Regular Expression pattern

The patterns used in RegExp can be very simple, or very complicated, depending on what you're trying to accomplish. To match a simple string like "Hello World!" is no harder than actually writing the string, but if you want to match an e-mail address or html tag, you might end up with a very complicated pattern that will use most of the syntax presented in the table below.

Pattern	**Description**
Escaping	
\	Escapes special characters to literal and literal characters to special. E.g: /\(s\)/ matches '(s)' while /(\s)/ matches any whitespace and captures the match.
Quantifiers	
{**n**}, {**n**,}, {**n**,**m**}, *, +, ?	Quantifiers match the preceding subpattern a certain number of times. The subpattern can be a single character, an escape sequence, a pattern enclosed by parentheses or a character set. {**n**} matches exactly *n* times. {**n**,} matches *n* or more times. {**n**,**m**} matches *n* to *m* times. * is short for {0,}. Matches zero or more times. + is short for {1,}. Matches one or more times. ? is short for {0,1}. Matches zero or one time. E.g: /o{1,3}/ matches 'oo' in "tooth" and 'o' in "nose".
Pattern delimiters	
(**pattern**),	Matches entire contained pattern.

Pattern	Description
(?:**pattern**)	(**pattern**) captures match. (?:**pattern**) doesn't capture match E.g: /(d).\1/ matches and captures 'dad' in "abcdadef" while /(?:.d){2}/ matches but doesn't capture 'cdad'. **Note:** (?:**pattern**) is a JavaScript 1.5 feature.
Lookaheads	
(?=**pattern**), (?!**pattern**)	A lookahead matches only if the preceding subexpression is followed by the pattern, but the pattern is not part of the match. The subexpression is the part of the regular expression which will be matched. (?=**pattern**) matches only if there is a following **pattern** in input. (?!**pattern**) matches only if there is not a following **pattern** in input. E.g: /Win(?=98)/ matches 'Win' only if 'Win' is followed by '98'. **Note:** Lookahead is a JavaScript1.5 feature.
Alternation	
|	Alternation matches content on either side of the alternation character. E.g: /(a|b)a/ matches 'aa' in "dseaas" and 'ba' in "acbab".
Character sets	
[**characters**], [^**characters**]	Matches any of the contained characters. A range of characters may be defined by using a hyphen.

Pattern	Description
	[**characters**] matches any of the contained *characters*. [^**characters**] negates the character set and matches all but the contained *characters* E.g: /[abcd]/ matches any of the characters 'a', 'b', 'c', 'd' and may be abbreviated to /[a-d]/. Ranges must be in ascending order, otherwise they will throw an error. (E.g: /[d-a]/ will throw an error.) /[^0-9]/ matches all characters but digits. **Note:** Most special characters are automatically escaped to their literal meaning in character sets.
Special characters	
^, $, ., ? and all the highlighted characters above in the table.	Special characters are characters that match something else than what they appear as. ^ matches beginning of input (or new line with *m* flag). $ matches end of input (or end of line with *m* flag). . matches any character except a newline. ? directly following a quantifier makes the quantifier non-greedy (makes it match minimum instead of maximum of the interval defined). E.g: /(.)*?/ matches nothing or '' in all strings. **Note:** Non-greedy matches are not supported in older browsers such as Netscape Navigator 4 or Microsoft Internet Explorer 5.0.
Literal characters	
All characters	Mapped directly to the corresponding character.

Pattern	Description
except those with special meaning.	E.g: /a/ matches 'a' in "Any ancestor".
Backreferences	
\n	Backreferences are references to the same thing as a previously captured match. *n* is a positive nonzero integer telling the browser which captured match to reference to. /(\S)\1(\1)+/g matches all occurrences of three equal non-whitespace characters following each other. /<(\S+).*>(.*)<\/\1>/ matches any tag. E.g: /<(\S+).*>(.*)<\/\1>/ matches '<div id="me">text</div>' in "text<div id=\"me\">text</div>text".
Character Escapes	
\f, \r, \n, \t, \v, \0, [\b], \s, \S, \w, \W, \d, \D, \b, \B, \c**X**, \x**hh**, \u**hhhh**	\f matches form-feed. \r matches carriage return. \n matches linefeed. \t matches horizontal tab. \v matches vertical tab. \0 matches NUL character. [\b] matches backspace. \s matches whitespace (short for [\f\n\r\t\v\u00A0\u2028\u2029]). \S matches anything but a whitespace (short for [^\f\n\r\t\v\u00A0\u2028\u2029]). \w matches any alphanumerical character (word characters) including underscore (short for [a-zA-Z0-9_]). \W matches any non-word characters (short for [^a-zA-Z0-9_]).

Pattern	Description
	\d matches any digit (short for [0-9]). \D matches any non-digit (short for [^0-9]). \b matches a word boundary (the position between a word and a space). \B matches a non-word boundary (short for [^\b]). \c**X** matches a control character. E.g: \cm matches control-M. \x**hh** matches the character with two characters of hexadecimal code *hh*. \u**hhhh** matches the Unicode character with four characters of hexadecimal code *hhhh*.