COMP 4021
Internet Computing

More on Autocomplete

Dik Lun Lee

“select(event, ui)” Event

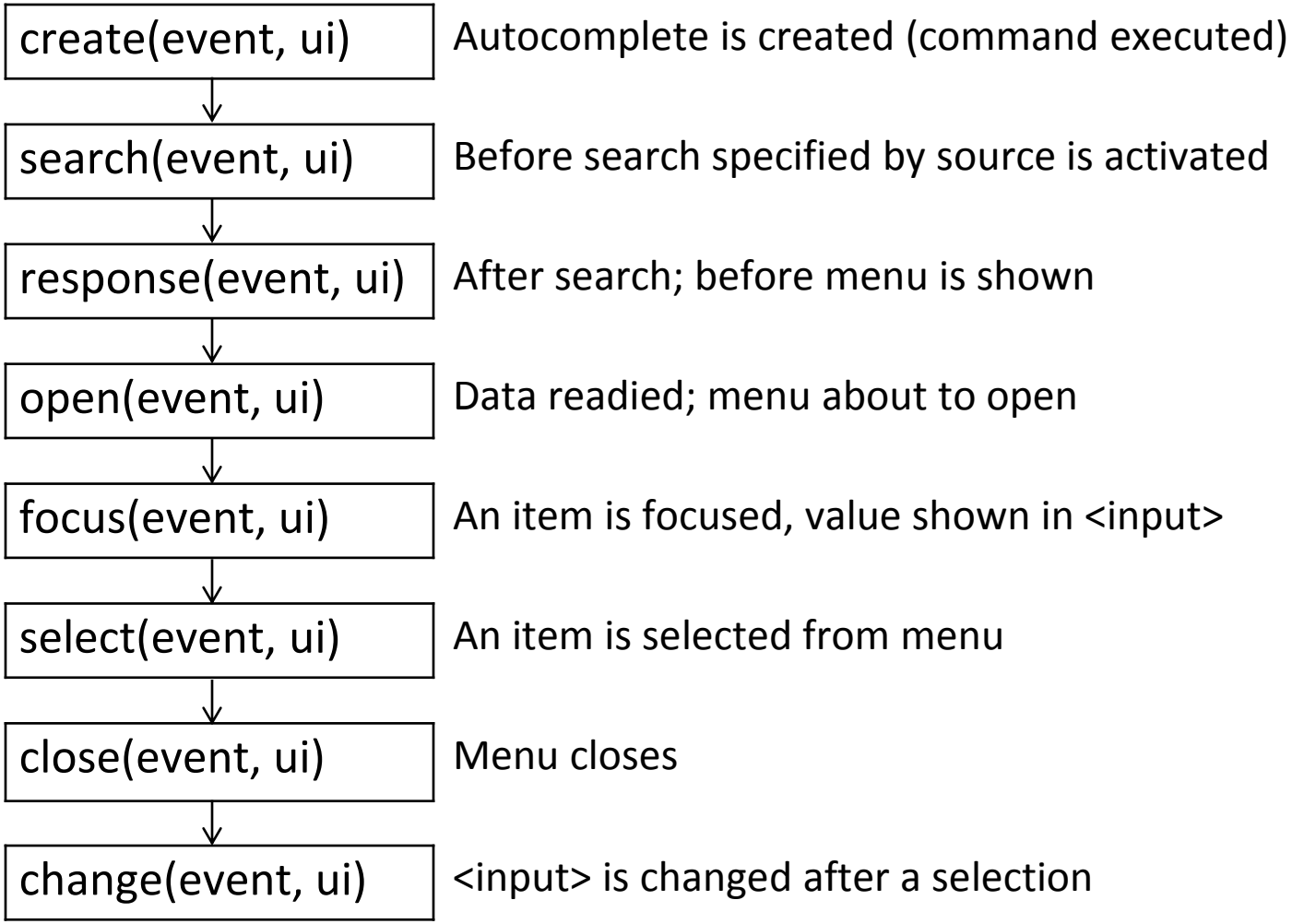
```
var items = [ { label: "Bianche Classic", value: " Bianche Classical Bicycle",
                        price: "HKD 1000" }, … ]
$( "#searchBox" ).autocomplete( {
    source: items;
    select: function( event, ui ) {
        var itemDesc = ui.item.value  + ui.item.price;
        $("#searchBox").val(itemDesc);
        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 … …
… element
- The code selects all <li> and appends a string to each item and finally append them to

Other Extensions For reference only

_renderItem(ul, item)	Creation of each item in the menu. Creates a new element, appends it to the menu and return it
_renderMenu(ul, items)	Building the menu
_resizeMenu()	Size of the menu before it is displayed. this.menu.element 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)	`[`
	`  { "Id" : "0123", "Name" : "Lee", },`
	`  { "Id" : "1123", "Name" : "Chan" }`
	`]`

Object values	`{ "Id " : "0123" ,`
	`  "Name" : { "fname" : "Dik", "lname" : "Lee" }`
	`}`

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

The Rest of Autocomplete

- Try out examples in course homepage
- Search and read the numerous tutorials on web

COMP 4021
Internet Computing

More on CSS Selectors

Dik Lun Lee

More Selectors: Class

```
<p class="important"></p>

<p></p>

<p class="warning"></p>

<p class="important warning"></p>
```

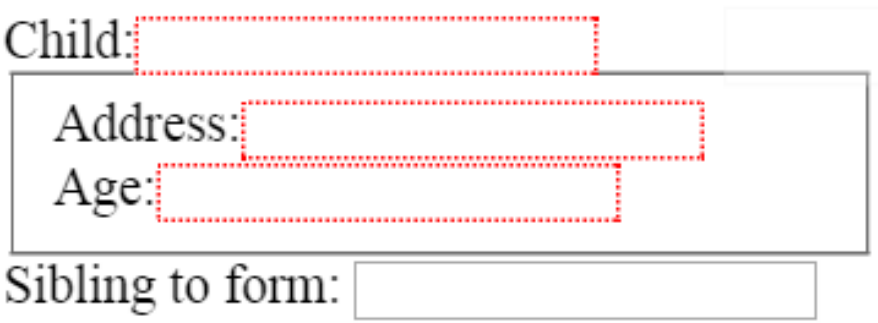
<p> belongs to two classes: «important» and «warning»

$('p.important, p.warning') returns which <p> tags?

$("ancestor descendant")

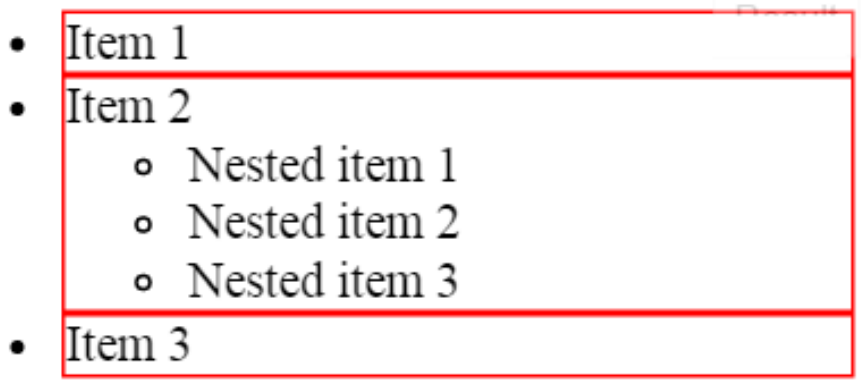
```
<form>
    <label>Child:</label><input name="name" />
    <fieldset>
        <label>Address:</label><input name="address" /><br/>
        <label>Age:</label><input name="age" /></fieldset></form>
Sibling to form: <input name="none" />

<script>$("form input").css("border", "2px dotted red");</script>
```

<input> tag with ancestor <form>



Child Selector: >

```
<ul class="topnav">
    <li>Item 1</li>
    <li>Item 2
        <ul>
            <li>Nested item 1</li>
            <li>Nested item 2</li>
            <li>Nested item 3</li>
        </ul></li>
    <li>Item 3</li>
</ul>
```



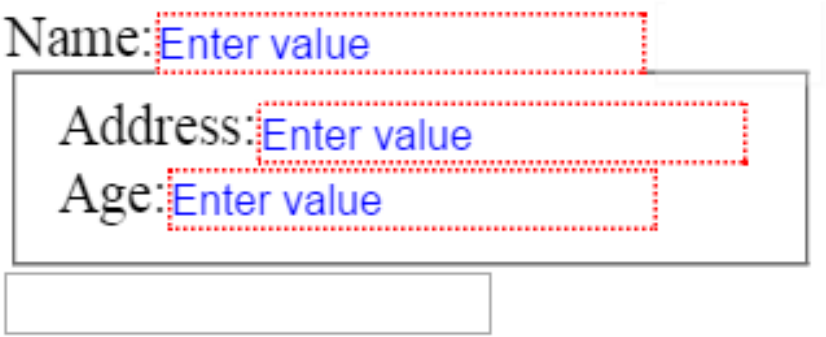
 tag with parent <ul> which has class “topnav”

```
<script>$("ul.topnav > li").css("border", “1px double red");</script>
```

Immediate Sibling Selector: +

<input> immediately preceded by <label>

```
<script>$("label + input").css({"border":"2px dotted red",
    "color":"blue"}).val("Enter value"); </script>

<form>
    <label>Name:</label><input name="name" />
    <fieldset>
        <label>Address:</label><input name="address" /><br/>
        <label>Age:</label><input name="age" /></fieldset></form>
<input name="none" />
```



What about:
<script>$("form input").css("color", "blue").val("Enter value")</script>

Next Sibling Selector: ~

$("**label ~ input**")

- <label> does not have to be immediately preceding <input>
- But sibling means <label> and <input> belong to the same parent

Combining Several Selectors

- **$("div > ul a")** reads:
  - All <a> elements which are descendants of <ul> elements which are direct children of elements

```
<div>
 <ul><li> This is a sublist:
        <ul>
          <li> <a href=…>Click here</a></li>
        </ul>
       <li>
 </ul>
 <a href=…>Click here</a>
</div>
```

Which <a> would be selected?

Select and Action Example

- Select all <li> elements within element with ID=orderedlist, and add the class "blue"

```
$(document).ready(function() {
  $("#orderedlist > li").addClass("blue");
 });
```

```
<ol id="orderedlist">
  <li class="blue">… </li>
```

```
$("#orderedlist > li").removeClass("blue");
```

```
<ol id="orderedlist">
  <li >… </li>
```

For another example see: https://jsfiddle.net/qb2n7h5L/2/

Some Useful Selectors (For Reference Only)

$('#id')	element with the specified ID
$('p')	all elements with the specified name
$('.class')	all elements with the specified class
$('*')	all elements
$('p.class')	<p> elements having the CSS class
$('p:first')	$('p:last') / $('p:odd') / $('p:even')
$('p')[1]	gets the 2nd <p> element (0 is first)
$('p a')	<a> elements, **descended** from a <p>
$('p>a')	<a> elements, **direct child** of a <p>
$('p+a')	<a> elements, **preceded by** a <p> (next sibling)
$('div, p, a')	, <p> and <a> elements
$('li:has(ul)')	elements that have at least one descendent
$(':not(p)')	all elements but <p> elements
$('p:hidden')	only <p> elements that are hidden
$('p:empty')	<p> elements that have no child elements

Useful Selectors (Cont.) (For Reference Only)

E[a]	all E elements with attribute 'a' of any value
E[a=v]	all E elements with attribute 'a' exactly 'v'
E[a^=v]	all E elements with attribute 'a' starting with 'v'
E[a$=v]	all E elements with attribute 'a' ending with 'v'
E[a*=v]	all E elements with attribute 'a' containing 'v'

$('img'[alt])	<img> elements having an alt attribute
$('a'[href^=http://])	<a> elements with href starting with 'http://'
$('a'[href$=.pdf])	<a> elements with href ending with '.pdf'
$('a'[href*=hkust])	<a> elements with href containing 'hkust'

jQuery Functions for Formatting

$("p").css(property, value)	Set property to value
$("p").html() $("p").html(htmlString)	Get the content of the first … Set the html contents of each …
$("input").val()	For form <input>
$("p").addClass('class')	Add class="class" property to all
$("p").removeClass('class')	Remove class="class" property from all

Add Page Elements

- $('#target').before(' Inserted before #target ');
- $('#target').after(' This is added after #target ');
- $('#target').append(' Goes inside #target, at end ');
- $('#target').wrap(' ');

Take Home Message

- CSS-like selectors are like a query language to the DOM to retrieve elements matching some fairly sophisticated conditions
- There are still many jQuery and autocomplete functions not covered
- Will cover jQuery ajax and server side programming (PHP) later in this course