Yahoo! Application Platform Developers Guide

Yahoo! Application Platform Developers Guide

Abstract

This guide provides overview and conceptual information for the Yahoo! Application Platform. It is intended for software developers who are familiar with technologies such as HTML, JavaScript, PHP, and web services.

Looking for more docs? See the Y!OS Documentation[1] landing page.

For the latest information, see the Yahoo! Application Platform Release Notes[2].

We welcome your feedback. Have a comment or question about this document? Let us know in the YDN Forum for Y!OS Documentation[3].

[1] /yos
[2] /yap/releasenotes/
[3] http://developer.yahoo.net/forum/index.php?showforum=64

Table of Contents

List of Figures

List of Tables

Chapter 1. Overview of the Yahoo! Application Platform (YAP)

What is YAP?

The Yahoo! Application Platform (YAP) is the software and services that enable developers to build Web applications that are available throughout Yahoo!-- the largest audience in the world. The Yahoo! Application Platform has the following components:

- Development environment: A browser-based tool that enables software developers to quickly create, preview, and publish Web applications.
- APIs and Web services: Programmatic access to OpenSocial functionality and popular Yahoo! Web services.
- Distribution and discovery infrastructure - The built-in features for publishing applications on galleries on Web pages such as Yahoo! Profiles. End users can discover applications by searching or browsing within application galleries.
- Runtime and rendering environment: The backend servers and software that run applications and convert the code into HTML.

How Do I Get Started?

To get started quickly with YAP, follow the step-by-step instructions at:

- Installing the PHP SDK for YSP[1]
- Creating an Open Application[2]

If you want to write a Flash application, see the following:

- Building Yahoo! Social Applications with Flash[3]
- Y!OS Flash ActionScript 3 API Reference[4]

Programming Models

YAP offers several programming models for the development of Open Applications [6]. The diagrams that follow show a simplified view of the runtime components for each model.

Server-Side

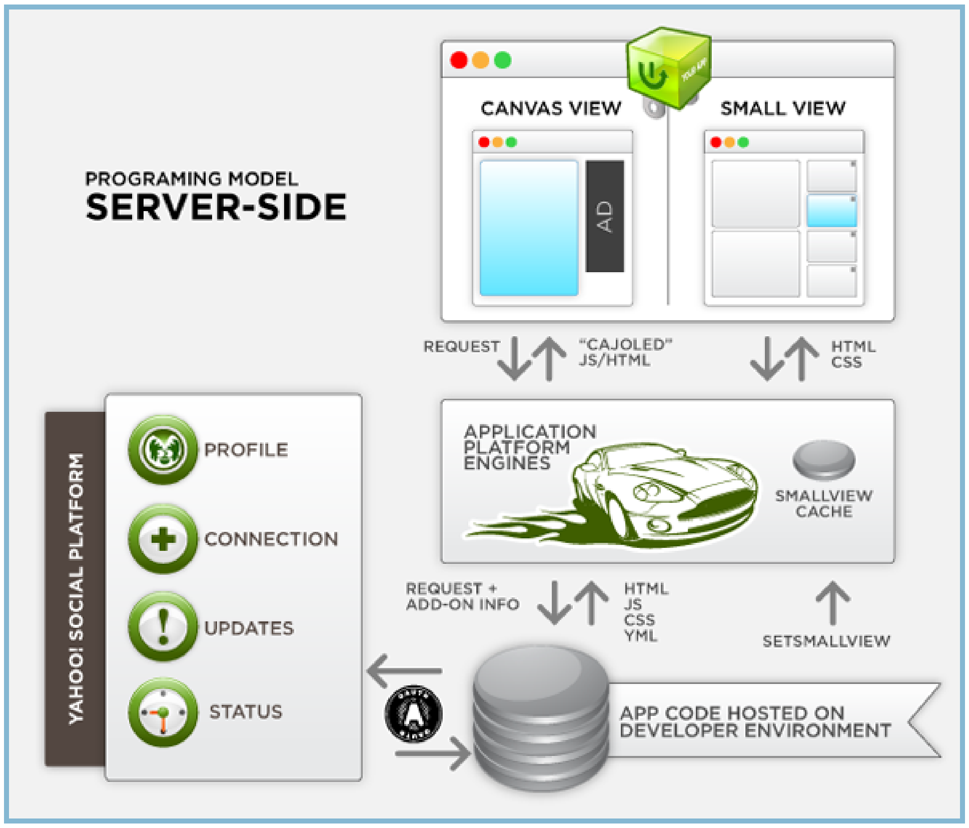
In this model, the code for your Canvas view [7] is a Web application that is hosted by and runs on your servers. You can write the code in the language of your choice, such as Python, Java, or PHP. To make it

[1] ../../yos/tutorials/installing_php_sdk.html
[2] ../../yos/tutorials/creating_open_app.html
[3] ../../flash/yos/
[4] ../../flash/yos/classreference/

easier to perform authorization and access the YSP APIs, YAP provides the Social API PHP SDK[5]. At runtime, the YAP engine proxies requests to your server, adding the additional information listed in Parameters Passed to an Open Application [32]. The Canvas view of your application can access this additional information programmatically. For example, a Canvas view coded in PHP can access the user's GUID from the `$_REQUEST` superglobal. YAP saves the Small view [6] code in a cache. To specify the default Small view code, you enter the HTML and YML statements on the Application Definition tab of the Application Editor. To personalize the Small view to each user, your application can call the `setSmallView` method of the PHP SDK.

YAP takes several measures to protect the user's private data. When the application accesses data through the Yahoo! Social Platform (YSP) API, OAuth verifies that the access is authorized. The YAP engine sanitizes the HTML and processes the JavaScript with Caja [16] before sending the content back to the browser.

Figure 1.1. Server-Side

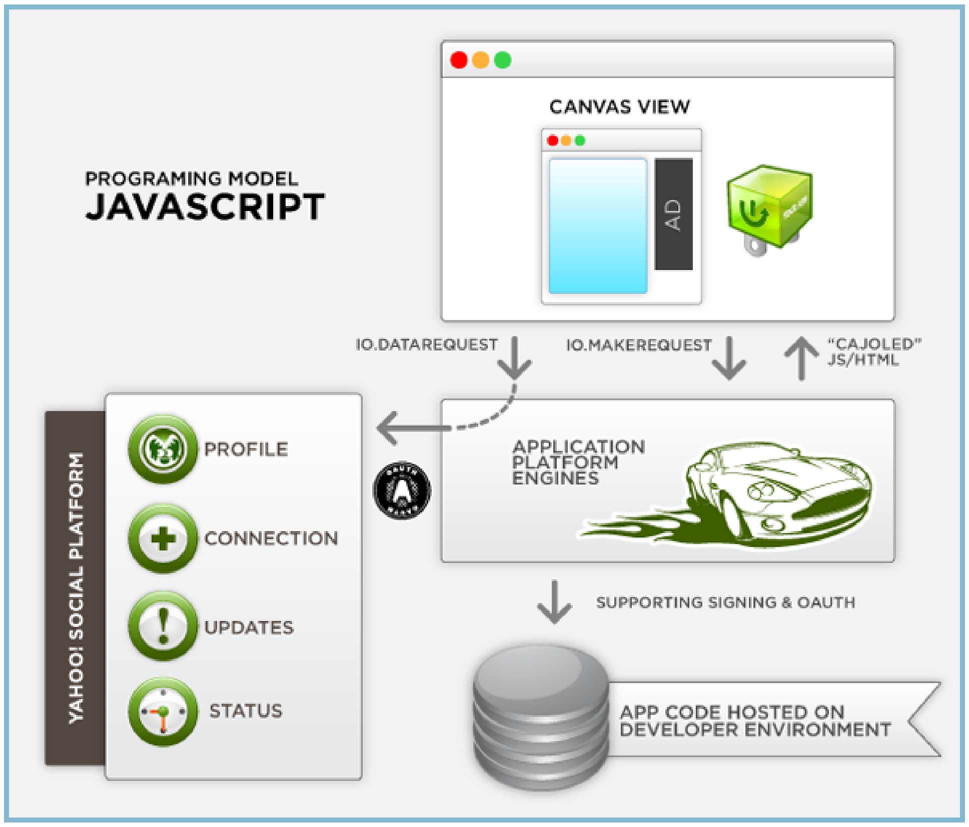


Browser-Side OpenSocial JavaScript

Most OpenSocial applications are written in JavaScript, which runs on the browser. When a Canvas view makes an OpenSocial `io.dataRequest`, the YAP engine retrieves the data by calling the corresponding YSP API. For a `io.makeRequest`, the YAP engine fetches content from a third-party site, sanitizes the HTML, and then sends the content back to the browser.

[5] ../../social/sdk/index.html

Figure 1.2. Browser-Side JavaScript



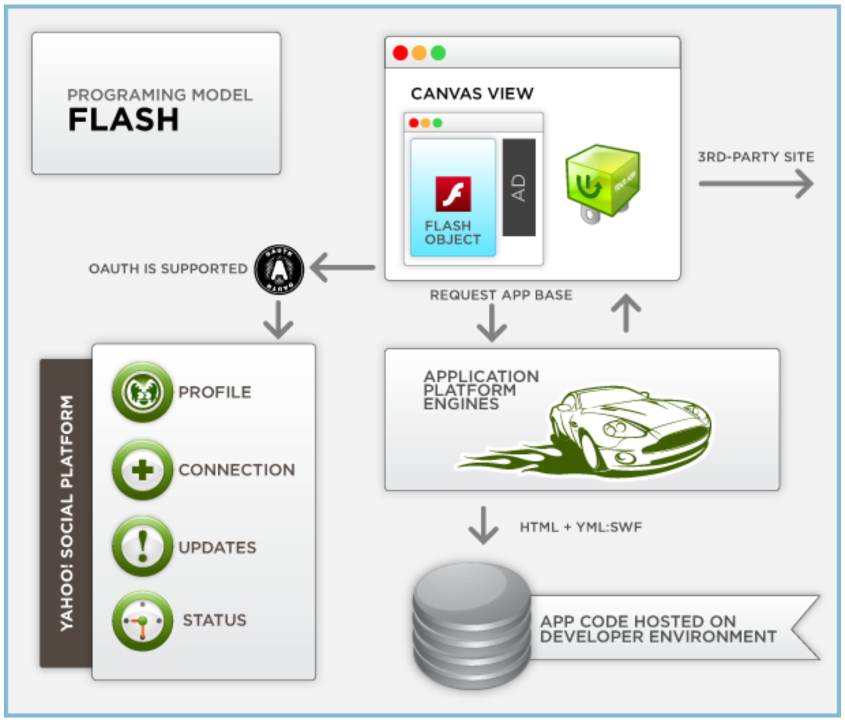
Browser-Side Flash

With the ActionScript 3 Social APIs[6], you can create Flash modules in a Canvas view. The Flash module has access to the viewer's session information and can obtain social information by making calls to the YSP APIs. To include a Flash Module in the Canvas view, insert a Yahoo! Markup Language[7] (YML) tag such as the following:

[6] ../../flash/yos/
[7] ../../yap/yml/

```
<yml:swf src="http://example.com/app.swf" width="780" height="1000"/>
```

Figure 1.3. Browser-Side Flash



Open Application Workflow

The following diagram shows the overall workflow for creating and deploying an Open Application [6] on YAP. During this process, an Open Application progresses through the following stages:

- Development: In this stage, you design and code the Canvas [7] and Small [6] views of your application. Within the Application Editor, you can check the views with the preview feature. Nobody else can view or run the application. For step-by-step instructions on developing an Open Application and running the Application Editor, see the tutorial, Creating an Open Application[8]. The tutorial also explains how to "push live" and publish an application.
- Pushed Live: The application is runnable on YAP. To share the application with others, you can send them the proxied URL. Typically, in this stage you are still testing the application with a select group of users. The application does not appear in the Yahoo! Application Gallery.
- Published: The application has been published in the Yahoo! Application Gallery. To find your application in the gallery, end users search by tag or browse through categories.
- Installed: An end user has installed the application in a landing page such as Yahoo! Profiles. The Small view of the application appears in the landing page. From the Small view, the user can launch the Canvas view.

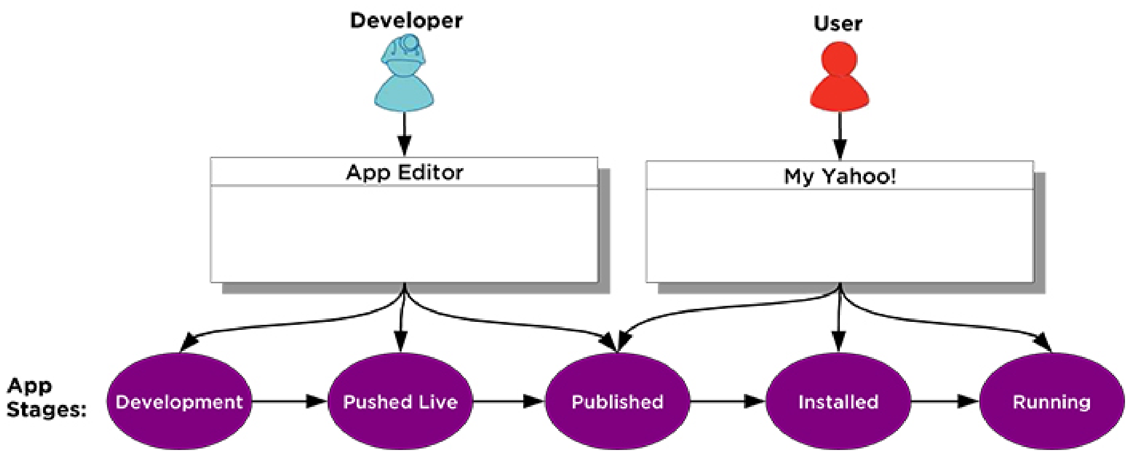
[8] ../../yos/tutorials/creating_open_app.html

- Running: The end user is running the Small or Canvas view of the application. At runtime, the YAP engine renders the code that appears in the end user's browser. For more information on the runtime process, see the section, Programming Models [1].

Note

The published, installed, and running stages are not available in v1.0 of YAP. However, you can test the running of your application by previewing it in the Application Editor.

Figure 1.4. Application Workflow



Encoding Requirements

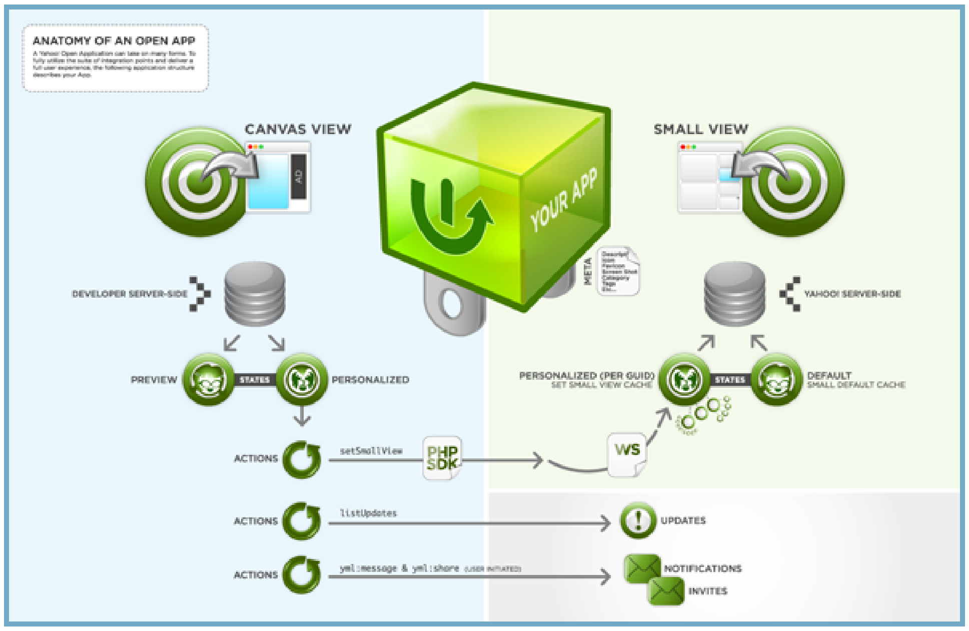
The character encoding for YAP is UTF-8 for both requests and responses. Invalid input will be rejected.

Chapter 2. Anatomy of an Open Application

Introduction

An Open Application is a Web application that has been registered on the Yahoo! Development Network[1] (YDN) and runs on the Yahoo! Application Platform (YAP). As seen by the end user, an Open Application has multiple views, integration points, and components.

Figure 2.1. Anatomy of an Open Application details the composition of an app with its different Views, states, and locations.



Click to see a larger image[2] or PDF[3] of this figure.

Small View

The Small view of an application appears to end users as a module contained within a Web page. As the application's teaser, the purpose of the Small view is to draw end users into the Canvas view [7], which provides a richer interaction.

The content of the Small view is up to you, but usually a Small view contains information such as the following:

- One or more triggers to launch the Canvas view.

[1] http://developer.yahoo.com
[2] images/anatomy_big.png
[3] images/anatomy_big.pdf

- A representation of the user, such as a name and an image. Your application can get this information from the user's Yahoo! Profile.
- Status or updates about other end users that will encourage the user running the Small view to open the Canvas view.

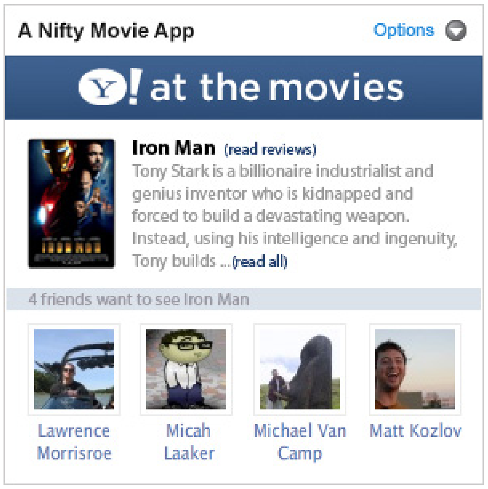
A Small view has two states:

- Default: To define the code for the default state, in the Application Editor, enter HTML or YML in the Small View Default Content field. Other markup or programming languages are ignored. This code is stored on a server within YAP and is the same for all end users. At runtime, YAP renders the Small view's default code if the personalized state for the end user is not available.
- Personalized: To personalize the Small view, an application calls the setSmallView method of the YSP PHP SDK. This method enables you to tailor the Small view for each end user (identified by GUID) and to dynamically update the Small view. You can use this method wherever you can run the YSP PHP SDK, for example, in the Canvas view and back-end processes.

Small views have the following constraints:

- The size of a Small view depends on its host Web page, but typically the size is 300 px wide by 250 px high (with fluid height).
- Unlike the Canvas view, advertisements and promotions cannot be served in a Small view.
- In the Small view, the code is restricted to HTML or YML[4]. You cannot specify JavaScript, PHP, or other languages for the Small view.

Figure 2.2. Example Small View



Iron Man (read reviews)
Tony Stark is a billionaire industrialist and genius inventor who is kidnapped and forced to build a devastating weapon. Instead, using his intelligence and ingenuity, Tony builds ...(read all)

Canvas View

The Canvas view is the application's largest and richest interface for the end user. Usually, the Canvas view is rendered within a Yahoo! landing page, surrounded by a Yahoo! header, footer, and an advertisement unit. End users access a Canvas view from a variety of locations: the Small view, links for user updates,

[4] http://developer.yahoo.com/yap/yml/

email links, and Web links. Unlike the Small view, the Canvas view supports third-party advertisements and promotions. The size of a Canvas view is 760 px wide with an infinite height.

The application code for the Canvas view is hosted on a server outside of YAP. To specify the code's location, in the Application Editor, enter the URL in the Application URL field.

The Canvas view code can be HTML, YML, CSS, and the subset of JavaScript allowed by Caja. This code can be generated by a variety of languages, such as PHP, XML/XSLT, and Ruby on Rails.

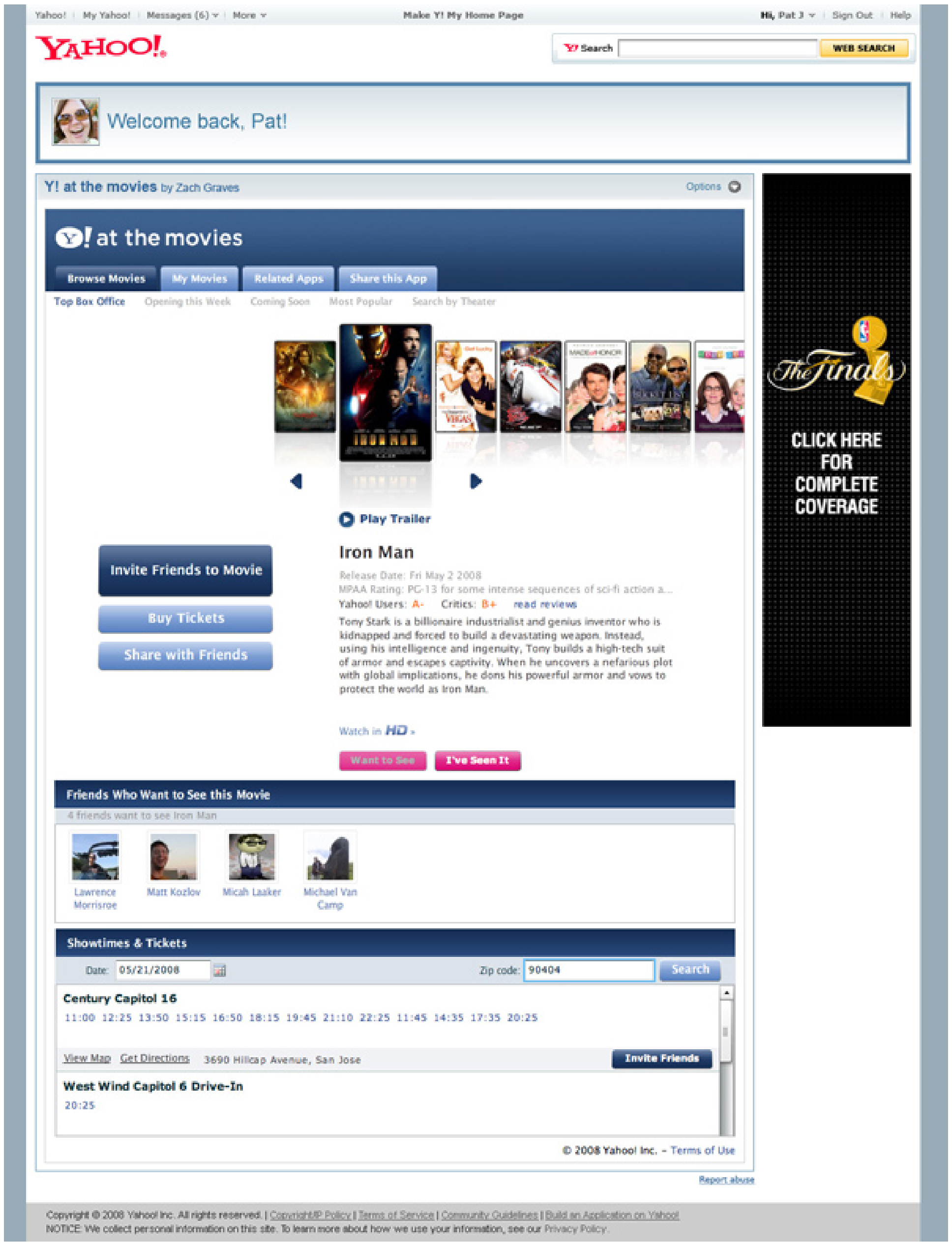
To protect the private information of end users, YAP applies the following security mechanisms to the Canvas view:

- The Canvas view is presented within an iframe.
- HTML is sanitized to remove unsafe code.
- JavaScript is translated by Caja.

You should design the Canvas view so that it has the following two states:

- Uninstalled: The end user is visiting your application but has not installed it. In this state, the end user has not logged into Yahoo!, so his or her identity (GUID) is unknown to the application. In this state, the Canvas view should encourage the end user to install the application.
- Personalized: The end user has installed your application, his or her identity is known, and the full functionality of the Canvas view is available.

Figure 2.3. Example Canvas View



Welcome back, Pat!

Y! at the movies by Zach Graves

Iron Man

Release Date: Fri May 2 2008
MPAA Rating: PG-13 for some intense sequences of sci-fi action a...
Yahoo! Users: A- Critics: B+ read reviews

Tony Stark is a billionaire industrialist and genius inventor who is kidnapped and forced to build a devastating weapon. Instead, using his intelligence and ingenuity, Tony builds a high-tech suit of armor and escapes captivity. When he uncovers a nefarious plot with global implications, he dons his powerful armor and vows to protect the world as Iron Man.

Friends Who Want to See this Movie

4 friends want to see Iron Man

Lawrence Morrisroe
Matt Kozlov
Micah Laaker
Michael Van Camp

Showtimes & Tickets

Date: 05/21/2008 Zip code: 90404

Century Capitol 16
11:00 12:25 13:50 15:15 16:50 18:15 19:45 21:10 22:25 11:45 14:35 17:35 20:25
3690 Hillcap Avenue, San Jose

West Wind Capitol 6 Drive-In
20:25

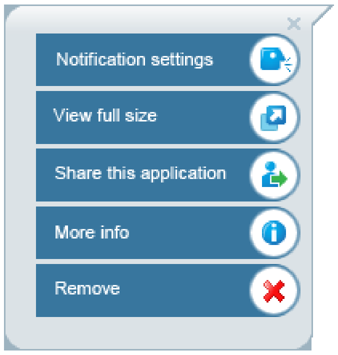
© 2008 Yahoo! Inc. - Terms of Use

Copyright © 2008 Yahoo! Inc. All rights reserved. | Copyright/IP Policy | Terms of Service | Community Guidelines | Build an Application on Yahoo!
NOTICE: We collect personal information on this site. To learn more about how we use your information, see our Privacy Policy.

Chrome

The chrome provides a visual boundary to an Open Application. As shown by the following screenshot, the chrome's menu enables the end user to manage the application.

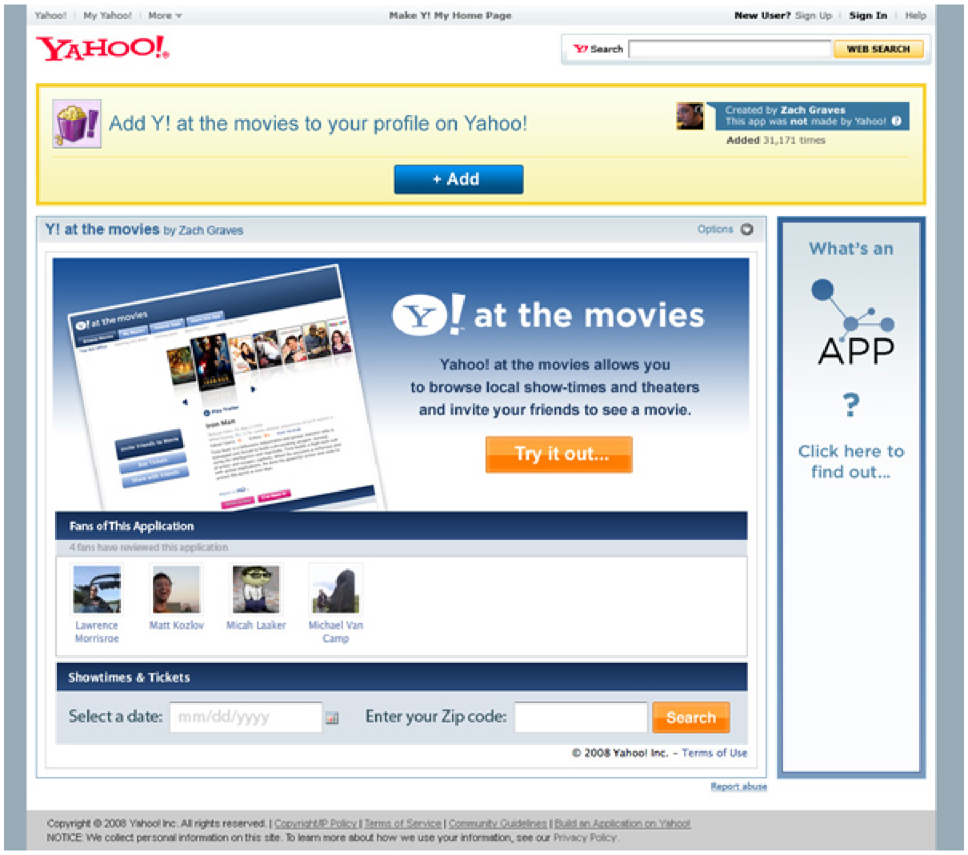
Figure 2.4. Chrome Screenshot



Landing Page

The Landing Page is the default location where an Open Application's Canvas view is presented. End users can access the Landing Page via a static URL, such as http://apps.yahoo.com/-appID. On a Landing Page, the Canvas view is enclosed by Yahoo!'s standard header, footer, and an advertisement unit (160 px wide by 600 px high).

Figure 2.5. Landing Page Screenshot



Add Y! at the movies to your profile on Yahoo!

Created by Zach Graves
This app was not made by Yahoo!
Added 31,171 times

Yahoo! at the movies allows you to browse local show-times and theaters and invite your friends to see a movie.

4 fans have reviewed this application

Select a date: mm/dd/yyyy Enter your Zip code:

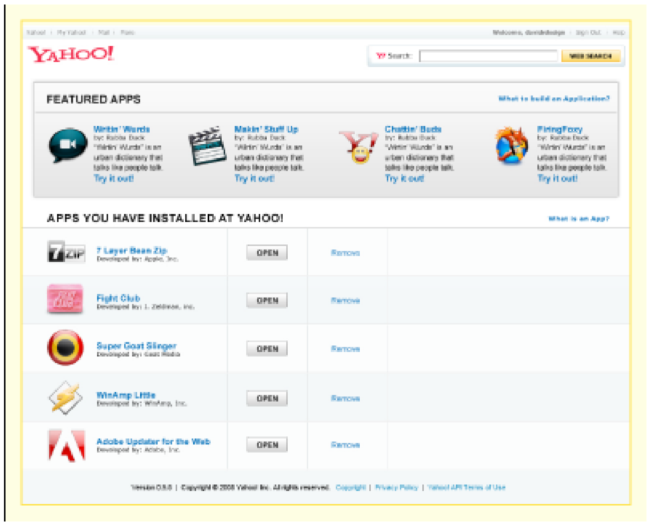
© 2008 Yahoo! Inc. - Terms of Use

Copyright © 2008 Yahoo! Inc. All rights reserved. | Copyright/IP Policy | Terms of Service | Community Guidelines | Build an Application on Yahoo!
NOTICE: We collect personal information on this site. To learn more about how we use your information, see our Privacy Policy.

My Applications

My Applications lists all Open Applications installed by an end user. When an application is installed, it is added to My Applications regardless of where the installation took place. In My Applications, every application is represented by its icon and name. Clicking on an application sends the end user to the application's Landing Page, displaying the Canvas view of the App. My Applications can be accessed at http://apps.yahoo.com/myapps, and will be promoted around the Yahoo! network in future releases.

Figure 2.6. My Applications Screenshot



FEATURED APPS

What is build an Application?

Writin' Wards
by: Rubba Duck
"Writin' Wurds" is an urban dictionary that talks like people talk.
Try it out!

Makin' Stuff Up
by: Rubba Duck
"Writin' Wurds" is an urban dictionary that talks like people talk.
Try it out!

Chattin' Buds
by: Rubba Duck
"Writin' Wurds" is an urban dictionary that talks like people talk.
Try it out!

FiringFoxy
by: Rubba Duck
"Writin' Wurds" is an urban dictionary that talks like people talk.
Try it out!

APPS YOU HAVE INSTALLED AT YAHOO!

What is an App?

7 Layer Bean Zip
Developed by: Apple, Inc.

Fight Club
Developed by: I. Zeldman, Inc.

Super Goat Slinger
Developed by: Goat Media

WinAmp Little
Developed by: WinAmp, Inc.

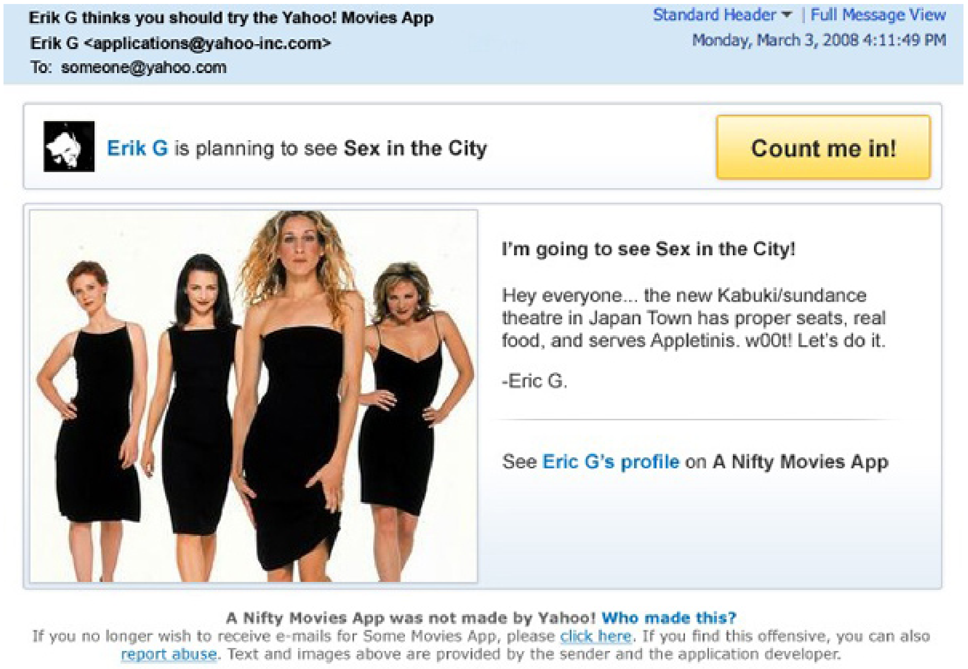
Adobe Updater for the Web
Developed by: Adobe, Inc.

Version 0.5.0 | Copyright © 2008 Yahoo! Inc. All rights reserved. Copyright | Privacy Policy | Yahoo! API Terms of Use

Notifications

Notification is a mechanism enabling end users to send messages to each others through Open Applications. When an end user performs an action that triggers this mechanism, he or she is asked to select the recipients of the action. The message must be approved by the sending user before it can be sent. A person who receives this message may opt out of receiving further Notifications from the Open Application that sent it.

Figure 2.7. Notifications Screenshot


Erik G thinks you should try the Yahoo! Movies App
Erik G <applications@yahoo-inc.com>
To: someone@yahoo.com

Monday, March 3, 2008 4:11:49 PM

Erik G is planning to see Sex in the City

I'm going to see Sex in the City!

Hey everyone... the new Kabuki/sundance theatre in Japan Town has proper seats, real food, and serves Appletinis. w00t! Let's do it.

-Eric G.

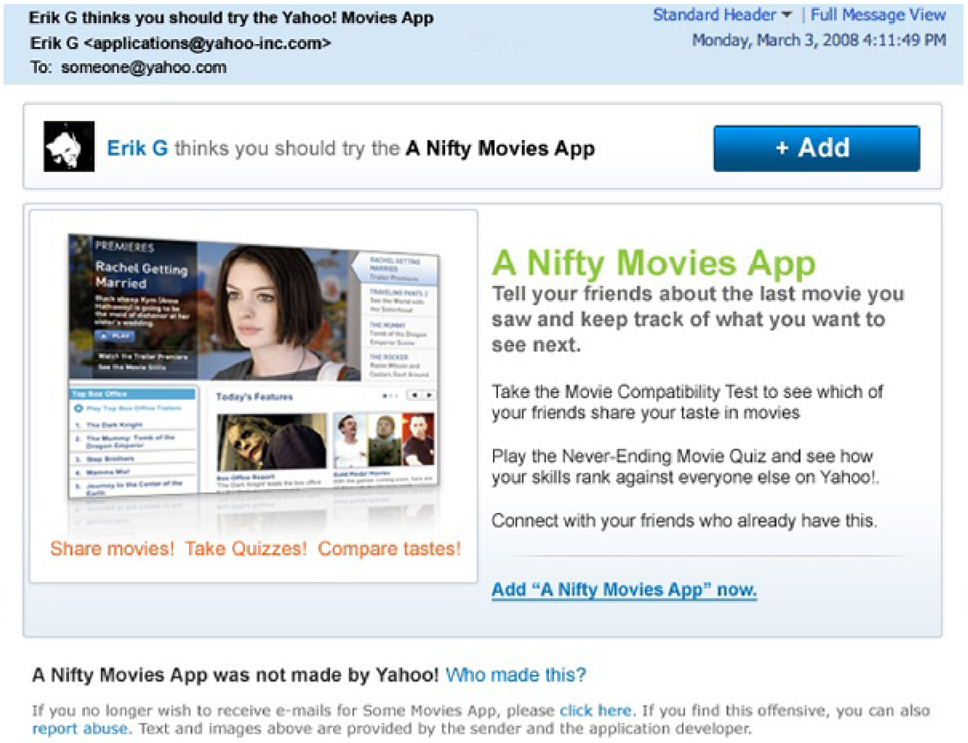
See Eric G's profile on A Nifty Movies App

A Nifty Movies App was not made by Yahoo! Who made this?
If you no longer wish to receive e-mails for Some Movies App, please click here. If you find this offensive, you can also report abuse. Text and images above are provided by the sender and the application developer.

Invitations

An invitation is a Notification that explicitly requests the recipient to install an Open Application. An Invitation must be initiated by an end user who has installed the Open Application, and can only be sent to the sending user's Connections (a mutually-confirmed relationships).

Figure 2.8. Invitations Screenshot



Erik G thinks you should try the Yahoo! Movies App
Erik G <applications@yahoo-inc.com>
To: someone@yahoo.com

Standard Header | Full Message View
Monday, March 3, 2008 4:11:49 PM

Erik G thinks you should try the A Nifty Movies App

+ Add

A Nifty Movies App

Tell your friends about the last movie you saw and keep track of what you want to see next.

Take the Movie Compatibility Test to see which of your friends share your taste in movies

Play the Never-Ending Movie Quiz and see how your skills rank against everyone else on Yahoo!.

Connect with your friends who already have this.

Share movies! Take Quizzes! Compare tastes!

Add "A Nifty Movies App" now.

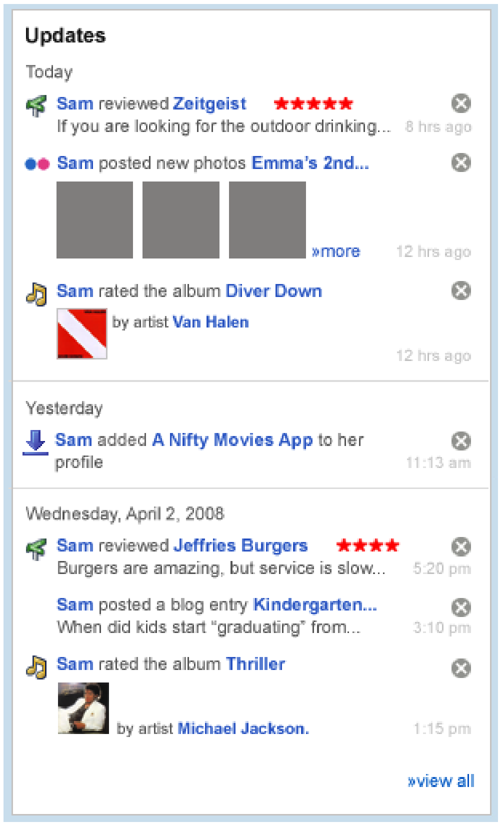
A Nifty Movies App was not made by Yahoo! Who made this?

If you no longer wish to receive e-mails for Some Movies App, please click here. If you find this offensive, you can also report abuse. Text and images above are provided by the sender and the application developer.

Updates

Updates are Yahoo!'s user event stream. When an end user installs an Open Application, the Application may be granted rights to read from and write to the user's Updates.

Figure 2.9. Updates Screenshot



Updates

Today

Sam reviewed Zeitgeist ★★★★★
If you are looking for the outdoor drinking... 8 hrs ago

Sam posted new photos Emma's 2nd...
»more 12 hrs ago

Sam rated the album Diver Down
by artist Van Halen
12 hrs ago

Yesterday

Sam added A Nifty Movies App to her profile 11:13 am

Wednesday, April 2, 2008

Sam reviewed Jeffries Burgers ★★★★
Burgers are amazing, but service is slow... 5:20 pm

Sam posted a blog entry Kindergarten...
When did kids start "graduating" from... 3:10 pm

Sam rated the album Thriller
by artist Michael Jackson. 1:15 pm

»view all

Future Application Elements

This release is primarily intended for application developers. As such, it does not include all the elements needed for full distribution of your application on the Yahoo! Network. In a future release, the following elements will be available:

- Applications Gallery: The place where end users discover your applications by searching for keywords or browsing pre-defined categories.
- Network Integration: End users will be able to install Open Application views on the Yahoo! Network.

Chapter 3. Yahoo! Markup Language

See the Yahoo! Markup Language Reference[1].

[1] ../../yap/yml/

Chapter 4. Caja Support

Introduction

What is Caja?

Caja is a system that transforms ordinary HTML and JavaScript into a restricted form of JavaScript. The transformation is called "cajoling", and the result is "cajoled script". Caja is an Open Source project sponsored by Google and hosted at Google Code[1].

The cajoled script is then run within a security sandbox created in your browser. This provides a way to safely include arbitrary third-party content on any Web page.

In principle, Caja should be transparent. Most JavaScript behaves the same whether it's run directly or cajoled. However, since Caja is currently incomplete and rapidly evolving, there are many noticeable differences.

Caja Status for This Release

Since Caja is used to transform an application's HTML and JavaScript into a restricted form that prevents malicious applications from doing damage, applications cannot contain arbitrary ActiveX objects, use `eval` to get around the ActiveX restriction, or use iframes to get around the `eval` restriction.

Other than that, most JavaScript application elements should work. Our goal is to make Caja as unobtrusive as possible for ordinary applications. However, we're not there yet. Caja still has many rough edges, and you may experience mysterious Caja behavior. This document will describe some of those mysteries in detail.

To get started right away, use Firefox with the Firebug[2] add-on and set `alert` to bring up the Firebug console. Other browsers are not currently supported.

Note the following restrictions that apply to this release:

- Complex libraries such as YUI, jQuery, and Prototype are not yet supported.
- The `document.write` method isn't supported. However, `innerHTML` and many commonly-used DOM interfaces are currently supported.
- Global variables that are not defined with `var` will cause compile-time and run-time errors.
- If something doesn't work, check your Firebug console, even if the Firebug icon doesn't indicate any errors. There are several common runtime errors that don't raise exceptions. Instead, they show up as plain messages in the console like this:

  ```
  Not readable: ([SomeClass]).foo
  ```

  Messages of this type usually means you're trying to do something that isn't yet supported, and you need to find an alternative .

[1] http://code.google.com/p/google-caja/
[2] http://getfirebug.com/

- Caja currently restricts `obj.prototype` and constructors in a way that blocks some common JavaScript idioms, such as monkey-patching.

Why Do We Need Caja?

When a website wants to include arbitrary third-party content, it needs to consider many potential security problems. One of the harder problems is "drive-by downloads": an attacker inserts malicious HTML that tries to install malware when you view the page.

A typical vector is an `<iframe src=...>` tag pointing at the attacker's website. Your browser automatically loads the iframe, which runs a script that figures out what browser and extensions you have, then downloads malware targeted specifically at the vulnerabilities known for your system.

The traditional solution to this problem is to aggressively sanitize third-party content by removing iframes, removing scripts, etc. That works well in many cases, but aggressive sanitization makes it difficult to create interesting applications.

Today, we want to allow anybody to create interesting applications that can appear on our site, but we also want to limits our users' exposure to scripts that install malware.

Sanitizing JavaScript is difficult, and that's what Caja is about.

How Does Caja Work?

Caja has two main parts:

- server-side translator
- client-side runtime support

The Server-Side Translator

The Caja translator rewrites arbitrary HTML and JavaScript into safe HTML and JavaScript, using white list security principles, by

- Removing anything it doesn't understand
- Removing HTML and CSS that isn't on a white list
- Modifying CSS rules, limiting them to a sandbox `<div>`
- Transforming JavaScript into forms known to be safe

The JavaScript transformation is the complicated part. It's basically a form of virtualization:

- Replaces references to global variables with references to a sandbox-specific `IMPORTS___` object
- Rewrites references to `this` to prevent access to the real global scope
- Replaces most JavaScript code with semantically similar code that has runtime checks for security
- Rejects some JavaScript code early, such as `with(obj){...}`.

Here's an example transformation. This JavaScript source code:

```
   function f(a, b) {
      a.j = b.k(G);
   }
```

is cajoled into something like this:

```

   var G = ___.readImport(IMPORTS___, 'G');
   var f = ___.simpleFunc(function (a, b) {
      var x0___;
      var x1___;
      var x2___;
      var x3___;
      x3___ = b,
      x2___ = c,
      x1___ = ___.callPub(x3___, 'k', [ x2___ ], 0);
      x0___ = a,
      ___.setPub(x0___, 'j', x1___, 1);
   }, 'f');
```

Note

The actual Caja transformation is slightly different. This example has been modified a bit to make it easier to see what Caja is doing under the hood.

The main purpose of the transformation is to guarantee that cajoled script can't access arbitrary global variables. Cajoled script can only use objects and functions that are explicitly given to it by the container. Basically, cajoled script conforms to an object-capability security model.

For more details about the JavaScript transformation, see the Caja project page.

(As of 9/2008, the Caja project is in the process of migrating to a new rewriting scheme called "Valija". The description in the Caja paper is accurate for the current Caja generation, but it's going to be wrong when Valija takes over.)

The client-side runtime

Cajoled script can't access any real global objects without help, and that's what the Caja runtime system is for. The runtime system creates a useful sandbox environment by adding objects to an `IMPORTS___` object that's given to the cajoled script.

Some of the imported objects are the real thing. For example, `IMPORTS___.Array` is identical to the browser's `Array`.

Some of the imported objects are proxies. For example, `IMPORTS___.document` is a proxy object that exposes a safe subset of the DOM interface. The proxy function `document.getElementById` will return objects that are also proxies. Basically, you never get direct access to a real DOM object, but that generally doesn't matter, because for most purposes the proxy objects are similar enough to the real thing.

The runtime system also enforces the Caja security model, by checking that objects and functions were properly tagged before they're used. You can see Caja's internal tagging when you examine objects in Firebug. Most objects will have properties that end with triple-underbar, such as `length_canRead___`, `___FROZEN___`, etc.

How Do I Debug My Application?

Your best bet at the moment is to work in Firefox with Firebug. Caja's runtime library will send errors and diagnostics to Firebug's console.

First, if you're getting weird error messages, check if they're explained here.

Most `LINT` and `WARNING` messages are harmless and can be ignored. However, there's one `WARNING` that's important:

```
WARNING: failed to load external url ...
```

The reason that's important: External scripts are not loaded currently, but Caja doesn't flag that as a fatal error. If you see that warning for a `<script>` tag, then your application probably won't work. Fix that problem first. Either inline the external script, or modify your code to eliminate the external dependency.

`alert` works, but the messages are redirected to Firebug's console.

Always check your Firebug console, even if the Firebug icon doesn't show any errors. Some of Caja's runtime errors will print an error message without throwing or re-throwing an exception, so Firebug never sees the error and doesn't update the error count.

Caja's JavaScript transformation makes it difficult to interpret stack traces in Firebug. To compensate, Caja has a debugging mode, which we've enabled automatically. If your application throws an exception, Caja's runtime will try to print a meaningful backtrace in Firebug's console.

YML Alters Caja Line Numbers

The line numbers in Caja's error messages are line numbers that Caja sees, which is not necessarily the same as the line numbers in your source code. In particular, YML transformation happens before Caja, so if you use any YML tags, the line numbers reported by Caja might be offset by an arbitrary amount.

At the moment, there isn't an easy way to determine the original line number.

For compile-time errors, Caja will print the source line text for each error, and hopefully that's enough to identify the actual location.

For run-time errors, you can sometimes figure out the real location by looking at the JavaScript that Caja generates.

What HTML Tags are Blacklisted?

Caja does not allow the following HTML tags:

```


<applet>
<base>
<basefont>
<embed>
<frame>
<frameset>
<iframe>
```

```
<isindex>
<meta>
<noframes>
<noscript>
<object>
<param>
<title>
```

What Works in Caja?

This is a brief overview of what's expected to work in Caja currently.

Browsers Supported	Firefox 3 works well, and Caja debugging is somewhat tailored for Firebug.
	IE 7 works, but it's a litlle flaky. In particular, `<input>` elements have several problems.
	Safari 3 works, but it doesn't get exercised much, so there might be some odd corners.
	Opera 9 probably works, but it doesn't get exercised at all.
HTML and CSS	Caja supports most of the HTML 4.01 and CSS 2.1 specs, as well as some common browser extensions. At the moment Caja tends to adhere more closely to the specs than browsers do, so it will warn or reject some things that are nonstandard but safe and supported by the major browsers.
YML	All YML tags should work with Caja, though some might run into issues with the flaky IE support.
DOM manipulation	Caja provides proxied access to the DOM. If you look under the hood, you can see that you're actually manipulating instances of `TameDocument`, `TameNode`, etc.
	Many of the common DOM operations work, such as `document.getElementById`, `document.createElement`, `node.firstchild`, etc. However, there are still large chunks missing in the DOM implementation.
	Note that `document.write` is deliberately *not* supported, as explained below. Setting and getting `innerHTML` *is* supported.
Events and Timers	Most event handlers work. You can attach handlers with HTML `onevent=` attributes, or with the DOM `addEventListener` method, or by assigning to the `node.onevent` property.
	Note: `node.onevent='...'` will not work. The value must be a function.
	`window.setTimeout` and `window.setInterval` both work.
	Note: `event.fromElement` is IE specific and is not supported by Caja. Instead, use `event.target` or `event.relatedTarget`, depending on the event type.

OpenSocial 0.8	Most of our OpenSocial 0.8 support is explicitly whitelisted for use in Caja. We're not supporting OpenSocial 0.7. If you get runtime "Not readable" errors related to OpenSocial objects, check if you're using an OpenSocial 0.7 interface that disappeared in 0.8.

What are Caja's Limitations?

This describes various things that don't work in Caja, along with some workarounds. Some of these are deliberately not supported, some are accidentally not supported.

This description is specific to our Caja instance. Caja has a flexible configuration, so you might not see exactly the same behavior elsewhere.

Server-Side vs. Client-Side Sanitization

Caja has two similar but distinct HTML sanitization processes.

Server-side sanitization is the full cajoler transformation applied to your application. Server-side sanitization supports complex features like scripts and stylesheets.

Client-side sanitization happens when you set `innerHTML` in your application. Client-side sanitization is more restrictive, and will silently strip out scripts and stylesheets.

HTML Limitations

`<a target=...>`	We allow `target=_blank` and `target=_top`. Other values are deleted with a warning. Omitting `target` is the same as `target=_top`.
`<embed>`	For Flash, use `<yml:swf>` instead. Other embeds are not supported.
`<head>`	All contents of the `head` element are stripped. Use plain HTML starting from inside the body.
`<iframe>`	Not supported yet. Workaround depends on what you're trying to do.
`<link rel=stylesheet>`	External stylesheets are not supported yet. Workaround is to inline the stylesheet.
`<object>`	For Flash, use `<yml:swf>` instead. Other embeds are not supported.
`<script>`	Inline scripts are supported by the server-side cajoler, but they're stripped by the client-side sanitizer. Workaround depends on what you're trying to do.
`<script src=...>`	External scripts are not supported yet. Workaround is to inline the script.
`<style>`	Stylesheets are supported by the server-side cajoler, but they're stripped by the client-side sanitizer. Workaround depends on what you're trying to do.
`javascript:void(0)`	Caja currently rejects any `javascript:` URLs.

If you're trying to use `javascript:void(0)` to make `<a>` buttons, try using an onclick handler instead, something like this:

```
<a href="#" onclick="click(); return
false">click</a>
```

Radio Buttons

Radion buttons do not work in Internet Explorer. There is no work around, but this should be fixed soon.

URL policy

We currently allow `http`, `https`, and `mailto` URLs. Relative URLs are not allowed.

This policy applies to any use of URLs, such as `<a>`, `<img>`, etc.

CSS Limitations

`[]` selectors	Not supported. Workaround is to use the `class` attribute to mark the nodes you want, then select on that class.
`expression()`	Not supported. Workaround depends on what you're trying to do.
`@import`	Not supported yet. Workaround is to inline the stylesheet.

JavaScript limitations

`eval()`

Not supported yet. Workaround depends on what you're trying to do.

`new Function()`

Not supported. This is similar to `eval`. Workaround depends on what you're trying to do.

Assigning strings as event handlers

Code like this is not supported:

```
node.onclick = '...';
```

That's an implicit `eval`, and has the same problem as explicit `eval`.

Instead, put your event-handling code in a function, and assign the function to the event handler:

```
function handle_click() { ... }
node.onclick = handle_click;
```

Names ending with underscore

You can't use names ending with triple-underscore. Those are reserved for Caja's internal bookkeeping.

You can't use names ending with double-underscore. Those are reserved for browser extensions.

You *can* use names ending with single-underscore, but Caja adds some restrictions. Properties ending with single-underscore are class-private, and they can only be accessed via `this`. Basically, `this.x_` is OK, but `foo.x_` is either a compile-time error or a run-time error.

`with (obj) { ... }`

This is not allowed since the dynamic behavior of `with` makes it difficult to analyze its security implications. The workaround is to remove the `with` statement and write `obj.foo` instead of just `foo`.

Assignment to implicit global variables

Caja rejects this at compile-time:

```
<script>
  window = 3;
</script>
```

The error message looks like this:

`FATAL_ERROR: ...: Cannot assign to a free module variable: ... ...`

Less obviously, this is the same compile-time error:

```
<script>
  function a() { foo = 3; }
</script>
```

This is because `foo` isn't declared anywhere.

Caja treats any undeclared variable as a name imported from the container, and it considers those bindings to be read-only.

The fix is to always declare your variables with `var`, in either a local or global scope.

Calling a method as a function

If `obj.foo` is a method that refers to `this`, then peeling it off as a function and calling it directly will cause a runtime error:

```
<script>
  var get = document.getElementById;
  get('x');  // fails
</script>
```

That doesn't really work in raw JavaScript either, but under Caja the error message is much more cryptic. In the Firebug console, you'll see something like this:

```
  Method function (id) {
      id += idSuffix;
      var node =
this.doc___.getElementById(id);
     return tameNode(node, this.editable___);
  } is already attached.
  this: [object Window]
  self: [Fake Document]
  ...
  {}["\nError: " + str] is not a function
```

The reason is, JavaScript doesn't have bound methods, so that code probably doesn't do what you want. The code is saying to call `getElementById` with `this` bound to the global object, which is `window`. So if it succeeded, `getElementById` would get `this=window`, instead of `this=document`.

The usual way of capturing a method as a function is to wrap the method call in a function:

```
  <script>
    var get = function(el) { return
document.getElementById(el); };
    get('x');  // works
  </script>
```

And that should work with or without Caja.

`select.length` — `select.length` is not readable.

DOM Limitations

Many of the holes in the DOM interface are currently due to missing implementation, not due to any particular security concern. In general, if the function or property you're trying to use isn't supported yet, you might be able to do the same thing using interfaces that are already supported.

`document.write()` — Not supported yet. `document.write` can install malware, and it's difficult to make a safe version that duplicates the way it handles partial HTML fragments.

Instead, use `innerHTML`:

```
<div id="x"></div>
<script>
  var x = document.getElementById('x');
  x.innerHTML = 'abc';
</script>
```

You can also build up a DOM tree in pieces using `document.createElement` and so forth.

`window.event` — `window.event` isn't supported yet. This problem shows up if you have event-handling code like this:

```
<a onclick="click()">click</a>
<script>
```

```
  function click() { alert(window.event); }
</script>
```

When you click on the text, you'll see messages like this in the Firebug console:

```
 Dispatch click event thisNode=<a onclick="return
plugin_dispatchEvent___(this, event || window.event,
 0, 'c_1___')">, event=click clientX=22, clientY=93,
 pluginId=0, handler="c_1___"
 Not readable: ([TameWindow]).event
 undefined
```

The workaround is to pass in the event as an argument. Caja always binds `event` within `onevent=` handlers, so you can rewrite the example this way:

```
<a onclick="click(event||window.event)">click</a>
<script>
  function click(event) { alert(event); }
</script>
```

Another workaround is to use `addEventListener` instead, but there's a bug in that. See below.

`node.parentNode()`	`node.parentNode` does not work until the Node has been added to the DOM. The workaround is to add the element to the DOM before trying to access the `parentNode`.
`node.attributes()`	`node.attributes` is not readable. Use `getAttribute/setAttribute` instead.

What Do These Messages Mean?

Compile-Time Errors

Compile-time errors show up as messages at the end of your application preview, and look something like this:

```
  stdin:72: </html>
                   ^^
  LINT: stdin:72+8 - 73+1: Non-space character in page trailer.
```

Most `LINT` and `WARNING` messages are harmless and can be ignored, but pay attention to any "failed to load external url" warnings.

Here are some of the common errors:

`FATAL_ERROR: ...: Cannot assign to a free module variable: ...`	You need to declare your global variables with `var` before you use them. See Assignment to implicit global variables [23].

```
WARNING: ...: failed to
load external url ...
```

External scripts and stylesheets are currently not supported. Note, failure to load a script is not a fatal error, so Caja will keep processing your application, and it will probably fail on some issue that's a side effect of the load failure. One common effect is "Cannot assign to a free module variable". Error messages like that are misleading in this case. The real problem is the load failure.

Inline the external content to get around this.

```
ERROR: ...: css property
 color has bad value:
==>...<==
```

Caja is currently pretty strict about color names in CSS rules. Only the 16 standard color names are recognized. Use hexadecimal colors to get around this.

```
LINT: Stray 'html' start
 tag.
LINT: 'body' start tag
found but the 'body'
element is already open.
LINT: Non-space character
 after body.
LINT: Non-space character
 in page trailer.
```

These are basically harmless irregularities that are caused by Caja's too-strict parser. You might get a lot of these if your HTML uses DOS CR-LF line endings. It's safe to ignore these errors

Runtime errors

Runtime errors usually show up as plain messages in the Firebug console. Sometimes a runtime error will raise an exception that Firebug will catch and report as an error, but these exceptions are often unrelated to the actual error.

Here are some of the common messages:

```
{}["\nError: " + str] is
 not a function
```

This shows up as an error in the Firebug console, but the actual error is something else. This is Caja attempting to throw an uncatchable exception after it has detected a problem. The actual problem is probably a plain message earlier in the console.

```
Not readable:
([Object]).foo
```

This shows up in the Firebug console when a script tries to access a nonexistent property or a nonexistent global variable. It isn't a fatal error. The operation will return `undefined`, and the script will continue, but an unexpected `undefined` can trigger other errors.

```
Not readable:
([Object]).foo
obj is undefined
  function canRead(obj,
name) { return !!obj[name
 + '_canRead___']; }
```

This shows up in the Firebug console when a script tries to access a property of a nonexistent object.

You might get this error if you try to access browser interfaces that aren't supported yet, such as `navigator.appName`.

Chapter 5. OpenSocial Compatibility

Version of OpenSocial Supported by YAP

YAP v1.0 supports the 0.8 version of the OpenSocial JavaScript APIs. However, YAP v1.0 does not support the OpenSocial RESTful API or Gadget XML definitions. (Support for these features will be added in a future release.) In your Yahoo! Open Application, the code for your Canvas view can include calls to the OpenSocial JavaScript APIs. (The Small view does not support JavaScript.) For the Canvas view, YAP filters all JavaScript code with Caja [16].

Yahoo! Inc. is a charter member of the OpenSocial Foundation. We are committed to supporting OpenSocial and are working with the foundation on defining future specifications such as OSML and OpenSocial templates.

This chapter covers OpenSocial support for YAP, but does not discuss OpenSocial in general. For more information on OpenSocial, see the following documentation:

- OpenSocial 0.8 JavaScript API Reference[1]
- OpenSocial 0.8 Gadgets Core API Reference[2]

OpenSocial Features Supported by YAP

Activity

Supported Activity Fields

YAP supports `opensocial.CreateActivityPriority.LOW` for `requestCreateActivity` but does not support `opensocial.CreateActivityPriority.HIGH.` Requests for `HIGH` are treated the same as `LOW.`

Because OpenSocial 0.8 does not specify `ActivityRequestFields,` offsets and limits (using "first" and "max") are not supported.

YAP supports the following `Activity` fields:

- `opensocial.Activity.Field.ID`
- `opensocial.Activity.Field.TITLE`
- `opensocial.Activity.Field.BODY`
- `opensocial.Activity.Field.URL`
- `opensocial.Activity.Field.USER_ID`
- `opensocial.Activity.Field.POSTED_TIME`

[1] http://www.opensocial.org/Technical-Resources/opensocial-spec-v08/opensocial-reference08
[2] http://www.opensocial.org/Technical-Resources/opensocial-spec-v08/gadgets-reference08

Application Data

YAP supports up to 1,024 bytes of data per key and up to 1 MB per application.

Messaging

YAP does not support `requestSendMessage` and `requestShareApp.`

Person and People

Supported Person Fields

The following fields are in every response:

- `opensocial.Person.Field.ID`
- `opensocial.Person.Field.NAME`
- `opensocial.Person.Field.THUMBNAIL_URL`

If the information for the following fields is in the Yahoo! Social Directory, then they are in the response:

- `opensocial.Name.UNSTRUCTURED`
- `opensocial.Person.Field.PROFILE_URL`
- `opensocial.Person.Field.ADDRESSES`
- `opensocial.Address.UNSTRUCTURED_ADDRESS`
- `opensocial.Person.Field.AGE`
- `opensocial.Person.Field.GENDER`
- `opensocial.Person.Field.TIME_ZONE` (This field is not available when fetching friends.)

Supported PeopleRequest Fields

YAP supports only the following `PeopleRequest` fields:

- `opensocial.DataRequest.PeopleRequestFields.FIRST`
- `opensocial.DataRequest.PeopleRequestFields.MAX`

YAP does not support other `PeopleRequest` fields, including `FILTER`, `PROFILE_DETAILS`, and `SORT_ORDER.` YAP behaves as if these fields are set to `ALL.`

Supported DataRequest Fields

YAP supports only the `ESCAPE_TYPE` `DataRequest` field. All requests behave as if `ESCAPE_TYPE` is set to `NONE.`

Using Environment.supportsField

The `Environment.supportsField` method returns true if the specified field is supported by the OpenSocial container. The method does not check to see if the field value exists for a given user.

A user might not provide information such as age, gender, and time zone. If this information is not provided, it is not returned in the response.

The `supportsField` method is useful for checking if the container stores data for a certain field, for example, location. If the container does not support location, then the application must store the location data. If the container does support location, but location is undefined, then the application can prompt the user to update location in the container's preferences.

Permissions

If your Yahoo! Open Application makes OpenSocial calls, on the Permissions tab of the Application Editor, specify Read (or Read/Write) for the following data:

- Yahoo! Profiles
- Yahoo! Updates

Gadget Core APIs

YAP supports the Gadget Core APIs, except for `Prefs`, `Views`, and the `Feature-Specific` API.

Differences Between the 0.7 and 0.8 JavaScript APIs

This section describes some of the important differences between versions 0.7 and 0.8 of the OpenSocial JavaScript APIs. For more information, see the OpenSocial Release Notes for v0.8[3].

Using opensocial.IdSpec

When making some data requests, such as `Activity`, `People`, and `newFetchPersonAppDataRequest`, you must use an `IdSpec` object instead of specifying `opensocial.IdSpec.PersonId` by itself.

Change the following 0.7 code:

```
var req = opensocial.newDataRequest();
req.add(req.newFetchPeopleRequest(opensocial.IdSpec.PersonId.VIEWER),
'viewer');
```

To the following 0.8 code:

```
var idSpec = opensocial.newIdSpec();
```

[3] http://www.opensocial.org/Technical-Resources/opensocial-release-notes

```
 idSpec.setField(opensocial.IdSpec.Field.USER_ID,
opensocial.IdSpec.PersonId.VIEWER);
var req = opensocial.newDataRequest();
req.add(req.newFetchPeopleRequest(idSpec), 'viewer');
```

When NOT to Use opensocial.IdSpec

The `newFetchPersonRequest` method requires a string ID as the first argument, not an `opensocial.IdSpec` object. The `newUpdatePersonAppDataRequest` and `newRemovePersonAppDataRequest` methods have the same requirement. Specifying `opensocial.IdSpec` as the first argument results in an error. Therefore, code such as the following is correct and does not need to be changed for 0.8:

```

var req = opensocial.newDataRequest();
req.add(req.newFetchPersonRequest(opensocial.IdSpec.PersonId.VIEWER),
'viewer');
```

Retrieving Friends With NETWORK_DISTANCE

Because `DataRequest.Group` has been removed from 0.8, `VIEWER_FRIENDS` and `OWNER_FRIENDS` are no longer available. Therefore, in 0.8 you must use `IdSpec.Field.NETWORK_DISTANCE` instead of `DataRequest.Group.`

Change the following 0.7 code:

```

var req = opensocial.newDataRequest();
req.add(req.newFetchPeopleRequest(opensocial.DataRequest.Group.VIEWER_FRIENDS),
 'viewerFriends');
```

To the following 0.8 code:

```

var idSpec = opensocial.newIdSpec();
idSpec.setField(opensocial.IdSpec.Field.NETWORK_DISTANCE, 1); // Greater
 than 1 is not supported.
idSpec.setField(opensocial.IdSpec.Field.USER_ID,
opensocial.IdSpec.PersonId.VIEWER);
// Specifying a GROUP_ID is unnecessary and is not supported. Our
container assumes you always want friends.
 var req = opensocial.newDataRequest();
 req.add(req.newFetchPeopleRequest(idSpec), 'viewerFriends');
```

Activity Response Format

In 0.7, `Activity` request responses are wrapped in an object with the `activities` property set to the actual response. However, 0.8 does not have this wrapper and the response is returned directly.

Change the following 0.7 code:

```
var activities = response.get('myActivityKey').getData()['activities'];
```

To the following 0.8 code:

```
var activities = response.get('myActivityKey').getData();
```

OpenSocial Code Samples

The following code samples demonstrate how you might make OpenSocial calls in the Canvas view of a Yahoo! Open Application.

Activities Demo

```
<xi:include></xi:include>
```

The JavaScript code for the previous listing is also available on a separate page: Activities Demo[4].

Gifts Demo

```
<xi:include></xi:include>
```

The JavaScript code for the previous listing is also available on a separate page: Gifts Demo[5].

[4] examples/activities_demo.txt
[5] examples/gifts.txt

Appendix A. Parameters Passed to an Open Application

The following tables describe the parameters passed by the YAP engine to an Open Application at runtime.

Table A.1. Parameters Passed to an Open Application

Information	Source	Format	Variable	Description
User Information				
Viewer Information	Cookie in the session	POST	yap_viewer_guid	GUID of the person logged into Yahoo! in the browser
Owner Information	Owner of the profile	POST	yap_owner_guid	GUID of the person whose identity information is in the profile
Page Language Requested	Publisher	HEADER	Accept-Language	Language priority list as defined in RFC 2616 section 14.4, with quality values assigned so as to maintain the priority order of the list provided by the publisher.
Page Timezone	Publisher	POST	yap_tz	Timezone Identifier, e.g., "Asia/Seoul".
Page Jurisdiction	Publisher	POST	yap_jurisdiction	ISO 3166[1] country code identifying the country whose rules should be used for content moderation.
Session Information				
Viewer's Access Token	Registration and YAP	POST	yap_viewer_access_token	OAuth 1.1 Access Token
Viewer's Access Token Secret	Registration and YAP	POST	yap_viewer_access_token_secret	OAuth 1.1 Token Secret
AppID	YAP	POST	yap_appid	Application ID
Dropzone	Publisher	POST	yap_dropzone_id	
Hosting Page URL	Publisher	POST	yap_page_url	The URL where the application is hosted.
YAP Authentication				
Consumer Key	YAP Engine	POST	yap_consumer_key	OAuth 1.0 Consumer Key
Request Time	YAP Engine	POST	yap_time	Timestamp of the request as per RFC 3339[2]
Request Signature Method	YAP Engine	POST	oauth_signature_method	HMAC_SHA1 signature method
Request Signature	YAP Engine	POST	oauth_signature	OAuth 1.0 Signature

[1] http://en.wikipedia.org/wiki/ISO_3166
[2] http://www.apps.ietf.org/rfc/rfc3339.html

Browser Information				
User Agent	Browser	HEADER	HTTP_USER_AGENT	
Accept	Browser	HEADER	HTTP_ACCEPT	
Accept Encoding	Browser	HEADER	HTTP_ACCEPT_ENCODING	
Viewer Information				
View	Publisher	POST	yap_view	The request is targeted to either the Small View or Canvas View.

Appendix B. YAP Developer Web Services

General Information

This document lists the REST web services for development on the Yahoo! Application Platform (YAP). The next two sections contain information that is common to all YAP developer web services.

Authorization

The calls to the web services must use OAuth 1.1, with 2-legged authorization that matches the consumer key. Only HMAC-SHA1 signatures are supported.

Response Codes

Response Code	Response Body Contents	Description
200	No content	OK
400, 401	No content	OAuth failure
403	No content	OAuth signature mismatch
5xx	Unstructured body	Internal failure

setSmallView

Description

Replaces the statically rendered small view for a single user.

URI

The `{guid}` string is the Global User ID (GUID) of the Yahoo! user whose small view is set.

```
http://appstore.apps.yahooapis.com/v1/cache/view/small/{guid}
```

Methods

- POST
- PUT

Request Body

YML or HTML for the small view contents. The content-type is ignored, but content encoding is honored.