. <code>this.menu.element</code> contains the Menu elements

# Source Data from Server (1)

- So far we assumed “source” is prepared on client side :
  - Source is an array variable preloaded with data, or
  - Computed with **source: function ( request, response)**
- Now learn about fetching data from server; needed when:
  - Source data is compiled in real time (e.g., most frequent queries in the last 5 minutes)

```
$( "#searchBox" ).autocomplete({  
    source: "language.js"  
});
```

- The source parameter is a URL; "language.js" points to the data file stored in the same directory as the accessing page
- language.js contains =>

```
[  
    { "label" : "java", "value" : "JAVA" },  
    { "label" : "javascript", "value" : "JAVASCRIPT" },  
]
```

# Source Data from Server (2)

- Autocomplete **does not do filtering**; all items from server will be displayed as suggestions
  - Server must produce the right data in language.js according to application requirement
- The data must be in **JavaScript array format** to be acceptable by the "source" parameter
  - The format is called JSON (JavaScript Object Notation)



# JSON: JavaScript Object Notation

- JSON is a standard for exchanging objects between applications (e.g., client programs and server programs)
- It has the same format as JavaScript array (thus its name)
  - We have already seen JSON many times in previous lectures
- Objects are binary inside JavaScript engine; JSON is a string representation of objects so that objects can be exchanged between applications as strings using web protocols

Name/Value  
(Object of  
1 field)

```
{ "Id" : "0123" }
```

Object  
(example  
has 2  
fields)

```
{  
  "Id " : "0123" ,  
  "Name" : "Lee"  
}
```

Array of  
objects  
(example  
has 3  
objects)

```
[  
  { "Id" : "0123" },  
  { "Id" : "1123" },  
  { "Id" : "2123" }  
]
```

# JSON Cont.

JSON allows nesting of objects and arrays

Array of objects (example object has 2 fields each)	<pre>[   { "Id" : "0123", "Name" : "Lee", },   { "Id" : "1123", "Name" : "Chan" } ]</pre>
Object values	<pre>{ "Id " : "0123" ,   "Name" : { "fname" : "Dik", "lname" : "Lee" } }</pre>

# Server Program to Generate Data

- Autocomplete "source" parameter invokes a server-side program (e.g., PHP) and passes request.item to it
- Server program uses request.item to get matching suggestions and passes the filtered data back to the client as json data

```
$(function() {  
    $( "#searchBox" ).autocomplete( {  
        source: "search.php",  
        minLength: 1  
    });  
});
```

# Server-Side Program Code

```
<?
$term = $_GET[ "term" ];
$languages = array(
    array( "label" => "java", "value" => "Good for applications" ),
    array( "label" => "javascript", "value" => "Good for browsers" ),
);
$result = array();
foreach ($languages as $language) {
    $languageLabel = $language[ "label" ];
    if ( strpos( strtoupper($languageLabel), strtoupper($term) ) !== false ) {
        array_push( $result, $language );
    }
}
echo json_encode( $result );
?>
```

- We have not covered PHP but you can read it and imagine what can be done
- Suggestions are obtained by the program
- Other languages besides PHP can be used