# Site Insites

## Online Version

The final version formatted for the magazine on the IEEE site is available here.

Site Insights: Refreshing

By Bob Frankston

Periodically, I need to refresh my website. Once again, I'm rebuilding the site from the ground up—or at least, taking a fresh look at the software behind the site. In my April 2016 "Bits Versus Electrons" column, I wrote about building a simple Internet of Things device and shared what I'd learned. As part of the process, I implemented a simple web server from scratch with both server- and client-side code.

Without the need to produce a large-scale commercial website, I was free to take a very simple approach using Node (http://nodejs.org/). Starting from scratch can be very refreshing. It turned out that with today's tools, it was far easier to start with a blank slate than to use heavy-duty environments. Apache and Internet information services (IIS) are valuable, but they also come with attitudes and assumptions. This refresh is my fourth iteration of my website. The story starts back in the 1990s.

MY FIRST SITE

In 1994, Dan Bricklin and I recorded a demonstration of the relatively new World Wide Web (http://rmf.vc/WWWDemo) (Figure 1). Fortunately, Newton, Massachusetts, was one of the first communities to get cable modems. I already owned the domain name frankston.com, so I just set up a server on my desktop computer. The page is still there (Figure 2), with my face, preserved as if in amber—and many of the links still work! The fact that I'm surprised that links still work is a topic in its own right: Why do we use technologies like the domain name system, which make links expire by default?

Starting from scratch can be very refreshing.

In July 1998, inspired by others, I started to write posts. They were very simple, and I wasn't familiar with advanced concepts like proofreading. I did a number of other projects, which gave me a chance to do more than just post text. One of my more ambitious projects was a site for my son's elementary school. Undertaking these projects gave me the opportunity to experiment with techniques before they became the norm, as when I found that I could use JavaScript (JS) to generate hypertext markup language (HTML) on the fly and dynamically download extensible markup language (XML) data. That approach became formalized as XMLHTTP.

FIGURE 1. My 1994 demonstration of the World Wide Web.

Fortunately, even desktop operating systems had full-capability web servers (IIS on Windows and Apache on Linux), and they still do—which is a reminder of the peer nature of the Internet. The standard server-side scripting was (and still is) to intermix code with markup using <% %> or a similar notation. The standard design assumes that a uniform resource locator (URL) points to a file using the standard file system naming convention including the suffix, such as .html or .asp (for a scripted file). By default, a website is on port 80, which makes it awkward to have independent servers on a machine. Multiple sites on a given machine would normally be managed by the web server.

LEAVING HOME

Around 1999, to ensure continued availability, I moved my site to Interland (now web.com). The remote site was a clone of my desktop. It allowed me to continue to use my desktop to develop my site and then deploy it by simply copying files to the remote site. Microsoft's FrontPage and, later, Expression Web, made this easy. In 2006, I wanted to modernize my approach and separate code from presentation. I wrote my own tools, including one that converted Word files to simplified HTML. On the server side, I took advantage of the then-new capabilities in IIS to do a server-

Digital Object Identifier 10.1109/MCE.2016.2614551
Date of publication: 14 December 2016

122 IEEE CONSUMER ELECTRONICS MAGAZINE ^ JANUARY 2017

## Refreshing

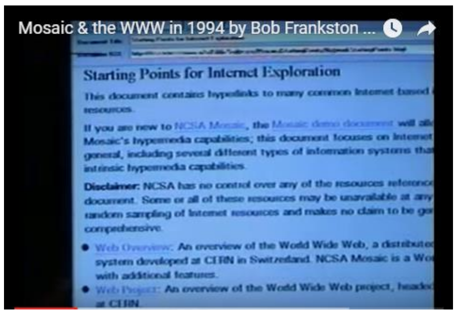
Periodically I need to refresh my website. Once again I'm rebuilding the site from the ground up or at least taking a fresh look at the software behind the site. In my April 2016 column I wrote about building a simple IoT device and shared what I'd learned in the process.

As part of the process I implemented a simple web server from scratch with both server side and client side code. Without the need to do a large-scale commercial website I was free to take a very simple approach using node. Starting from scratch can be very refreshing. It turned out that with today's tools it was far easier to start with a blank slate than to use heavy duty environments. Apache and IIS are valuable but they also come with attitudes and assumptions.

This refresh is my fourth iteration of my website. The story starts back in the 90's ...

## My First Site

In 1994 Dan Bricklin and I recorded this demonstration of the relatively new World Wide Web. Fortunately, Newton Massachusetts was one of the first communities to get cable modems. I already owned the domain name Frankston.com so I just setup a server on my desktop computer. The page is still there with my face preserved as if in amber and many of the links still work! The fact that I'm surprised that links still work is a topic in its own right – why do we use technologies like the DNS (Domain Name System) which make links expire by default?


Mosaic & the WWW in 1994 by Bob Frankston ...
Starting Points for Internet Exploration

In July of 1998, inspired by others, I started to write posts. It was very simple and I wasn't familiar with advanced concepts like proofreading. I did a number of other projects which gave me a chance to do more than just post text. One of the more ambitious was a site for my son's elementary school.

Doing these projects gave me an opportunity to experiment with technique before they become the norm as when I found I could use JavaScript to generate HTML on the fly and dynamically download XML data. That approach become formalized as XMLHTTP.

Fortunately, even desktop operating systems had full capability web servers (IIS on Windows and Apache on Linux). And they, still do which is a reminder of the peer nature of the Internet.

The standard server-side scripting was (and still is) to intermix code with markup using <% %> or a similar notation.

The standard design assumes that a URL points to a file using the standard file system naming convention including the suffix such as .HTML or .ASP (for a scripted file). By default, a website is on port 80 which makes it awkward to have independent servers on a machine. multiple sites on a given machine would normally be managed by the web server.

## Leaving Home

In about 1999 to assure availability I moved my site to Interland (now Web.com). The remote site was a clone of my desktop. It allowed me to continue to use my desktop to develop my site and then deploy it by simply copying files to the remote site. Microsoft's FrontPage and, later, Expression Web, made this easy.

In 2006 I wanted to modernize my approach and separate code from presentation. I wrote my own tools. I wrote a tool that converted Word files to simplified HTML. On the server side, I took advantage of the then new capabilities in IIS to do a server side program to read those files and present them at HTML.

One of the challenges I face in maintaining compatibility is that there are URLs pointing to my site that reside on other sites. I want them to continue to work. That meant maintaining the .asp files but changing them to redirect to the new names.

> (As an aside people forget that the problem of the passive observers who are invisible. We see this again in the excitement over "whitespace" in spectrum allocation. The flaw in that approach is that we don't know which listeners will get confused by the new uses of that portion of the spectrum).

I started to use http://rmf.vc for my own keyword system similar to http://bit.ly but under my own control. I implemented the keywords by redirecting the references to a "name=" parameter on the URL.

The new sophistication meant I had less control and found myself trying to work around the standard practices. I also want to update the look of my site. This led me to, once again, reinvent my site.

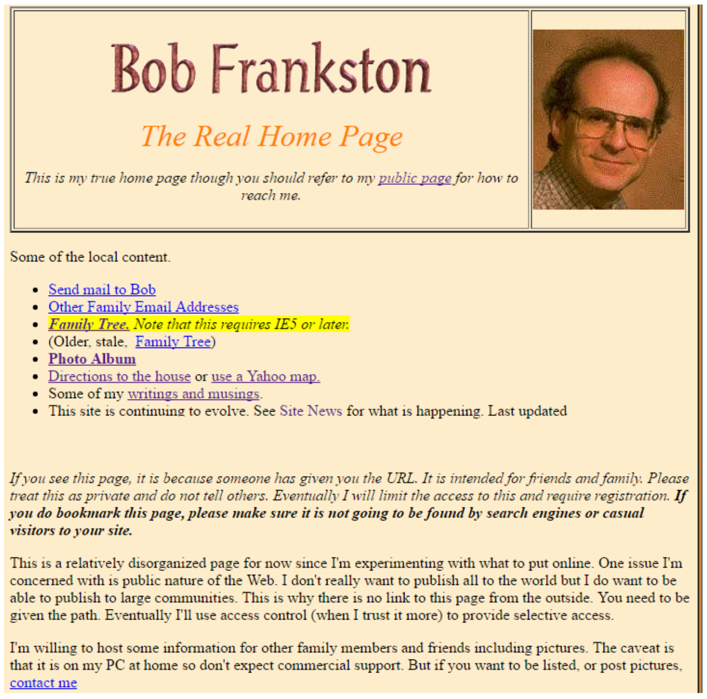
While it would be nice to take advantage of the features of sites like WordPress I would lose the ability of innovating on the server and would have to work within the WordPress conventions.

One point I made ien writing about my "blinky" project (April 2016 column) is that a web server itself can be fairly simple. I decided it was easier to write my own server than to work around favors from others.

The bonus is that I get to learn so much in the process and am still learning.

## Understanding the Web Protocols

For background it helps to understand the web protocols. They are relatively simple.



Bob Frankston

*The Real Home Page*

*This is my true home page though you should refer to my public page for how to reach me.*

Some of the local content.

- Send mail to Bob
- Other Family Email Addresses
- ***Family Tree.*** *Note that this requires IE5 or later.*
- (Older, stale, Family Tree)
- **Photo Album**
- Directions to the house or use a Yahoo map.
- Some of my writings and musings.
- This site is continuing to evolve. See Site News for what is happening. Last updated

*If you see this page, it is because someone has given you the URL. It is intended for friends and family. Please treat this as private and do not tell others. Eventually I will limit the access to this and require registration.* ***If you do bookmark this page, please make sure it is not going to be found by search engines or casual visitors to your site.***

This is a relatively disorganized page for now since I'm experimenting with what to put online. One issue I'm concerned with is public nature of the Web. I don't really want to publish all to the world but I do want to be able to publish to large communities. This is why there is no link to this page from the outside. You need to be given the path. Eventually I'll use access control (when I trust it more) to provide selective access.

I'm willing to host some information for other family members and friends including pictures. The caveat is that it is on my PC at home so don't expect commercial support. But if you want to be listed, or post pictures, contact me

The URL specifies which protocol to use (as in http) and then supplies parameters – the query string. You can go to https://en.wikipedia.org/wiki/Uniform_Resource_Identifier for more detail.

## http

**http** (or **https**) is just one of the possible protocols or schemes used. This specification of the scheme is a key part of the success of the web. If we only wanted to fetch files from a server, we could use ftp. http makes the information in the exchange explicit including having the server specify the type of processing (such as text/html)

In a sense what http does is add a level of indirection so I can spoof the earlier protocols. With http I'm still serving up a file as with ftp but the file can be created on the fly.

The request and response have both a header and a payload with the header describing the payload.

For an **http** request the header specifies the type of the payload (the body). It may be the contents of a form or it may be XML or JSON content. This is why it was easy to implement and JSON requests and APIs using standard protocols.

**http** also allows one to provide the name of the service. A single listener at an IP address can service multiple site names. This allows multiple IIS and Apache to host multiple sites at given IP address sharing a single port.

This is also why we've tolerated legacies of history such as the idea of having multiple applications sharing a single port 80. This made a lot of sense in the days of mainframes. A single mail server on a timesharing system would listen on port 25 and receive the mail on behalf all users.

A far better approach would be to associate names with software applications as end points rather than devices (computers). But that's a topic in its own right.

The protocol (http, ftp…) is implemented in the client on the user's computer. A browser is one possible client but not the only one. In Windows you can type "`start` `http://frankston.com`" which launches the browser or `start` "`mailto:IEEE1701@bob.ma`" which launches your mail program. Windows has embraced this notation for other purposes and, coincidently, it is similar to the file system. Linux has similar capabilities.

### HTML and web pages.

As with JavaScript, HTML has accumulated baggage as it has evolved. A modern site separates out the elements of presentation and code:

- **HTML** (HyperText Markup Language) itself – the <p>…</p> syntax. Today the main web page only has text and markup. The code and style are in separate files.
- **CSS** (Cascading Style Sheets) are now separated out into a separate file and putting markup directly into the HTML is considered bad form. CSS has its own heritage with a syntax more like JSON than XML.
- **JS** (JavaScript) is now typically in its own set of files with modules providing additional structure.

## A Fresh Start for my Site

### Platform Droppings

The first step was to choose a platform. I tried Microsoft's Azure and Amazon's web services. Since I am familiar with the Microsoft tool suite I went with Windows Server which is a restricted version of what I run on my desktop. In both hosting sites I got frustrated by the inability to connect to port 80. There were just too many firewall rules in the path second-guessing everything I do.

These firewall rules and other impediments in the path between the end points works against the edge-to-edge architecture of the Internet as I wrote about in my October 2015 column.

I found a simple alternative in DigitalOcean droplets which are small Ubuntu virtual machines. They start out with 20GB of storage at $5/month which is more than adequate for my modest site. They are so inexpensive that I got two of them so one could be my beta site and the other a production site. This is important since running on Ubuntu isn't exactly the same as running on the droplets. I still do most debugging on my desktop which is now my alpha test site.

The interface is my old friend the command line. I quickly got up to speed with what I needed for my applications:

- The latest version of **`node.js`** **(6)**
- **`PM2`** which is a useful program that keeps my app running across failures and reboots
- **`Emacs`** I would have a local editor
- **`Dropbox`** for deployment
- **`Python2`** for the tools
- **`LetsEncrypt`** for certificates.

Deployment is very simple. I created an additional Dropbox account and shared my "site" directory with it. Any changes I make automatically appear on the droplet sites and, thanks to PM2, if I change the beta or production JavaScript files the server program automatically restarts.

For safety I keep a separate file tree on my home machine for experimenting and also separate file trees for beta and production versions.

The sharing works both ways. I can write to a log file on my droplet and simply read it on my desktop. Having a separate Dropbox account provides me with a degree of isolation and safety.

Given how simple the droplets are I treat them as disposable if I need to experiment or just need another clean build.

### Choosing a Language and Environment

I'm used to using C# which has a mature library and advanced capabilities. It now even runs on Ubuntu. I may go back to using C# but for now I'm using JavaScript.

Actually I'm using TypeScript which is Microsoft's open source implementation of superset of TypeScript that includes type annotation and constructs that make it easier to write well-formed JavaScript.

TypeScript allows me to use Visual Studio on my desktop much like I use it for C#. The static type checking and hinting have been great boons in productivity and facilitate experimentation.

I use Node as my production environment. The NPM (Node Package Manager) gives me access to many thousands of packages though the primary capabilities for handling http are already baked into Node, While there are some features of C# I miss, I enjoy the freedom of JavaScript and its features. For example, using template strings. I can write:

```
html`<h1>${title}</h1>`
```

which inserts the value of the title variable in the string. The tag HTML is the name of a procedure which is given the values to insert. It can assure that even if title contains characters like "<" it will be properly quoted. It provides a clean way of generating HTML.

## The Site

The site itself is still a work-in-progress. The first step was to match the old site and achieve essentially the same look. 2006 vintage site was a mixture of clean code using CSS and whatever happened to work.

One of the motivations for redoing the site is the ability to use the URL as a resource rather than just a path into the file system. I no longer need to convert a keyword into a name=. In fact, I can do the reverse and normalize URLs to the format of http://rmf.vc/*keyword* even if the original link had a different syntax.

I now have the task of taking URLs that happened to work and keeping them working. Thus http://frankston.com/public/essays/leapseconds.asp used to go to an asp file that would redirect to the proper file. I now process such URLs and check to see if the file name (leapseconds) matches an existing keyword. The user will then see http://rmf.vc/LeapSeconds as the normalized form. This should also help with search engines.

One of the challenges has been taking what worked implicitly and making it work explicitly. This means trying to assure that the existing URLs work.

Conversely I'm tightening up the rules when I can. I'm trying to assure that keywords only work for http://rmf.vc and redirect most of my auxiliary sites to http://Frankston.com. I also support subsites: http://EleanorElkin.com for my wife's art and http://Connectivty.xyz for signing up for when (and if) I put together a site focused on connectivity. Each presented its own challenges and techniques.

Given how lightweight processes are, such sites can be implemented independently using separate droplets or using an http forwarder.

I use LetsEcrypt for certificates and need to complete that as well as assuring automatic renewal. A certificate doesn't guarantee that you can trust the site. It just says that the site you reached corresponds to the name on the certificate.

If the traffic becomes too heavy I can easily spin up additional droplets and can do the same for the subsites if I want very different behaviors.

## Beyond Websites

The ability to have scalable services on the net has spawned industries around other approaches. Rather than virtual machines these often use containers which are really just timesharing processes. Both VMs and processes go back to the 1960's.

Many of the "sites" aren't really sites at all but provide services which may be used as middleware by other sites or as services consumed by apps via JSON APIs. The apps may be traditional applications or, increasingly, HTML5 applications.

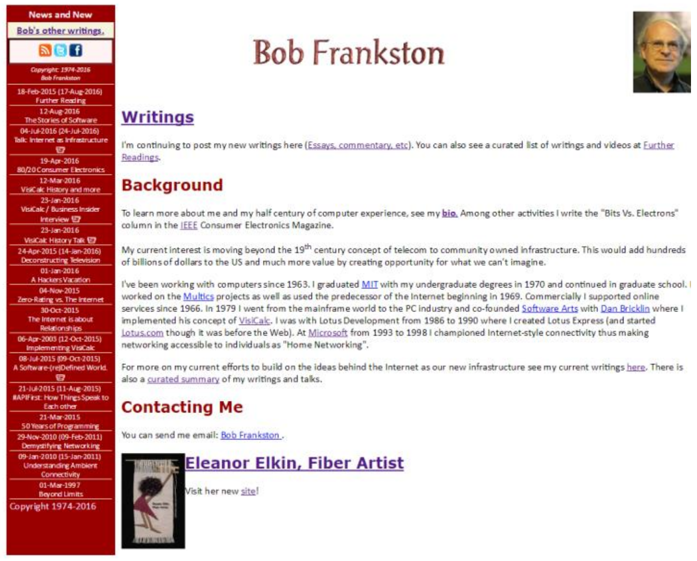
While these trends have lots of advantages there is a danger in complexity building up as sites depend on each other and on libraries such as JQuery. This is another reason I've embraced TypeScript with HTML5 – it's easy to develop apps with minimal dependence on large libraries.

In creating a simple site, I didn't need to deal with complex issues of scalability, authentication and many other issues. And that is part of the point. By keeping things simple I am back in control. Periodically returning to simplicity is very healthy.

To be clear, this isn't simplicity in the sense of simple but rather appropriate architectures. It's a chance to refactor applications and discover new effective architecture moving forward and making the implicit choices of the past explicit.

## Software

This column is bits vs. electrons. In a sense writing about the website is akin to writing about a hardware project. Software + connectivity gives me a chance to not only tinker but to share that tinkering with the world in a pragmatic way. If you do visit http://frankston.com it may look very different how it looks at the time I wrote this column. It is a work-in-progress and also a chance to experiment with new approaches.



News and New

Bob's other writings.

Copyright: 1974-2016
Bob Frankston

18-Feb-2015 (17-Aug-2016)
Further Reading

12-Aug-2016
The Stories of Software

04-Jul-2016 (24-Jul-2016)
Talk: Internet as Infrastructure

19-Apr-2016
80/20 Consumer Electronics

12-Mar-2016
VisiCalc History and more

23-Jan-2016
VisiCalc / Business Insider Interview

23-Jan-2016
VisiCalc History Talk

24-Apr-2015 (14-Jan-2016)
Deconstructing Television

01-Jan-2016
A Hackers Vacation

04-Nov-2015
Zero-Rating vs. The Internet

30-Oct-2015
The Internet is about Relationships

06-Apr-2003 (12-Oct-2015)
Implementing VisiCalc

08-Jul-2015 (09-Oct-2015)
A Software-(re)Defined World.

21-Jul-2015 (11-Aug-2015)
#APIFirst: How Things Speak to Each other

21-Mar-2015
50 Years of Programming

29-Nov-2010 (09-Feb-2011)
Demystifying Networking

09-Jan-2010 (15-Jan-2011)
Understanding Ambient Connectivity

01-Mar-1997
Beyond Limits

Copyright 1974-2016

Bob Frankston

### Writings

I'm continuing to post my new writings here (Essays, commentary, etc). You can also see a curated list of writings and videos at Further Readings.

### Background

To learn more about me and my half century of computer experience, see my **bio.** Among other activities I write the "Bits Vs. Electrons" column in the IEEE Consumer Electronics Magazine.

My current interest is moving beyond the 19th century concept of telecom to community owned infrastructure. This would add hundreds of billions of dollars to the US and much more value by creating opportunity for what we can't imagine.

I've been working with computers since 1963. I graduated MIT with my undergraduate degrees in 1970 and continued in graduate school. I worked on the Multics projects as well as used the predecessor of the Internet beginning in 1969. Commercially I supported online services since 1966. In 1979 I went from the mainframe world to the PC industry and co-founded Software Arts with Dan Bricklin where I implemented his concept of VisiCalc. I was with Lotus Development from 1986 to 1990 where I created Lotus Express (and started Lotus.com though it was before the Web). At Microsoft from 1993 to 1998 I championed Internet-style connectivity thus making networking accessible to individuals as "Home Networking".

For more on my current efforts to build on the ideas behind the Internet as our new infrastructure see my current writings here. There is also a curated summary of my writings and talks.

### Contacting Me

You can send me email: Bob Frankston .

### Eleanor Elkin, Fiber Artist

Visit her new site!

## Postscript

After I completed this project I revisited the "Blinky" project I wrote about in my April 2016 column (http://rmf.vc/IEEEBlinky). I was able to quickly port it to the Raspberry pi Linux system while retaining my ability to do remote debugging. It's a reminder that the benefits of the project in what I learn as much, if not, more than narrow goals of the project.