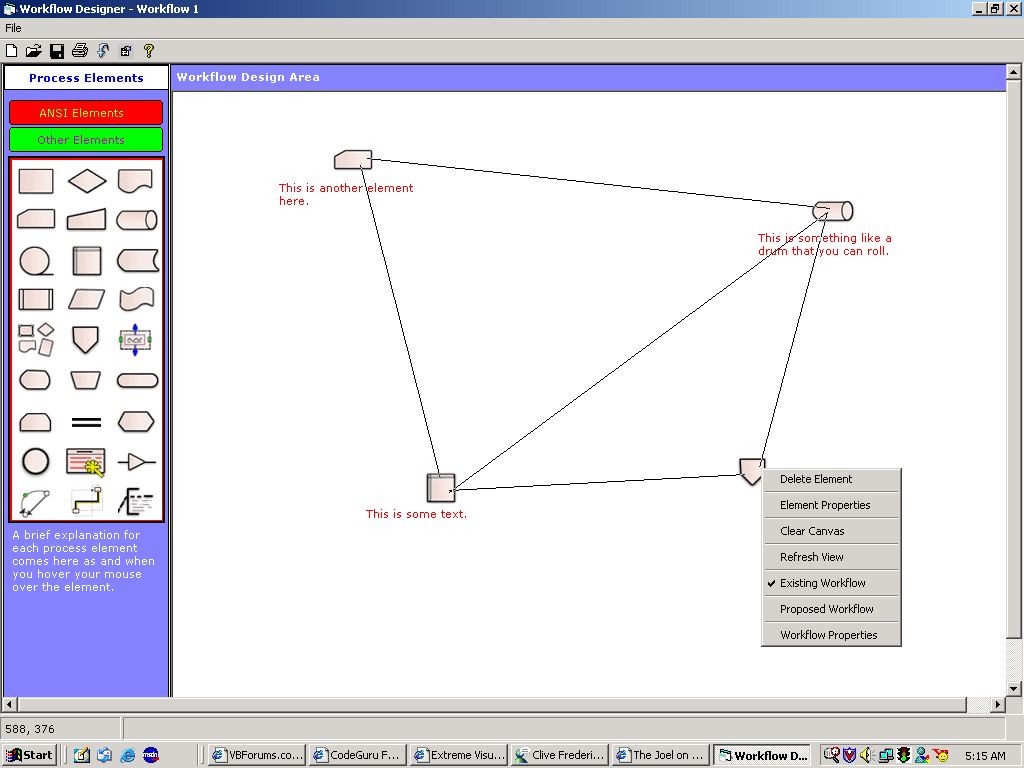
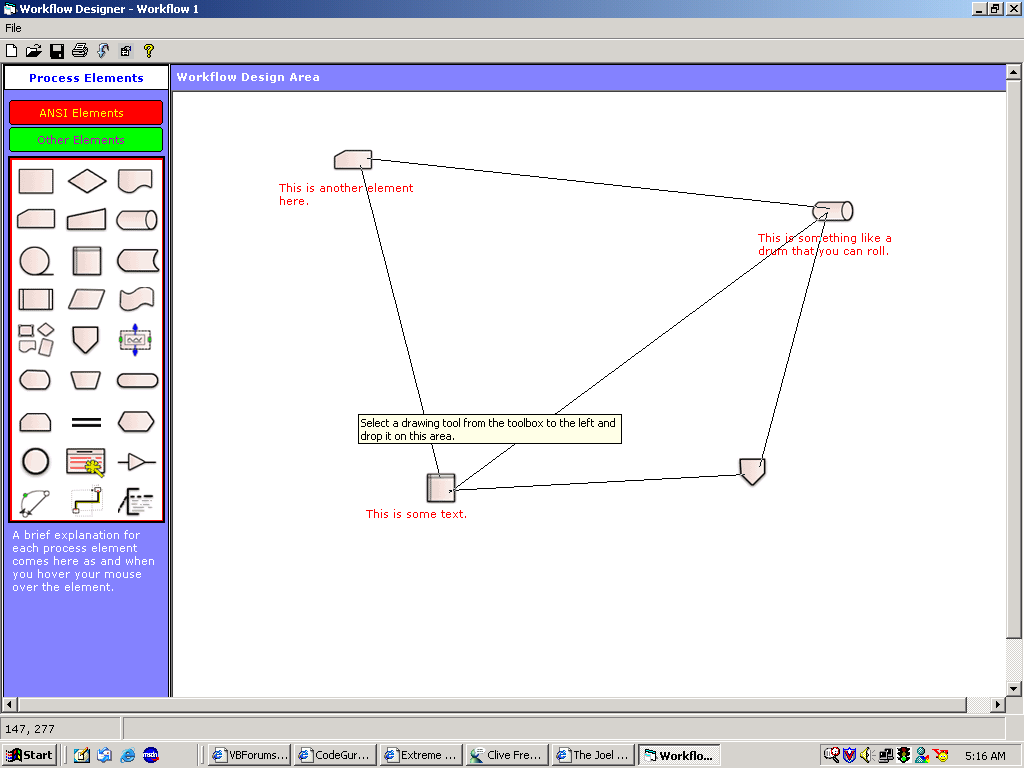
**Questions and Answers for Technical Writer Position with Auth0**

1. What’s your proudest achievement? It can be a personal project or something you’ve worked on professionally. Just a short paragraph is fine, but I’d love to know why you’re proud of it.

**Answer:** In 2003, without knowing anything about how difficult it would be, I agreed with my employer to make a software tool in 2 days that would allow a user to drag and drop items from a toolbox and connect them with lines and redraw those connections, etc.  
  
It took me a week or more and I suffered through it but also enjoyed it. I did it all using Win32 API with VB 6, mostly using Win32 raster operations and bitmap API for the drawing. It didn't have a lot of bells and whistles but it made me really proud.   
  




1. Tell me about an article/blog post/Github repo you read recently, why you liked it, and why I should read it. Also, link me to something you wrote recently that you really like

**Answer:**  I wanted to know what a monad was so I watched more than a dozen videos on YouTube and everywhere else about it, and read an equal number of articles. They added some but confused me some.

The two artefacts that helped me most was Brian Beckman’s video titled “Don’t fear the Monad” and Eric Lippert's series on monads and liked it because it spoke to me in the imperative programmer's tone and didn't have any mathematics in it. Brian Beckman hits the high notes. I just love that video. It is one of the most beautiful videos I have watched in a long time.  
  
<https://www.youtube.com/watch?v=ZhuHCtR3xq8>

<https://ericlippert.com/category/monads/>

I also enjoy reading Rx.NET source code because it sings, esp. the trampoline and the way actions are recursively scheduled.

<https://github.com/Reactive-Extensions/Rx.NET>

Another repo I like reading from is MoreLinq, which is an effort to add more operators to the IEnumerable<T> interface in .NET.  
  
<https://github.com/morelinq/MoreLINQ>   
  
I also love reading from the Linq Expression API in .NET because it makes a fantastic use of the Visitor design pattern to deconstruct LINQ expressions.

<http://referencesource.microsoft.com/#q=system.linq.expressions.expression>

And Lee Campbell’s Rx Cookbook is a very well-written book explaining Rx, which I really enjoy reading slowly and over and over again from time to time while I practice Rx, often making my own examples to test the hypothesis he presents in his book.  
  
<http://www.introtorx.com/content/v1.0.10621.0/01_WhyRx.html>

Two articles of my own that I enjoyed writing were:

1. Oh, the job titles these days: This is a satire on the meaningless job titles in the IT industry today.

<http://sathyaish.net/writing/jobtitles.aspx>

1. Famous Actors Who Never Went to Acting School

<http://sathyaish.net/writing/ActingSchool.aspx>

1. Tell me about one feature of Auth0 you really like, and why.  
     
   **Answer:** Auth0 is to OAuth what jQuery is to JavaScript. Take something really ugly and give users a very intuitive and easy to use API.

I say this from experience. I’ve read the OAuth specifications. All of them. OAuth 1, 1a, 2, the bearer tokens, thread model and security considerations, and I have practiced implementing OAuth clients for Facebook and Google using C#. It is ugly.

Even the Microsoft abstraction that I have been using, i.e. their Owin implementation Katana is horrendous.

And with raw HTTP calls, you have to remember all the parameters, have lots of redirects, then, de-authorization with Facebook sometimes never used to call my callback, implementations differ widely, some OAuth providers use Unix timestamps for refresh token validity, some use absolute date and time, some have OAuth 1a implementations (Twitter, and LinkedIn until recently had 1a implementation), and the variety is mind-boggling. With Auth0, you’ve made it simple and stupid.

1. Auth0 breathes a hacking culture. Hack something (whatever you want) using webtask.io. Here is an example of a cool hack: <https://auth0.com/blog/2015/07/28/if-this-then-node-dot-js-extending-ifttt-with-webtask-dot-io/>

**Answer:** I sometimes make fixed deposits with my bank. The bank allows me to specify the number of days I'd like to maintain the deposit for. It doesn't show me the maturity date of the deposit, though, until 2 hours after having made the deposit. This makes me want to just add up the number of days of the deposit to today's date to quickly get to know the maturity date.

Sometimes, I fire up Visual Studio to write this little function, and at other times, I fire up LINQPad. If I were a JavaScript developer, I'd fire up the Node REPL or the ScratchPad in Developer Tools that most modern browsers offer.

Here is a quick function that I made a Webtask out of that will help me add the number of days to a base date. The usage is:

url?b={yyyy-mm-dd}&n={numberOfDays}

where b is the base date to which the number of days is to be added:

<https://webtask.it.auth0.com/api/run/wt-sathyaish-gmail_com-0/addDays?webtask_no_cache=1>