- Home
- About

FrameAutomation.com

Making FrameMaker faster and more efficient

Using a regular expression to convert an image name to a path

by Rick Quatro on July 1, 2011

Mike and I started an interesting discussion about using ExtendScript to import images from a base file name that the user enters. One of the key tasks is to figure out how to take a base file name that uses their syntax and derive a full path from it. Based on Mike's examples, I came up with the following ExtendScript function:

```
alert (getImagePath ("MSP4321"));

function getImagePath (imageName) {

  var regex = /([A-Z]{3})((\d)\d)(\d\d)/;
  var path = "\\\\server\\Graphics\\$1\\$1$3000-$3999\\$1$200-$299\\$1$2$4.pdf";

  if (imageName.search(regex) !== -1) {
    return imageName.replace (regex, path);
  } else {
    return null;
  }
}
```

This function illustrates the power of regular expressions. The basic regular expression is this:

Find 3 uppercase letters: [A-Z]{3}

followed by four digits: \d\d\d\d

The parentheses "capture" parts of the match and put them into special variables in the form of $#, where # is a number from 1 to 9. To figure out what number goes with what set of parentheses, you start counting from the left. So, in our regular expression

([A-Z]{3}) will be $1. In our example, this will be MSP.

((\d)\d) will be $2. In our example, this will be 43.

(\d) nested in $2 will be $3. In our example, this will be 4.

and (\d\d) will be $4. In our example, this will be 21.

We can use these variables to build up our path. Notice the $# variables in the replacement string (color-coded for clarity):

```
"\\\\server\\Graphics\\$1\\$1$3000-$3999\\$1$200-$299\\$1$2$4.pdf"
```

This will become

```
"\\\\server\\Graphics\\MSP\\MSP4000-4999\\MSP4300-4399\\MSP4321.pdf"
```

The backslash in JavaScript is an escape character, so to get the literal backslash character, we have to use 2 where we want 1.

Copy the function into the ExtendScript Toolkit and try a few filenames that use this format, and you will get the correct path every time. Regular expressions are a deep topic, but as you can see, they are very powerful for parsing tasks like this.

-Rick

{ 1 comment… read it below or add one }

Mike July 5, 2011 at 12:06 pm

> Wow, your regex approach is far more streamlined than what I had in mind. This function is the core of the script that I hope to develop. In addition, it can be used in other scripts that do things such as retrieve files for editing or viewing. Very cool, thanks.
>
> -mike

Leave a Comment

Name *

E-mail *

Website

Submit

Previous post: FrameMaker 10 ExtendScript: The Object Model

Next post: Bypass: Part 1

- To search, type and

- Categories
  - ExtendScript (7)
  - Forms (1)
  - FrameScript (1)
  - Free Scripts (2)
  - Images (3)
  - Personal (2)
  - Tips and Tricks (3)
  - Word (1)
  - XML (2)
- Blogroll
  - FrameScript
  - MicroType
- Pages
  - About

WordPress Admin