Ideaburst
2040 W. Middlefield Rd., Suite 2
Mountain View, CA 94043
Phone: 650-564-0050
Fax: 650-564-9950

W3C Web Applications Workshop 2004 Position Paper

Corporate Background

Ideaburst, Inc. has been developing Web applications since 1997. In 2000, we started working with Adobe Systems, Inc. to create SVG-based applications to generate excitement about the emerging SVG standard. A few of those applications became well known and are still some of the leading examples of the power and flexibility of SVG. Those applications included an online ticket reservation system; a visual phonebook for locating and accessing information on employees and rooms; and a free-form drawing application. Recently, Ideaburst, Inc. has been developing an SVG-based application framework and a set of GUI widgets to allow us to build a highly graphical facilities management system. In short, due to the length and breadth of our experience developing Web and SVG-based applications, we feel the upcoming workshop can greatly benefit from our input.

Goals

The main goal of our position paper is to express what we feel is necessary to facilitate the creation of future and/or next generation Web-based applications (webapps). To begin, we believe if SVG and its runtime environments were extended with application development in mind, it would become a perfect language with which to develop powerful and highly distributed webapps. Since it was unclear as to whether the W3C was thinking about simply expanding SVG or creating a new and separate set of standards, we decided to include much of the functionality already available to us through SVG. So please forgive us if we state the obvious or recount things that have been rehashed for years.

The Runtime Environment

To us, webapps are like any other type of computer application. In order to run them, there needs to be a solid and supported runtime environment that adheres to a set of standard or published APIs. Ideally, this runtime environment would operate on any device capable of browsing the Web. This would be a challenge on some of today's smaller devices such as cell phones, but as CPU power and memory increases for these tiny devices, it seems logical that people will come to expect these devices to run the same types of applications they can run on their computers. This would also help give a significant advantage to Web-based applications as they would be usable from any device with a network connection.

That being said, the runtime environment should have a rich vector-based graphical interface with standard text and GUI widgets. This would facilitate the interaction between the user and the application. The runtime environment should also be adaptable enough to handle input from a wide array of devices such as keyboards, mice, touch screens, pens, and even voice commands. Without a standard set of interfaces, developers would be forced to code their own widgets which would take an exceptional amount of time and effort. Furthermore, a lack of standard interfaces and behaviors would result in applications that behave in unexpected ways thereby reducing usability and adoption rates. The success of GUI-based platforms such as Windows demonstrate how having a standard set of interfaces allows for faster application development and make computer applications easy to use by most people. It also demonstrates standard interfaces can take years to develop and evolve.

The Runtime Environment - continued

In addition to standard interfaces and widgets, the runtime environment should provide developers with dynamic layouts in the form of a layout manager or HTML-like tables. Unlike HTML-content or downloadable graphics which tend to be static, webapps are always changing based on user interaction and downloaded data. Without simple tables, developers would need to code mechanisms to calculate the exact positions of each and every piece of data. In other words, displaying 100+ rows from a database query could be a calculation nightmare. If the runtime environment supported these types of layouts, the application would run faster and the developers would have a much easier time displaying data.

Of course, more is needed than the ability to simply display application interfaces. In order to enable the application to do anything, the runtime environment needs to be able to run a standard programming or scripting language. This would enable a high degree of user interaction, cut down on the amount of communication between the client and the server and make the webapps faster, more responsive, and avoid tying up expensive bandwidth and server CPU cycles.

And if you can write code for an application, there should be a way of protecting that code against theft and misuse. To help with this, the runtime environment should support some mechanism to prevent intellectual property from being lost.

Furthermore, if webapps become intelligent by being able to run their own logic on a local client, the runtime environment will need to support two-way communication with one or more servers. That way, the application can get and send data without losing its state by being forced to reload. Additionally, it would be nice if the runtime environment would compress and encrypt such communications to protect user data and content. And lastly, having the option to maintain data connections could be highly beneficial especially when the server needs to contact the application.

Yet communication with server systems is not enough to make webapps ubiquitous. Most of today's modern applications and web sites are able to access and save information to the local client filesystem. This type of mechanism is critical for saving application and user defaults, preferences, and state.

And speaking of states, the runtime environment should support methods to allow an application to determine whether its resources have been downloaded, rendered, and/or initialized. Otherwise, timing and an assortment of other errors could occur due to the application's resources not being ready for use.

Lastly, all of this needs to be wrapped up in a small, downloadable package that is easy to distribute and install. Performance is also a key issue. If the runtime environment is too difficult to obtain or runs applications too slowly, developers and users will dismiss the technology as being too immature to use.

Web Application APIs

In addition to the items listed under the Runtime Environment section of this document, we believe Web applications should also have the following APIs:

- The ability to parse XML
- Upload and download data to and from the client filesystem
- The ability to open new windows, populate those windows with content, and establish bi-directional communication between the windows
- Dynamically link to and display new content/documents
- Bi-directional communication with a parent HTML document and the browser displaying that document
- Load status indicators

Standard User Interface Controls

Microsoft developed Windows and we are talking about next generation Web applications because graphical user interfaces are more intuitive and easy to use than their text-based counterparts. Also, Windows and HTML continue to be popular because they provide a standard set of interfaces people can use to develop and deliver applications and content. Having standard interface controls will also promote the creation of development tools which will greatly facilitate the development of webapps. One only has to look at the proliferation of Web sites after the first GUI-based HTML development tools were released.

Expecting developers to create webapps without a standard set of user interface controls is asking a lot. Not only would you need the expertise required to create your application logic, but you would also need to understand how user interfaces work in order to design usable and reasonably performing controls. There's also the sheer size of the task required. We've been developing these types of widgets for two years and are only now getting ready to release our latest application. If these types of interfaces had been available to us, we could have released this application within a few months.

As to the interface controls sharing the look and feel of their native platform, one must ponder why a Web-based application should look identical to a local application. Wouldn't that be confusing to a user? What happens if they forget and disconnect from the network? Web users are accustomed to every site looking a little different. As long as there is a default look to the standard user interface controls, leave it to the developers to decide if they want to skin their applications for a variety of platforms. If the interface set were controllable by something like CSS, skins would be easy to develop and share and thereby solve the matter for those who think such things are an issue.

Declarative Statements vs. Scripting

Our experience developing SVG-based applications has led us to believe scripting is more flexible than declarative syntax and provides more control over an application's behavior. Declarative statements can often be restrictive and limited whereas programmatic code can easily be added to enhance an application's functionality. Unless there was an easy way to add declarative statements, their functionality will always lag behind the development curve.

Security Issues

As previously stated, we strongly believe there must be some mechanism of protecting a webapp and its resources from theft and misuse, especially since these applications may be broadcast over the Internet. Without such protection, companies will be leery of spending time and money to develop an application that can easily be stolen, copied, and modified once it is released. One solution to this would be to encrypt the application and its resources and have the runtime environment decrypt them without saving any plaintext files to the local client. Additionally, if such a security feature were used, the runtime client should not, in any way shape or form, expose or allow those resources to be copied by a user. If more security were desired, perhaps the runtime environment could require a password in order to decrypt the application to make it more difficult for someone to intercept and decrypt the application.

Web Application Resource Packaging

Having the ability to download a webapp and all of its resources through a single communication with the server seems like a nice and logical idea. Packaging would reduce network traffic and server CPU cycles and guarantee the application had all of its initial resources. Our requirements for such packaging would be:

- Easy to create and modify
- Secure against theft and misuse
- Compressed to minimize download times
- Accessible in whole or in part by the application
- Have methods or checksums to ensure the package arrived uncorrupted
- Would not prevent other resources from being dynamically downloaded

Yet downloading a webapp's core resources in a package does not solve the more critical problem of knowing when resources are ready for use. In order for a webapp to initialize itself or its resources, it needs to have a mechanism to check if resources have completed downloading (whether they be packaged or not), have been rendered, and whether any of those resources have run and completed their own initialization processes. Again, webapps are highly dynamic in nature and will need these types of mechanisms to avoid timing errors especially since they will run on a wide array of platforms and runtime environments.

Closing Comments

We feel SVG satisfies a good portion of what is necessary to create next generation Web applications. However, the following items need to be addressed to facilitate the creation and adoption of these kinds of applications:

- A hearty, well-supported runtime environment that runs on all platforms and devices
- A standard set of interfaces and GUI widgets
- A layout manager and/or table layouts
- Methods of securing content, resources, and code
- Ways of checking ready states for downloaded application resources
- The ability to access and save data to the local filesystem
- The means of opening new windows with bi-directional communication
- And of course, developer tools to help create and debug these applications