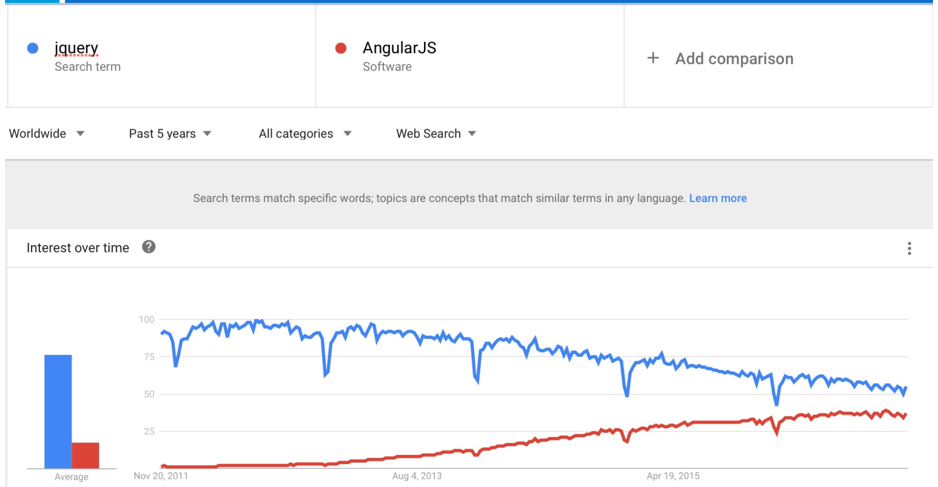
JQuery

Trends: jQuery vs. AngularJS


● jquery
Search term

● AngularJS
Software

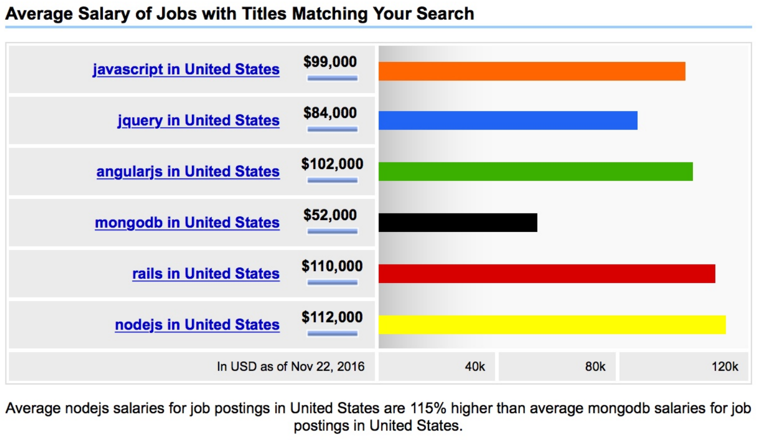
+ Add comparison

Worldwide ▾ Past 5 years ▾ All categories ▾ Web Search ▾

Search terms match specific words; topics are concepts that match similar terms in any language. Learn more

Interest over time

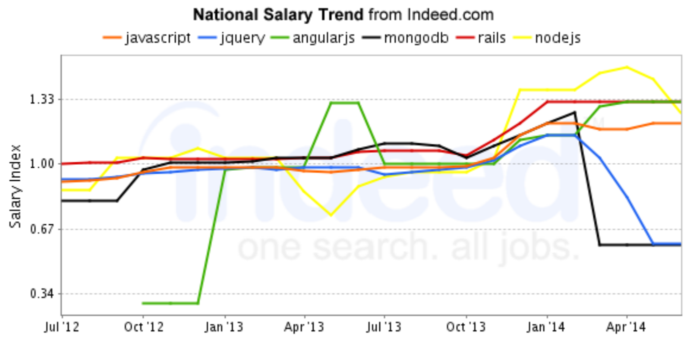
Salaries: indeed.com


Average Salary of Jobs with Titles Matching Your Search

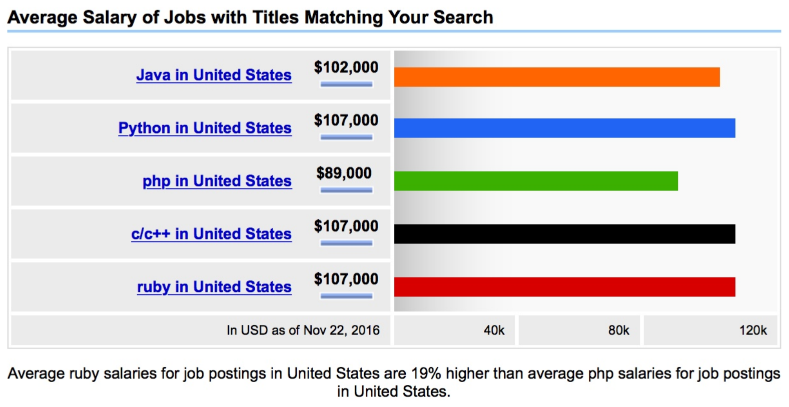
javascript in United States	$99,000	🟧
jquery in United States	$84,000	🟦
angularjs in United States	$102,000	🟩
mongodb in United States	$52,000	⬛
rails in United States	$110,000	🟥
nodejs in United States	$112,000	🟨

Average nodejs salaries for job postings in United States are 115% higher than average mongodb salaries for job postings in United States.

Salary Trends: indeed.com



Salaries: Programming Languages

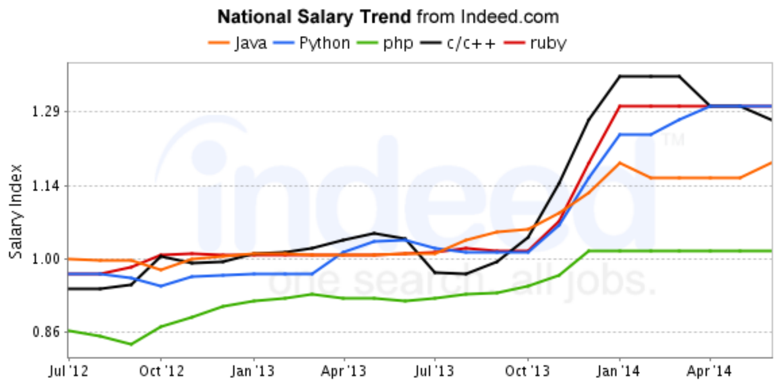


Average Salary of Jobs with Titles Matching Your Search

Java in United States	$102,000	🟧
Python in United States	$107,000	🟦
php in United States	$89,000	🟩
c/c++ in United States	$107,000	⬛
ruby in United States	$107,000	🟥
In USD as of Nov 22, 2016		40k 80k 120k

Average ruby salaries for job postings in United States are 19% higher than average php salaries for job postings in United States.

Salaries Trends : Programming Languages



7 On to jQuery…

Downloading and Installation

- Download
  - http://docs.jquery.com/Downloading_jQuery
    - Download single minimized file (e.g., jquery-1.3.2.min.js)
      - Recommend renaming to jquery.js to simplify later upgrades
- Online API and tutorials
  - http://docs.jquery.com/
- Browser Compatibility
  - Firefox: 2 or later (vs. 1.5 or later for Prototype)
  - Internet Explorer: 6.0 or later (does not work in IE 5.5)
  - Safari: 3.0 or later (vs. 2.0 or later for Prototype)
  - Opera: 9.0 or later (vs. 9.25 or later for Prototype)
  - Chrome: 1.0 or later
  - To check, run the test suite at http://jquery.com/test/

Downloading and using jQuery and jQuery UI

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>

<link rel="stylesheet"
href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/themes/smoothness/jquery-ui.css">

<script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js"></script>
```

- or download it, extract its .js files to your project folder
- documentation available on the jQuery UI API page
- the CSS is optional and only needed for widgets at the end

About jQuery

- jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, user interface, and Ajax interactions for rapid web development
- jQuery is about writing less and doing more:
  - Performance
  - Plugins
  - It's standard
  - … and fun!

Syntax

- Select some HTML Elements and perform some action on them

```
$(selector).action()
```

- Usually define functions only after the document is finished loading, otherwise elements may not be there.

```
$(document).ready(function(){
   // jQuery functions go here...
});
```

Bread and Butter: $(), or jQuery()

- This is very different from prototype's $ function.
- If this confuses you or you need prototype as well, you can try using jQuery's noConflict() method or use the jQuery() function instead
- More about this later!

`window.onload()`

- Recall that one cannot use the DOM before the page has been constructed
- jQuery uses `$(document).ready()`
  - Similar to `window.onload` but helps handle some inconsistencies across browsers
  - Similar to Prototype's `document.observe()`
- jQuery provides a compatible way to do this

`$(document).ready()`

- The DOM way

```
window.onload = function() {
// do stuff with the DOM
}
```

- The direct jQuery translation

```
$(document).ready(function() {
// do stuff with the DOM
});
```

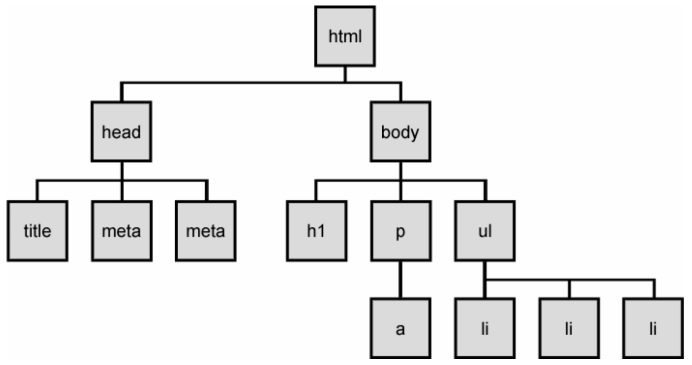
- The jQuery way

```
$(function() { // do stuff with the DOM });
```

Aspects of the DOM and jQuery

- **Identification:** how do I obtain a reference to the node that I want.
- **Traversal:** how do I move around the DOM tree.
- **Node Manipulation:** how do I get or set aspects of a DOM node.
- **Tree Manipulation:** how do I change the structure of the page.

The DOM tree



Selecting groups of DOM objects

Name	Description
getElementById	Returns a reference to the element by its ID such as "div"
getElementsByTagName	Returns all elements in the document with the specified tag name.
getElementsByName	Get all elements with the specified name.
querySelector	Returns the first element that is a descendant of the element on which it is invoked that matches the specified group of selectors.
querySelectorAll	Returns a non-live NodeList of all elements descended from the element on which it is invoked that matches the specified group of CSS selectors

jQuery Node Identification

- var List = $('a');
  - Equivalent to var List = document.getElementsByTagName('a') in DOM
- $('#banner')
  - Select a tag with a specific ID of banner
  - # part indicates that we are identifying an ID
- $('#banner').html('<h1>JavaScript was here</h1>');
  - Change the HTML inside an element
- Select all elements with the same class name
  - $('.submenu')
- Use $("css selector") to get a set of DOM elements

jQuery Node Identification

- Target a tag inside another tag
  - Use a descendant selector
    - A selector, followed by a space, followed by another selector
  - $('#navBar a'): select all links inside the unordered list
- Target a tag that's the child of another tag
  - List the parent element, followed by a > and then the child
  - $('body > p'): select all <p> tags that are the children of the <body> tag

jQuery Node Identification

- Select a tag that appears directly after another tag
  - Add a plus sign between two selectors
  - $('h2 + div')
- Select elements based on whether the element has a particular attribute
  - $('img[alt]'): find <img> tags that have the alt attribute set

More jQuery Attribute Selectors

- $("*") select all elements
- $("p:first") select the first p element
- $("[href]") select all elements with an href attribute.
- $("[href=' default.html']") select all elements with a href attribute value equal to "default.html".
- $("[href!='default.html']") select all elements with a href attribute value not equal to "default.html".
- $("[title^='def']") select all elements with an href attribute that starts with "def ".
- $("[href$='.jpg']") select all elements with an href attribute that ends with ".jpg".

jQuery Attribute Selectors: Examples

- $("p") returns all <p> elements
- $(".blah") return all elements that have class="blah"
- $("p.intro") returns all <p> elements with class="intro".
- $("#some-id") returns 1-element set (or empty set) of element with id
- $("p#demo") returns all <p> elements with id="demo"
- $$("li b span.blah")
  - Return all <span class="blah"> elements that are inside b elements, that in turn are inside li elements

CSS Selectors

- jQuery CSS selectors can be used to change CSS properties for HTML elements.
- The following example changes the background-color of all p elements to yellow
  - `$("p").css("background-color","yellow");`
- Other Examples
  - `$("#myElement").css("color", "red");`
  - `$(".myClass").css("margin", "30px");`
  - `$("body").css("background-color", "#FFFF00");`

jQuery Method Parameters

- **getter syntax:**

```
$("#myid").css(propertyName);
```

- **setter syntax:**

```
$("#myid").css(propertyName, value);
```

- **multi-setter syntax:**

```
$("#myid").css({
        'propertyName1': value1,
        'propertyName2': value2,
        ...
        });
```

- **modifier syntax:**

```
$("#myid").css(propertyName, function(idx, oldValue) {
        return newValue;
});
```

Getting/setting CSS classes in jQuery

```
function highlightField() {
      if (!$("#myid").hasClass("invalid")) {
            $("#myid").addClass("highlight");
      }
}
```

- addClass, removeClass, hasClass, toggleClass manipulate CSS classes
- similar to existing className DOM property, but don't have to manually split by spaces

jQuery method returns

method	return type
$("#myid");	jQuery object
$("#myid").children();	jQuery object
$("#myid").css("margin-left");	String
$("#myid").css("margin-left", "10px");	jQuery object
$("#myid").addClass("special");	jQuery object

What does this do?

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("p").html("<b>Hello Class!</b>");
});
</script>
</head>
<body>


<p>A simple example on <b>how to use jQuery</b>.</p>
<p>Click me away!</p>
<p>Click me too!</p>


</body>
</html>
```

What does this do?

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#test b").html("<b>Hello World</b>");
});
</script>
</head>
<body>


<p id="test">An example on <b>how to target a tag inside another tag</b>.</p>
<p>Click me away!</p>
<p>Click me too!</p>


</body>
</html>
```

What does this do?

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#test > b").html("<b>Hello World</b>");
});
</script>
</head>
<body>


<p id="test">An example on <b>what will happen here?</b>.</p>
<p>Click me away!</p>
<p>Click me too!</p>


</body>
</html>
```

What does this do?

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#test > b").html("<b>Hello World</b>");
});
</script>
</head>
<body>


<p id="test">An example on <i><b>what will happen here?</b></i>.</p>
<p>Click me away!</p>
<p>Click me too!</p>


</body>
</html>
```

What does this do?

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#test+b").html("<b>Hello World</b>");
});
</script>
</head>
<body>


<p id="test">An example on <b>how to target a tag inside another tag</b>.</p>
<p>Click me away!</p>
<p>Click me too!</p>


</body>
</html>
```

jQuery Node Identification: Summary

Syntax	**Description**
$(this)	Current HTML element
$("p")	All <p> elements
$("p.intro")	All <p> elements with class="intro"
$("p#intro")	All <p> elements with id="intro"
$("p#intro:first")	The first <p> element with id="intro"
$(".intro")	All elements with class="intro"
$("#intro")	The first element with id="intro"
$("ul li:first")	The first <li> element of the first <ul>
$("ul li:first-child")	The first <li> element of every <ul>
$(“ul li:nth-child(3)”	The third <li> element of every <ul>
$("[href$='.jpg']")	All elements with an href attribute that ends with ".jpg"
$("div#intro .head")	All elements with class="head" inside a <div> element with id="intro"

See http://www.w3schools.com/jquery/jquery_ref_selectors.asp for a complete list

Manipulating DOM Elements

- Common functions on matched elements
  - $(“tr:even”)
    $("#some-id").val()
    - Returns value of input element. Used on 1-element sets.
  - $("selector").each(function)
    - Calls function on each element. “this” set to element.
    - More about this one later!
  - $("selector").addClass("name")
    - Adds CSS class name to each. Also removeClass, toggleClass
  - $("selector").hide()
    - Makes invisible (display: none). Also show, fadeOut, fadeIn, etc.
  - $("selector").click(function)
    - Adds onclick handler. Also change, focus, mouseover, etc.
  - $("selector").html("<tag>some html</tag>")
    - Sets the innerHTML of each element. Also append, prepend

Manipulating DOM Elements

jQuery method	functionality
.hide()	toggle CSS display: none on
.show()	toggle CSS display: none off
.empty()	remove everything inside the element, innerHTML = ""
.html()	get/set the innerHTML without escaping html tags
.text()	get/set the innerHTML, HTML escapes the text first
.val()	get/set the value of a form input, select, textarea, ...
.height()	get/set the height in pixels, returns a Number
.width()	get/set the width in pixels, return a Number

Traversing Element Trees

- parent(), parents(), children(), find()
  - `$(“#myDiv”).find(“span”);`
    - Return all span descendants
  - `$(“#myDiv”).find(“*”);`
    - Return all descendants
- siblings(), next(), nextAll(), nextUntil(),
- prev(), prevAll(), prevUntil()
- first(), last(), eq(), filter(), not()

Other Useful Methods

- append(), prepend(), after(), before()
- remove(), empty()
- addClass(), removeClass(), toggleClass(), css()
- width(), height(), etc.

What does this do?

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("p").click(function(){
         $(this).html("test");
    });
});
</script>
</head>
<body>


<p id="test">Yet another example on using <b>jQuery</b></p>
<a href="http://default.html">Contact Us</a>
<p>Click me away!</p>
<p>Click me too!</p>


</body>
</html>
```

Manipulating DOM Elements: Example

- $(this).hide()
  Demonstrates the jQuery hide() method, hiding the current HTML element.
- $("#test").hide()
  Demonstrates the jQuery hide() method, hiding the element with id="test".
- $("p").hide()
  Demonstrates the jQuery hide() method, hiding all <p> elements.
- $(".test").hide()
  Demonstrates the jQuery hide() method, hiding all elements with class="test".

Chaining

- $ always returns an array of elements and methods operate on either every element when appropriate or just the first
- Example

```
var ps = $('p');
ps.css('backgroundColor', 'green');
```

```
$("#p1").css("color", "red")
.slideUp(2000)
.slideDown(2000);
```

- What will happen if there are many <p> tag on the page?

$.each

- `$.each()` takes a function and gives it both the key and the value as its first two parameters.
- Using the DOM

```
var elems = document.querySelectorAll("li");
for (var i = 0; i < elems.length; i++) {
        var e = elems[i];
        // do stuff with e
}
```

- Using jQuery

```
$("li").each(function(idx, e) {
        // do stuff with e
});
```

`$.each` Example

```
div {
    color: red;
    text-align: center;
    cursor: pointer;
    font-weight: bolder;
    width: 300px;
}
```

```
$( document.body ).click(function() {
  $( "div" ).each(function( i ) {
    if ( this.style.color !== "blue" ) {
      this.style.color = "blue";
    } else {
      this.style.color = "red";
    }
  });
});
```

jQuery Events

- Common Mouse Events:
  - `click, dblclick, mouseenter, mouseleave, hover`
- Common Keyboard Events:
  - `keypress, keydown, keyup`
- Common Form Events:
  - `submit, change, focus, blur`
- Common Document Events:
  - `load, resize, scroll, unload`

Useful jQuery Effects

$(selector).function(speed, callback)
params are optional
callback: function that is called when finished

- hide(), show(), toggle()
  - `$("#myDiv").hide(500, function() { alert("I am hidden.") });`
- fadeIn(), fadeOut(), fadeToggle(), fadeTo()
  - `$("#myDiv").fadeTo("slow", 0.5); // second param is an optional callback parameter`
- slideUp(), slideDown(), slideToggle()
- animate({params}, speed, callback)
  - goes to given params over time stop - stop animation before it's finished

Event Example

```
$("#myElement").click( function() {
    alert("You clicked me!");
});

$("p").dblclick( function() {
     $(this).hide();
});

$(".colorful").hover( function() {
    $(this).css("background-color: FF0000"); // mouse enter
  }, function () {
     $(this).css("background-color: 0000FF"); // mouse exit
}
```

jQuery Events: Example

```
<div id="outer">
  Outer
  <div id="inner">
    Inner
  </div>
</div>
<div id="other">
  Trigger the handler
</div>
<div id="log"></div>

$( "#outer" ).mouseenter(function() {
  $( "#log" ).append( "<div>Handler for
.mouseenter() called.</div>" );
});
```

Content and Attributes

- Getting and Setting Content from DOM:
  - `text(), html(), val(), attr()`
- Example:
  - `alert("Your input is: " + $("#myDiv").text()); alert`
  - `("The HTML is: " + $("#myDiv").html());`
  - `$("#myDiv").text("Hello, World!"); // set text`
  - `$("#myDiv").html("<b>Hello, World!</b>"); // set html`
- Attribute Example:
  - `alert("The URL is: " + $("#myLink").attr("href"));`

AJAX

jQuery and AJAX

- jQuery provides a nice wrapper around AJAX
  - Similar to Prototype
- `jQuery.ajax( url [, options] )`
  - Perform an asynchronous HTTP (Ajax) request

jQuery and AJAX: Options

- url: A string containing the URL to which the request is sent
- type: The type of request to make, which can be either "POST" or "GET"
- data: The data to send to the server when performing the Ajax request
- success: A function to be called if the request succeeds
- accepts: The content type sent in the request header that tells the server what kind of response it will accept in return
- dataType: The type of data expected back from the server
- error: A function to be called if the request fails
- async: Set this options to false to perform a synchronous request
- cache: Set this options to false to force requested pages not to be cached by the browser
- complete: A function to be called when the request finishes (after success and error callbacks are executed)
- contents: An object that determines how the library will parse the response
- contentType: The content type of the data sent to the server
- password: A password to be used with XMLHttpRequest in response to an HTTP access authentication request
- statusCode: An object of numeric HTTP codes and functions to be called when the response has the corresponding code
- timeout: A number that specifies a timeout (in milliseconds) for the request

jQuery and AJAX: AjaxEvents

- **.ajaxComplete()**: Register a handler to be called when Ajax requests complete.
- **.ajaxError()**: Register a handler to be called when Ajax requests complete with an error.
- **.ajaxSend()**: Attach a function to be executed before an Ajax request is sent
- **.ajaxStart()**: Register a handler to be called when the first Ajax request begins
- **.ajaxStop()**: Register a handler to be called when all Ajax requests have completed.
- **.ajaxSuccess()**: Attach a function to be executed whenever an Ajax request completes successfully

jQuery and AJAX: Methods

- **jQuery.ajax()**: Perform an asynchronous HTTP (Ajax) request.
  - **.load()**: Load data from the server and place the returned HTML into the matched element
  - **jQuery.get()**: Load data from the server using a HTTP GET request.
  - **jQuery.post()**: Load data from the server using a HTTP POST request.
  - **jQuery.getJSON()**: Load JSON-encoded data from the server using a GET HTTP request.
  - **jQuery.getScript()**: Load a JavaScript file from the server using a GET HTTP request, then execute it.
  - **.serialize()**: Encode a set of form elements as a string for submission.
  - **.serializeArray()**: Encode a set of form elements as an array of names and values.

jQuery and AJAX

```
$.ajax({
   url: "someURL.php",
   type: "POST",
   data: {},             // data to be sent to the server
   dataType: "xml"
   }).done(function(data) {
      // Do stuff with data
   }).fail(function(xhr, status) {
     // Respond to an error
});
```

Ajax: GET and POST

- Similar to Prototype
  - $.get(URL, callback);
  - $.post(URL, data, callback);

```
var myURL = “someScript.php”; // or some server-side script
$.post(
myURL, // URL of script
{ // data to submit in the form of an object
name: “John Smith”,
age: 433
},
function(data, status) { ... } // callback function
);
```

jQuery and AJAX

```
$("button").click(function(){
    $.post("demo_test.asp", function(data, status){
        alert("Data: " + data + "\nStatus: " + status);
    });
});

$(document).ajaxSuccess(function(){
    alert("AJAX request successfully completed");
});
```

jQuery and AJAX

```
$.ajax({
   url: 'http://api.joind.in/v2.1/talks/10889',
   data: {
      format: 'json'
   },
   error: function() {
      $('#info').html('<p>An error has occurred</p>');
   },
   dataType: 'jsonp',
   success: function(data) {
      var $title = $('<h1>').text(data.talks[0].talk_title);
      var $description = $('<p>').text(data.talks[0].talk_description);
      $('#info')
         .append($title)
         .append($description);
   },
   type: 'GET'
});
```

AJAX

- $(selector).load(URL, data, callback)

```
var myURL = “http://www.mysite.com/myFile.txt”;
$(“#myButton”).click( function() {
   // Pass in the URL and a callback function.
   // xhr is the XMLHttpRequest object.
   $(“#myDiv”).load(myURL, function(response, status, xhr) {
      if(status == “success”)
         alert(response);
      else if(status == “error”)
         alert(“Error: “ + xhr.statusText);
   });
});
```

Useful Links

- jQuery manipulation methods
  - http://api.jquery.com/category/manipulation/
- jQuery Selectors
  - http://api.jquery.com/category/selectors/

Creating new nodes

name	description
document.createElement("*tag*")	creates and returns a new empty DOM node representing an element of that type
document.createTextNode("*text*")	creates and returns a text node containing given text

```
// create a new <h2> node
var newHeading = document.createElement("h2");
newHeading.innerHTML = "This is a heading";
newHeading.style.color = "green";
```

Create nodes in jQuery

- The $ function to the rescue again

```
var newElement = $("<div>");

$("#myid").append(newElement);
```

- The previous example becomes with jQuery

```
$("li:contains('child')").remove();
```

JQUERY VISUAL EFFECTS

Visual Effects

- Appear
  - show
  - fadeIn
  - slideDown
  - slide effect
- Disappear
  - hide
  - fadeOut
  - slideUp
  - Blind effect
  - Bounce effect
  - Clip effect
  - Drop effect
  - Explode effect
  - Drop effect
  - Explode effect
  - Fold effect
  - Puff effect
  - Size effect

Visual effects

- Getting attention
  - Highlight effect
  - Scale effect
  - Pulsate effect
  - Shake effect

Applying effects to an element

```
element.effect(); // for some effects

element.effect(effectName); // for most effects

$("#sidebar").slideUp();

// No need to loop over selected elements, as usual
$("#results > button").effect("pulsate");
```

- the effect will begin to animate on screen (asynchronously) the moment you call it
- One method is used behind the scenes to do most of the work, animate()

Effect options

```
element.effect(effectName, {

      option:  value,
      option:  value,
      ...
});

$("#myid").effect("explode", {
      "pieces": 25
});
```

Effects chaining

```
$('#demo_chaining')
      .effect('pulsate')
      .effect('highlight')
      .effect('explode');
```

- Effects can be chained like any other jQuery methods
- Effects are queued, meaning that they will wait until the previous effects finish

Effect duration

- You can specify how long an effect takes with the duration option
- Almost all effects support this option
- Can be one of slow, normal, fast or any number in milliseconds

```
$('#myid').effect('puff', {}, duration)
```

Custom effects - animate()

```
$('#myid').animate(properties, [duration]);
```

- You can animate any numeric property you want
- You can also animate these
  - color
  - background-color

```
$('#myid')
      .animate({
            'font-size': '80px',
            'color': 'green'
      }, 1000);
```

Custom effects easing

```
$('#myid')
      .animate(properties, [duration], [easing]);
```

- Your animations don't have to progress linearly
- There are many other options
  - slide
  - easeInSin

```
$('#myid')
      .animate({
            'font-size': '80px',
            'color': 'green'
      }, 1000, 'easeOutBounce');
```

Better Custom Effects* - toggleClass()

- * if you don't need easing or special options
- use the toggleClass method with its optional duration parameter

```
.special {
        font-size: 50px;
        color: red;
}
$('#myid').toggleClass('special', 3000);
```

Adding delay()

```
$('#myid')
        .effect('pulsate')
        .delay(1000)
        .slideUp()
        .delay(3000)
        .show('fast');
```

Effect complete event

```
$("#myid").effect('puff', [options], [duration], [function]);
```

- All effects can take a fourth optional callback parameter that is called when the animation ends
- the callback can use the this keyword as usual to address the element the effect was attached to

```
$('#myid').effect('clip', {}, 'default', function() {
        alert('finished');
});
```

Drag and drop

jQuery UI provides several methods for creating drag-and-drop functionality:

- Sortable : a list of items that can be reordered
- Draggable : an element that can be dragged
- Dropable : elements on which a Draggable can be dropped

Sortable

```
$('#myid ul').sortable([options]);
```

- specifies a list (ul, ol) as being able to be dragged into any order
- with some stylings you can get rid of the list look and sort any grouping of elements
- implemented internally using Draggables and Droppables
- to make a list un-sortable again, call .sortable('destroy') on the sortable element

Sortable

- options:
  - disabled
  - appendTo
  - axis
  - cancel
  - connectWith
  - containment
  - cursor
  - cursorAt
  - delay
  - distance
  - dropOnEmpty
  - forceHelperSize
  - opacity
  - revert
  - tolerance

Sortable demo

```
<ol id="simpsons">
        <li>Homer</li>
        <li>Marge</li>
        <li>Bart</li>
        <li>Lisa</li>
        <li>Maggie</li>
</ol>


$(function() {
        $("#simpsons").sortable();
});
```

Sortable list events

event	description
change	when any list item hovers over a new position while dragging
update	when a list item is dropped into a new position (more useful)

```
$(function() {
        $("simpsons").sortable({
                'update': function(event, ui) {
                        // Do stuff here
                 }
        });
});
```

Sortable list events example

```
$(function() {
      $("#simpsons").sortable({
            'update': listUpdate
      });
});

function listUpdate(event, ui) {
      // can do anything I want here; effects,
      //an Ajax request, etc.
      ui.item.effect('shake');
}
```

Sortable "methods"

```
$('#my_list').sortable('methodName', [arguments]);

// Some examples
$('#my_list').sortable('destroy');
$('#my_list').sortable('option', 'cursor', 'pointer');
```

- jQuery plugins, like jQuery UI have an odd syntax for methods
- sortable methods
  - destroy
  - disable
  - enable
  - option
  - refresh
  - cancel

Draggable

```
$('#myid').draggable([options]);
```

- specifies an element as being able to be dragged

Draggable

- Options:
  - disabled
  - appendTo
  - addClasses
  - connectToSortable
  - delay
  - distance
  - grid
- Methods:
  - destroy
  - disable
  - enable
  - option
  - widget
- Events:
  - create
  - start
  - drag
  - stop

Draggable example

```
<div id="draggabledemo1">Draggable demo 1. Default options.
</div>
<div id="draggabledemo2">Draggable demo 2.
       {'grid': [40,40], 'revert': true}
</div>


$('#draggabledemo1').draggable();
$('#draggabledemo2').draggable({
       'revert': true,
       'grid': [40, 40]
});
```

Droppable

```
$('#myid').droppable([options]);
```

- specifies an element as being able to receive draggables

Droppable

- Options:
  - disabled
  - accept
  - activeClass
  - hoverClass
  - scope
  - greedy
  - tolerance
- Methods:
  - destroy
  - disable
  - enable
  - option
  - widget
- Events:
  - create
  - over
  - out
  - **drop**
  - activate
  - deactivate

Drag/drop shopping demo

```
<img id="shirt" src="images/shirt.png" alt="shirt" />
<img id="cup" src="images/cup.png" alt="cup" />
<div id="droptarget"></div>

$('#shirt').draggable();
$('#cup').draggable();
$('#droptarget').droppable({
       'drop': productDrop
});

function productDrop(event, ui) {
       alert("You dropped " + ui.item.attr('id'));
}
```

Auto-completing text fields

Scriptaculous offers ways to make a text box that auto-completes based on prefix strings :

- Local Autocompleter

```
var data = ["foo", "food", "foobar", "fooly", "cake"];
$('#my_text_input').autocompleter({
        'source': data
});
```

- Ajax Autocompleter: The autocompleter will make AJAX calls to the given URL providing a term parameter with the current value of the input field

```
$('#my_text_input').autocompleter({
        'source': 'http://foo.com/webservice.php'
});
```

Using a local autocompleter

```
var data = ["foo", "food", "foobar", "foolish", "foiled", "cake"];
$('#myid').autocompleter({
        'source': data
});
```

- pass the choices as an array of strings
- You can also pass an array of objects with label and value fields

```
var data = [ {'label': 'Track and Field', 'value': 'track'},
             {'label': 'Gymnastics', 'value': 'gymnastics'},
             ...
           ];
```

- the widget injects a ul elements full of choices as you type
- use the appendTo option to specify where the list is inserted

Local autocompleter demo

```
<input id="bands70s" size="40" type="text" />
<div id="bandlistarea"></div>


$('#bands70s').autocomplete({
        'source': data,
        'appendTo': '#bandlistarea'
});
```

Using an AJAX autocompleter

```
$('#my_input').autocomplete({
        'source': 'http://foo.com/webservice.php'
});
if (!isset($_GET['term'])) {
        header('HTTP/1.1 400 Invalid Request –
        No term parameter provided');
        die('No term parameter provided.');
}
$term = $_GET['term'];
$results = getCompleterResults($term);
// an array() return value print
json_encode($results);
```

- when you have too many choices to hold them all in an array, you can instead fetch subsets of choices from a server using AJAX
- instead of passing choices as an array, pass a URL from which to fetch them
  - the AJAX call is made with a term parameter
  - the choices are sent back from the server as a JSON array of strings or array of objects with label and valuefields

accordion widget

- your HTML should be pairs of headers with anchors and containers
- make the parent of these pairs an accordion

```
<div class="accordion">
     <h1><a href="#">Section 1</a></h1>
     <div>Section 1 Content</div> ...
</div>
```

```
$(function() {
     $( "#accordion" ).accordion();
});
```

tabs widget

- your HTML should be a list of link to element on your page
- the href attributes should match ids of elements on the page

```
<div class="tabs">
      <ul>
            <li><a href="#tab1">Tab 1</a></li>
            <li><a href="#tab2">Tab 2</a></li> ...
      </ul>
<div id="tab1">Tab 1 Content</div>
<div id="tab2">Tab 2 Content</div> ... </div>
```

```
$(function() { $( "#tabs" ).tabs(); });
```

jQuery UI theming

- jQuery UI uses classes gratuitously so that we can style our widgets however we want
- there are two kinds of classes used
  - framework classes which exist for all widgets
  - widget specific classes

kind	**classes**
Layout Helpers	.ui-helper-hidden, .ui-helper-reset, .ui-helper-clearfix
Widget Containers	.ui-widget, .ui-widget-header, .ui-widget-content
Interaction States	.ui-state-default, .ui-state-hover, .ui-state-focus, .ui-state-active