JavaScript

Regular Expression

Regular Expressions and Pattern Matching

- When need to validate the form elements' values, such as:
  - name, address, and birth date
- You can use simple JavaScript expression to validate the form input values.
- To check exact value, it is really simple.
- if(form1.username.value==“bat”)

  alert(“Welcome, Bat”);
- But, when you need to check non-exact values for example, check whether it has only characters, or numbers.
  - check every character ???
- When need to check the correct format such as email address (someone@abc.com)

What is regular expression

- With the addition of regular expressions, form validation can be much more sophisticated and precise.
- Regular expressions are useful for searching for patterns in input data, and replacing the data with something else or splitting it up into substrings.
- A regular expression is really just a sequence of characters that specify a pattern to be matched against a string of text when performing searches and replacements.
- A simple regular expression consists of a character or set of characters that matches itself.
- The regular expression is normally delimited by forward slashes; for example, */abc/.*

Metacharacter

- JavaScript provides a large variety of regular expression metacharacters to match the pattern.
- A metacharacter is a special character that represents something other than itself, such a a ^, $,*, and so on.
- They are placed within in the regular expression to control the search pattern;
- for example, /^abc/ means look for the pattern *abc at the beginning of the line.*
- *You can look for* strings containing
  - only digits,
  - only alphas,
  - a digit at the beginning of the line followed by any number of alphas,
  - a line ending with a digit, and so on.
- When searching for a pattern of characters, the possibilities of fine-tuning your search are endless.

Creating a Regular Expression

- A regular expression is a pattern of characters. It shouldn't be any surprise by now.
- Java-Script regular expressions are objects.
- When you create a regular expression, you test the regular expression against a string.
  - For example, the regular expression */green/ might* be matched against the string *"The green grass grows". If green is contained in the string,* then there is a successful match.
- Building a regular expression is like building a JavaScript string.
- You can create a *String object the literal way or you can use the String() constructor method.*
- *To* build a regular expression object, you can assign a literal regular expression to a variable, or you can use the *RegExp constructor to create and return a regular expression object.*

The Literal Way

- var variable_name = /regular expression/options;

EXAMPLE

- var myreg = /mongolia/;
- var reobj = /student/ig;

Option	***Purpose***
• *i*	*Used to ignore case.*
• *g*	*Used to match for all occurrences of the pattern in the string.*
• *m*	*Used to match over multiple lines.*

The Constructor Method

- var variable_name = new RegExp("regular expression", "options");

EXAMPLE

- var myreg = new RegExp(”mongolia");
- var reobj = new RegExp(“student", "ig");

Testing the Expression

- The *RegExp object has two methods that can be used to test for a match in a string,*
  - the *test() method*
  - *the exec() methodwhich are quite similar.*
- *The test() method* searches for a regular expression in a string and returns *true if it matched and false if* it didn't.

- The *exec() method succeeds, it returns an array of information including the search string,* and the parts of the string that matched.
  - If it fails, it returns *null. This is similar to the match() method of the String object.*

Test() method

- var string=“string to be tested";

  // Literal way
- var **regex = /regular expression/;**

  // Constructor way
- var **regex=new RegExp("regular expression");**
- **regex.test(string);**

// Returns either true or false

Or

- /regular expression/.test("string");

The test() mehtod example

```
<html>
<head><title>Regular Expression Objects the Literal Way</title>
<script language = "JavaScript">
var myString="My gloves are worn for wear.";
var regex = /love/;                    // Create a regular expression object
if (regex.test(myString))
    alert("Found pattern!");
else
    alert("No match.");
</script>
</head>
<body></body>
</html>
```

The *exec() Method*

- array = regular_expression.exec(string);

EXAMPLE

- list = /ring/.exec("mongolia");

exec() method example

```
<html>
<head><title>The exec() method</title>
<script type="text/javascript">
var myString="My lovely gloves are worn for wear, Love.";
var regex = /love/i;                    // Create a regular expression object
var array=regex.exec(myString);
if (regex.exec(myString))
    alert("Matched! " + array);
else
    alert("No match.");
</script>
</head>
<body></body>
</html>
```

Class Properties of the *RegExp Object*

Property	What It Describes
• *input*	*Represents the input string being matched.*
• *lastMatch*	*Represents the last matched characters.*
• *lastParen*	*Represents the last parenthesized substring pattern match.*
• *leftContext*	*Represents the substring preceding the most recent pattern match.*
• *RegExp.$**	*Boolean value that specifies whether strings should be searched over* multiple lines; same as the multiline property.
• *RegExp.$&*	*Represents the last matched characters.*
• *RegExp.$_*	*Represents the string input that is being matched.*
• *RegExp.$‘*	*Represents the substring preceding the most recent pattern match (see* the *leftContext property).*
• *RegExp.$’*	*Represents the substring following the most recent pattern match (see* the *rightContextproperty).*
• *RegExp.$+*	*Represents the last parenthesized substring pattern match (see the lastParen property).*
• *RegExp.$1,$2,$3...*	*Used to capture substrings of matches.*
• *rightContext*	*Represents the substring following the most recent pattern match.*

String Methods Using Regular Expressions

- *match(regex)*

  Returns substring in regex or null.
- *replace(regex, replacement)*

  Substitutes regex with replacement string.
- *search(regex)*

  Finds the starting position of regex in string.
- *split(regex)*

  Removes regex from string for each occurrence.

The match() method

- The *match() method, like the exec() method, is used to search for a pattern of characters* in a string and returns an array where each element of the array contains each matched pattern that was found.
- If no match is found, returns *null. With the g flag, match()* searches globally through the string for all matching substrings.

- array = String.match(regular_expression);

EXAMPLE

- matchList = "Too high, too low".**match(/too/ig);**

The search() method

- The *search() method is used to search for a pattern of characters within a string, and* returns the index position of where the pattern was found in the string.
- The index starts at zero.
- If the pattern is not found, –1 is returned. For basic searches, the *String object's indexOf() method works fine, but if you want more complex pattern matches, the search() method is used, allowing you to use regular expression metacharacters to further* control the expression.
- var index_value = String.search(regular_expression);

EXAMPLE

- var position = "The world".**search(/world/);**

The search() method example

```
<html>
<head>
<title>The search() Method</title>
</head>
<body bgcolor="yellow">
<big>
<font face="arial, helvetica">
<script type="text/javascript">
var myString="I love the smell of clover.“
var regex = /love/;
var index=myString.search(regex);
document.write("Found the pattern "+ regex+ " at position “ +index+"<br />");
</script>
</font></big>
</body>
</html>
```

The *replace() Method*

- The *replace() method is used to search for a string and replace the string with another* string.
- The *i modifier is used to turn off case sensitivity*
- *The g modifier makes the* replacement global; that is, all occurrences of the found pattern are replaced with the new string.

The *replace() Method example*

```
<html>
<head>
<title>The replace() Method</title>
</head>
<body bgcolor="yellow">
<script type = "text/javascript">
var myString="Tommy has a stomach ache.“
var regex = /tom/i;        // Turn off case sensitivity
var newString=myString.replace(regex, "Mom");
document.write(newString +"<br />");
</script>
</body>
</html>
```

The *split() Method*

- The *String object's split() method splits a single text string into an array of substrings.*
- array = String.split(/delimiter/);

EXAMPLE

- **splitArray = "red#green#yellow#blue".split(/#/);**
- *splitArray is an array of colors. splitArray[0] is "red"*

The Metacharacters

- /^a...c/
- The expression reads: Search at the beginning of the line for an *a, followed by any* three single characters, followed by a *c.*
- *It will match, for example, abbbc, a123c, a c, aAx3c, and so on, but only if those patterns were found at the beginning of the line.*

Single Characters and Digits

Metacharacter/Metasymbol	*What It Matches*
.	Matches any character except newline
[a–z0–9]	*Matches any single character in set*
[^a–z0–9]	*Matches any single character* ***not*** *in set*
\d	*Matches one digit*
\D	*Matches a nondigit, same as [^0–9]*
\w	*Matches an alphanumeric (word) character*
\W	*Matches a nonalphanumeric (nonword) character*

Whitespace Characters

\0	*Matches a null character*
\b	*Matches a backspace*
\f	*Matches a formfeed*
\n	*Matches a newline*
\r	*Matches a return*
\s	*Matches whitespace character, spaces, tabs, and newlines*
\S	*Matches nonwhitespace character*
\t	*Matches a tab*

Anchored Characters

^	*Matches to beginning of line*
$	*Matches to end of line*
\A	*Matches the beginning of the string only*
\b	*Matches a word boundary (when not inside [ ])*
\B	*Matches a nonword boundary*
\G	*Matches where previous m//g left off*
\Z	*Matches the end of the string or line*
\z	*Matches the end of string only*

Anchored chars example

- var reg_expression = **/6\d\d/;**
  var textString=a612a;
  var result=reg_expression.test(textString);
  //Result is true

- var reg_expression = **/^6\d\d$/;**
  var textString=a612a;
  var result=reg_expression.test(textString);
  //Result is false

Repeated Characters

• *x?*	*Matches 0 or 1 of x*
• *x**	*Matches 0 or more of x*
• *x+*	*Matches 1 or more of x*
• *(xyz)+*	*Matches one or more patterns of xyz*
• *x{m,n}*	*Matches at least m of x and no more than n of x*

•
```
var string1="ab12345678344555443 7AB“
string1=string1.replace(/ab[0-9]*/, "X");
//Result is "XAB“
```

Alternatives

• was|were|will	Matches one of was, were, or will

Metasymbols

Symbol	*What It Matches*	*Character Class*
• *\d*	*One digit*	*[0-9]*
• *\D*	*One nondigit*	*[^0-9]*
• *\s*	*One whitespace character* (*tab, space, newline,*carriage return,formfeed, vertical tab)	
• *\S*	*One nonspace character*	
• *\w*	*One word character*	*[A-Za-z0-9_]*
• *\W*	*One nonword character*	*[^A-Za-z0-9]*

Example

```
var reg_expression = /[A-Za-z0-9_]/;
    // A single alphanumeric word character
var textString=prompt("Type a string of text","");
var result=reg_expression.test(textString);
```

-
  ```
  var regex = /^\(?\d{1,3}\)?-?\s*\d{8}$/;
  var phone="(9765)- 99008800“
  if(regex.test( phone))
          alert("true");
  else
          alert("false")
  ```

JavaScript library

JQuery

What is JavaScript libraries

- It is a library of pre-written JavaScript which allows for
  - easier development of JavaScript-based applications
  - especially for AJAX
  - web related technologies

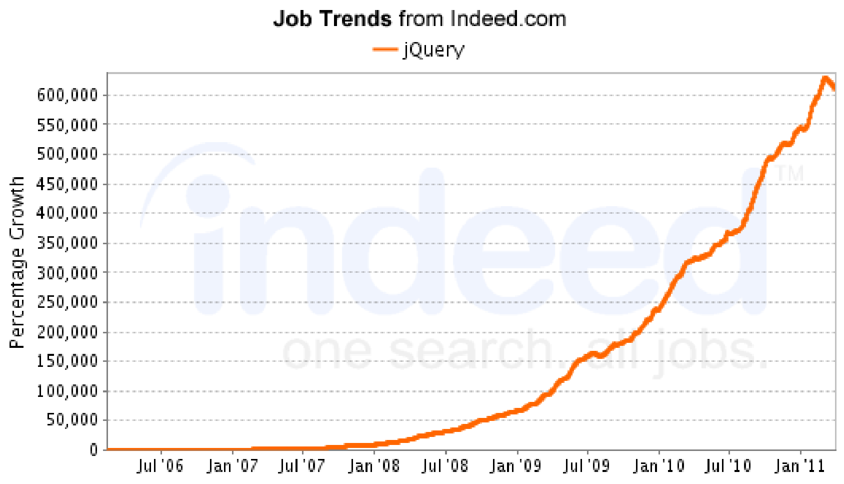
- Some JavaScript libraries, such as YUI, are classified as frameworks since they exhibit full-stack capabilities and properties not found in general JavaScript libraries.

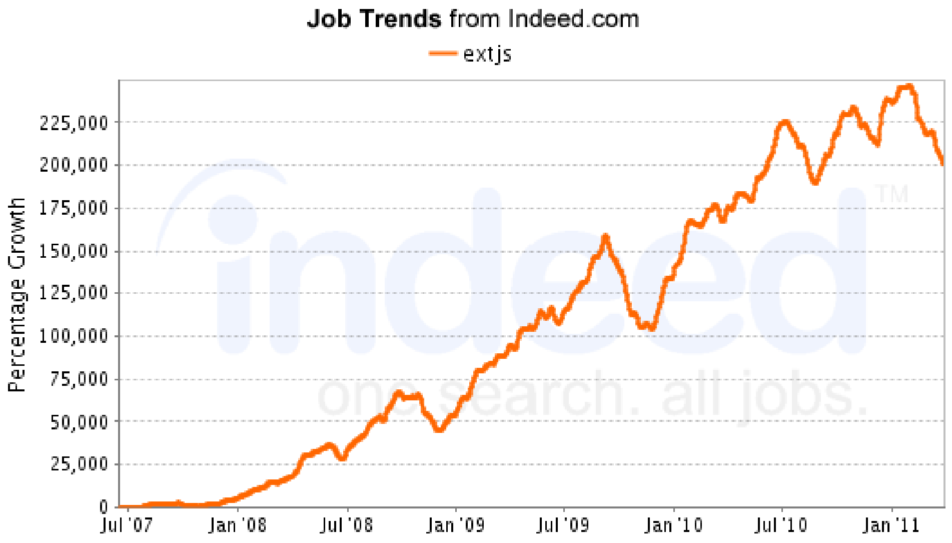
Some statistics

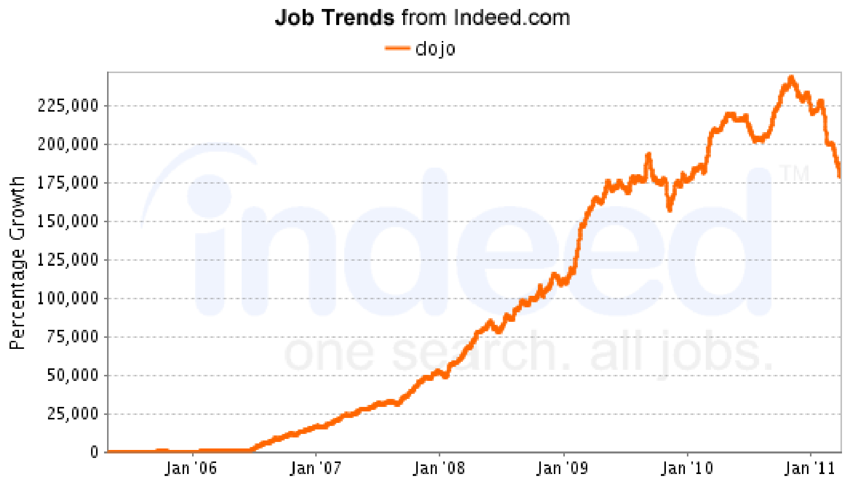
- Indeed.com searches millions of jobs from thousands of job sites.

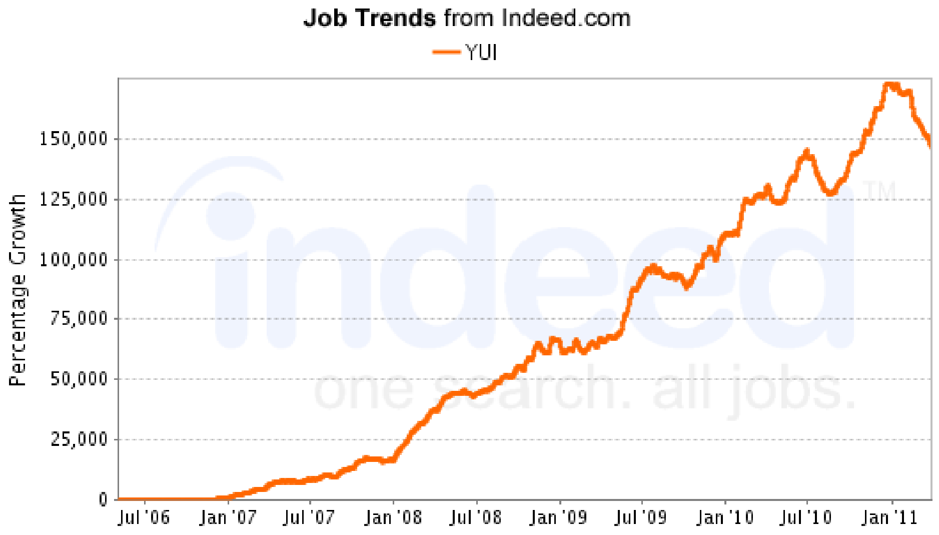
- This job trends graph shows relative growth for jobs we find matching your search terms.

jQuery trend









About the library size

Usually JavaScript libraries available in two formats:

- Uncompressed
  Good for debugging and to understand what is behind
- Compressed (Minimized and Gzipped)
  Which allows smaller file size

How to use JS libraries

Use local copy

```
<head>
< script type="text/javascript" src="jquery.js"></script>
< /head>
```

Use CDN

```
<script src="http://ajax.googleapis.com/ajax/libs/dojo/1.6/dojo/dojo.xd.js"
    type="text/javascript"></script>
```

jQuery

- jQuery is "write less, do more" JavaScript library.
- lightweight ~83kb
- CSS3 Compliant
- Cross-Browser IE6+, FF2.0+, Safari 3.0, Opera 9.0+, Chrome
- The jQuery library contains the following features:
  - HTML element selections
  - HTML element manipulation
  - CSS manipulation
  - HTML event functions
  - JavaScript Effects and animations
  - HTML DOM traversal and modification
  - AJAX
  - Utilities

jQuery example

```
<!DOCTYPE html> <html lang="en"> <head> <meta charset="utf-8“> <title>jQuery demo</title>
 <script src="../js/jquery-2.1.4.min.js"></script>
</head>

<body>
<a href="http://jquery.com/">jQuery</a>

     <script>
    $(document).ready(
           function() {
                    $("a").click(
                             function(event){
                                      alert("Link removed");
                                      event.preventDefault(); });
                     }
    );
</script> </body> </html>
```

jQuery syntax

- The jQuery syntax is tailor made for **selecting** HTML elements and perform some **action** on the element(s).

  syntax is: **$(selector).action()**

- A dollar sign to define jQuery
- A (selector) to "query (or find)" HTML elements
- A jQuery action() to be performed on the element(s)

jQuery Syntax Examples

- **$("this").hide()**
  Hiding the current HTML element.

- **$("#test").hide()**
  Hiding the element with id="test".

- **$("p").hide()**
  Hiding all <p> elements.

- **$(".test").hide()**
  Hiding all elements with class="test".

jQuery Element Selectors

- Borrowing from CSS 1–3, and then adding its own, jQuery offers a powerful set of tools for matching a set of elements in a document.
- If you wish to use any of the meta-characters (such as !"#$%&'()*+,./:;<=>?@[\]^`{|}~) as a literal part of a name, you must escape the character with two backslashes: \\. For example, if you have an element with id="foo.bar", you can use the selector $("#foo\\.bar"). The W3C CSS specification contains

  jQuery uses CSS selectors to select HTML elements.

- $("p") selects all <p> elements.
- $("p.intro") selects all <p> elements with class="intro".
- $("p#demo") selects the first <p> element with id="demo".

jQuery Attribute Selectors

jQuery uses XPath expressions to select elements with given attributes.

- $("[href]") select all elements with an href attribute.
- $("[href='#']") select all elements with an href value equal to "#".
- $("[href!='#']") select all elements with an href attribute NOT equal to "#".
- $("[href$='.jpg']") select all elements with an href attribute that ends with ".jpg".

jQuery CSS Selectors

- $("p").css("background-color","yellow");

jQuery Event functions

- The jQuery event handling methods are core functions in jQuery.
- Event handlers are method that are called when "something happens" in HTML.

- **$(document).ready(function)**
  Binds a function to the ready event of a document
  (when the document is finished loading)

- **$(*selector*).click(function)**
  Triggers, or binds a function to the click event of selected elements

- **$(*selector*).dblclick(function)**
  Triggers, or binds a function to the double click event of selected elements

- **$(*selector*).focus(function)**
  Triggers, or binds a function to the focus event of selected elements

- **$(*selector*).mouseover(function)**
  Triggers, or binds a function to the mouseover event of selected elements

jQuery Event example

- **$("button").click(function() {..some code... })**

-
  ```
  <html>< head>
  < script type="text/javascript" src="jquery.js"></script>
  < script type="text/javascript">

  $(document).ready(function(){
                                        $("button").click(function()            {
                                                                                $("p").hide();
                                                                                });
                                           });



  < /script>
  < /head>

  < body>
  < h2>This is a heading</h2>
  < p>This is a paragraph.</p>< p>This is another paragraph.</p>
  < button>Click me</button>< /body></html>
  ```

jQuery Effects

- Hide, Show, Toggle, Slide, Fade, and Animate.
- $(selector).**hide**(speed,callback)
- $(selector).**show**(speed,callback)
- $(selector).**toggle**(speed,callback)
- $(selector).**slideDown**(speed,callback)
- $(selector).**slideUp**(speed,callback)
- $(selector).**slideToggle**(speed,callback)
- $(selector).**fadeIn**(speed,callback)
- $(selector).**fadeOut**(speed,callback)
- $(selector).**fadeTo**(speed,opacity,callback)
- $(selector).**animate**({params},[duration],[easing],[callback])

- speed parameters can have:
  - "slow", "fast", "normal", or milliseconds.
- Callback The callback parameter is the name of a function to be executed after the function completes.

Effect Examples

-
  ```
  $("#hide").click(function(){  $("p").hide();});
  $("#show").click(function(){ $("p").show(); });
  $("button").click(function(){$("p").hide(1000);});
  $("button").click(function(){$("div").fadeTo("slow",0.25);});
  $("button").click(function(){$("div").fadeOut(4000); });
  $("div").animate({left:"100px"},"slow");
  ```

jQuery Callback Functions

- A callback function is executed after the current animation (effect) is finished.
- JavaScript statements are executed line by line.
- However, with animations, the next line of code can be run even though the animation is not finished. This can create errors.
- To prevent this, you can create a callback function. The callback function will not be called until after the animation is finished.

-
  ```
  $("p").hide(1000, function(){
          alert("The paragraph is now hidden");
                          });
  ```

Changing HTML Content

- **$(selector).html(content)**
  replaces html
- $("p").html("hello");

- **$(selector).append(content)**
  inside html, after
- **$(selector).prepend(content)**
  inside html, before
- **$(selector).after(content)**
  after element
- **$(selector).before(content)**
  before element

jQuery CSS Manipulation

- jQuery has one important method for CSS manipulation: css()
- The css() method has three different syntaxes, to perform different tasks.

css(name) -	Return CSS property value
css(name,value) -	Set CSS property and value
css({properties}) -	Set multiple CSS properties and values

- $(this).css("background-color");
- $("p").css("background-color","yellow");
- $("p").css({"background-color":"yellow","font-size":"200%"});

Size Manipulation

- jQuery has two important methods for size manipulation.
  - height()
  - width()

- $("#div1").height("200px");
- $("#div2").width("300px");

jQuery UI
The user interface library of jQuery

What is jQuery UI

- To build highly interactive web applications
  - abstractions for low-level
    - interaction
    - animation
    - advanced effects
  - high-level
    - themeable widgets
    - built on top of the jQuery JavaScript Library

UI

- Interactions
  provide the ability to drag/drop, resize ,…
- Widgets
  prebuilt components button, dialog box,slider, tab
- Effects
  provide change style, animation to element
- Utilities

- Examples from site