PHP Form Validation

Character Sets in Regular Expressions

Regular expression character sets denoted by a pair of brackets `[]` will match any of the characters included within the brackets. For example, the regular expression `con[sc]en[sc]us` will match any of the spellings `consensus`, `concensus`, `consencus`, and `concencus`.

Optional Quantifiers in Regular Expressions

In Regular expressions, optional quantifiers are denoted by a question mark `?`. It indicates that a character can appear either 0 or 1 time. For example, the regular expression `humou?r` will match the text `humour` as well as the text `humor`.

Literals in Regular Expressions

In Regular expression, the `literals` are the simplest characters that will match the exact text of the literals. For example, the regex `monkey` will completely match the text `monkey` but will also match `monkey` in text `The monkeys like to eat bananas.`

Fixed Quantifiers in Regular Expressions

In Regular expressions, fixed quantifiers are denoted by curly braces `{}`. It contains either the exact quantity or the quantity range of characters to be matched. For example, the regular expression `roa{3}r` will match the text `roaaar`, while the regular expression `roa{3,6}r` will match `roaaar`, `roaaaar`, `roaaaaar`, or `roaaaaaar`.

Alternation in Regular Expressions

Alternation indicated by the pipe symbol `|`, allows for the matching of either of two subexpressions. For example, the regex `baboons|gorillas` will match the text `baboons` as well as the text `gorillas`.

Anchors in Regular Expressions

Anchors (hat `^` and dollar sign `$`) are used in regular expressions to match text at the start and end of a string, respectively. For example, the regex `^Monkeys: my mortal enemy$` will completely match the text `Monkeys: my mortal enemy` but not match `Spider Monkeys: my mortal enemy` or `Monkeys: my mortal enemy in the wild`. The `^` ensures that the matched text begins with `Monkeys`, and the `$` ensures the matched text ends with `enemy`.

Regular Expressions

Regular expressions are sequence of characters defining a pattern of text that needs to be found. They can be used for parsing the text files for specific pattern, verifying test results, and finding keywords in emails or webpages.

Wildcards in Regular expressions

In Regular expression, wildcards are denoted with the period `.` and it can match any single character (letter, number, symbol or whitespace) in a piece of text. For example, the regular expression `.....` will match the text `orangutan`, `marsupial`, or any other 9-character text.

Regular Expression Ranges

Regular expression ranges are used to specify a range of characters that can be matched. Common regular expression ranges include: [A-Z]. : match any uppercase letter [a-z]. : match any lowercase letter [0-9]. : match any digit [A-Za-z] : match any uppercase or lowercase letter.

Shorthand Character Classes in Regular Expressions

Shorthand character classes simplify writing regular expressions. For example, `\w` represents the regex range `[A-Za-z0-9_]`, `\d` represents `[0-9]`, `\W` represents `[^A-Za-z0-9_]` matching any character not included by `\w`, `\D` represents `[^0-9]` matching any character not included by `\d`.

Kleene Star & Kleene Plus in Regular Expressions

In Regular expressions, the Kleene star(`*`) indicates that the preceding character can occur 0 or more times. For example, `meo*w` will match `mew`, `meow`, `meooow`, and `meooooooooooooow`. The Kleene plus(`+`) indicates that the preceding character can occur 1 or more times. For example, `meo+w` will match `meow`, `meooow`, and `meooooooooooooow`, but not match `mew`.

Grouping in Regular Expressions

In Regular expressions, grouping is accomplished by open `(` and close parenthesis `)`. Thus the regular expression `I love (baboons|gorillas)` will match the text `I love baboons` as well as `I love gorillas`, as the grouping limits the reach of the `|` to the text within the parentheses.

$_POST Superglobal Variable

`$_POST`, the PHP superglobal variable, is an array that contains data from the client's POST request. This data is translated from the HTTP request headers

PHP filter_var function

In PHP, `filter_var` is a function that filters a variable with a specified filter.

As its first argument, `filter_var()` takes a variable. As its second, it takes an ID representing the type of filtering that should be performed. The third argument is optional and can be used to specify options for the filter.

There are several filters for sanitizing common input types, including `FILTER_SANITIZE_EMAIL`, which is a common filter that removes illegal characters for an email address.

Filters can also be used to validate that text matches a pattern. For example, `FILTER_VALIDATE_EMAIL` returns the variable if it contains only valid email characters. Otherwise, it returns `false`.

```
echo filter_var("
<p>u</p>pies@codecademy.com",
FILTER_SANITIZE_EMAIL);
# prints "puppies@codecademy.com"
echo filter_var("
<p>u</p>pies@codecademy.com",
FILTER_VALIDATE_EMAIL);
# returns false and prints nothing
```

PHP preg_match function

In PHP, `preg_match` performs a regular expression match. The first argument is the pattern to match and the second argument is the variable to check. It returns `1` if it matches, `0` if it doesn't, and `FALSE` if there was an error.

```
$pattern = '/^[(]*([0-9]{3})[- .)]*[0-9]
{3}[- .]*[0-9]{4}$/';

preg_match($pattern, "(999)-555-2222");
// Returns: 1

preg_match($pattern, "555-2222"); //
Returns: 0
```

PHP strlen function

In PHP, `strlen` is a function that returns the length of the string passed in to it.

```
echo strlen("Codecademy");
# prints "10"
```

PHP preg_replace function

In PHP, `preg_replace` is a function that can replace portions of an input string based on a regular expression.
The first argument is the pattern to search for. The second argument is the new text to replace the pattern with. The third argument is the string to operate on.

```
$one = "codeacademy";
$two = "CodeAcademy";
$three = "code academy";
$four = "Code Academy";

$pattern = "/[cC]ode\s*[aA]cademy/";
$codecademy = "Codecademy";

echo preg_replace($pattern, $codecademy, $one);
// Prints: Codecademy

echo preg_replace($pattern, $codecademy, $two);
// Prints: Codecademy

echo preg_replace($pattern, $codecademy, $three);
// Prints: Codecademy

echo preg_replace($pattern, $codecademy, $four);
// Prints: Codecademy
```

PHP header function

The PHP `header()` function can be used to perform redirects.
We call the `header()` function on a string that begins with `"Location: "`, followed by the URL we want to redirect the user to. For example: `"Location: https://www.codecademy.com/learn/learn-php/"`.
After invoking the header() function we'll want to use the language construct `exit` to terminate the current script.

```
if (/* Is the submission data validated? */) {
  header("Location: https://www.codecademy.com/learn/learn-php/");
  exit;
}
```

PHP htmlspecialchars function

In PHP, `htmlspecialchars` is a function that transforms special characters into HTML entities.

```
$text = "This text is <b>bold</b>.";

echo htmlspecialchars($text);
// prints This text is &lt;b&gt;bold&lt;/b&gt;.
```

PHP trim function

The built-in PHP `trim()` function is used to remove whitespace from the beginning and end of a string.

```
echo trim("          hello world     ");
# prints "hello world"
```

Print Share