VISTA Community Meeting
Day Three: January 16, 2011
Afternoon Session

Topic: Integrating Legacy M Code into the Web (without touching the M code!)

Presenter: Rob Tweed, M/Gateway

Mr. Tweed began by noting that he had mentioned in passing a few times before a new technology called Node.js. It's the secret to how the EWD group at the conference has been able to get that VT emulator running in a browser.

The idea of getting VISTA into a browser has been around for years. VISTA has too much legacy code to rewrite in a timely or cost-effective way. The legacy code works well; it just looks old-fashioned.

This has been difficult in the past because the HTTP protocol is a request/response protocol. A response cannot be sent without a request. Developers generally end up with polling mechanisms, which will either tie up resources or result in stuttery programs. VISTA needs to be able to write stuff to the UI when it wants to.

Node.js is a server environment, single-threaded for simplicity and scalability. There is a single event loop. It uses asynchronous, non-blocking I/O. Everything is event-driven, and this is how browsers and Javascript work.

Node.js can fill many roles at once: web server, web client, database client, web socket server. It can invoke child processes (directly, through telnet, or through ssh). Because it is event-driven there are no delays or polling overheads.

Web sockets are part of the HTML5 standard. They allow a direct duplex socket between the browser and the back-end, tunneled through the standard HTTP/HTTPS connection. And it's all event-driven.

Using Node.js, a browser can talk to MUMPS code, and the code thinks it's talking to a regular dumb terminal.

Mr. Tweed used a diagram to demonstrate how strings from M could be parsed in Node.js before being passed to the browser. Conversely, input from the browser could be normalized in Node.js before being passed back to M.

Companies like Yahoo are starting to realize how powerful this could be, in expanding support to older browsers.

Canvas is a technology in HTML5 that allows a browser to build real-time graphical displays. For example, a live EKG could be sent directly to a handheld. Canvas displays can be mixed and matched with the other stuff [EWD].

Q: I like the idea of moving the data to a device. That's the kind of thing that's usually in an HL7

message, and it would be great to be able to see it in a browser.
A: Yes, it's very powerful technology.

Q: This is all over encrypted, secure connections?
A: Correct. Node.js can run behind a firewall. Communications can be done over SSL.

Q: Is there a preferred browser?
A: No.

Q: Most Javascript seems very convoluted and disorganized.
A: That's usually the result of the brain that wrote it. Well-constructed Javascript is easier to follow. And the beauty is there's usually not much of it.

Q: What supports HTML5?
A: Safari, Firefox, Chrome/Chromium browsers, Android, Blackberry, iPhone, iPad, etc. Most people in the node community use library modules called socket I/O. So, in theory, this could get the dreaded IE6.

Q: I assume at some level we're wrapping selects on sockets.
A: Yes. I've set up demos in the UK for people to test in Chicago. And they said it was as if the server was right there. It's pretty damn fast.

Q: This is all open-source?
A: All open-source. There's never been a more exciting time to be in IT. All these technologies have come together at just the right time.

Q: It's fascinating; we'll discover more ways these tools will help us. We're not done here.
A: Absolutely; we're just scratching the surface. A lot of the tasks we've got now is to start thinking laterally. It's a really cool time.

Q: In the diagram, that child process could be Caché or GT.M with M running inside it?
A: Yes
Q: I'm thinking of Western State Hospital, which is stuck using Caché on Windows. This looks like you could get around the CSP, with Node.js running on your server.
A: Yes. Absolutely.

Mr. Tweed noted that NOSQL options could make this process even faster.