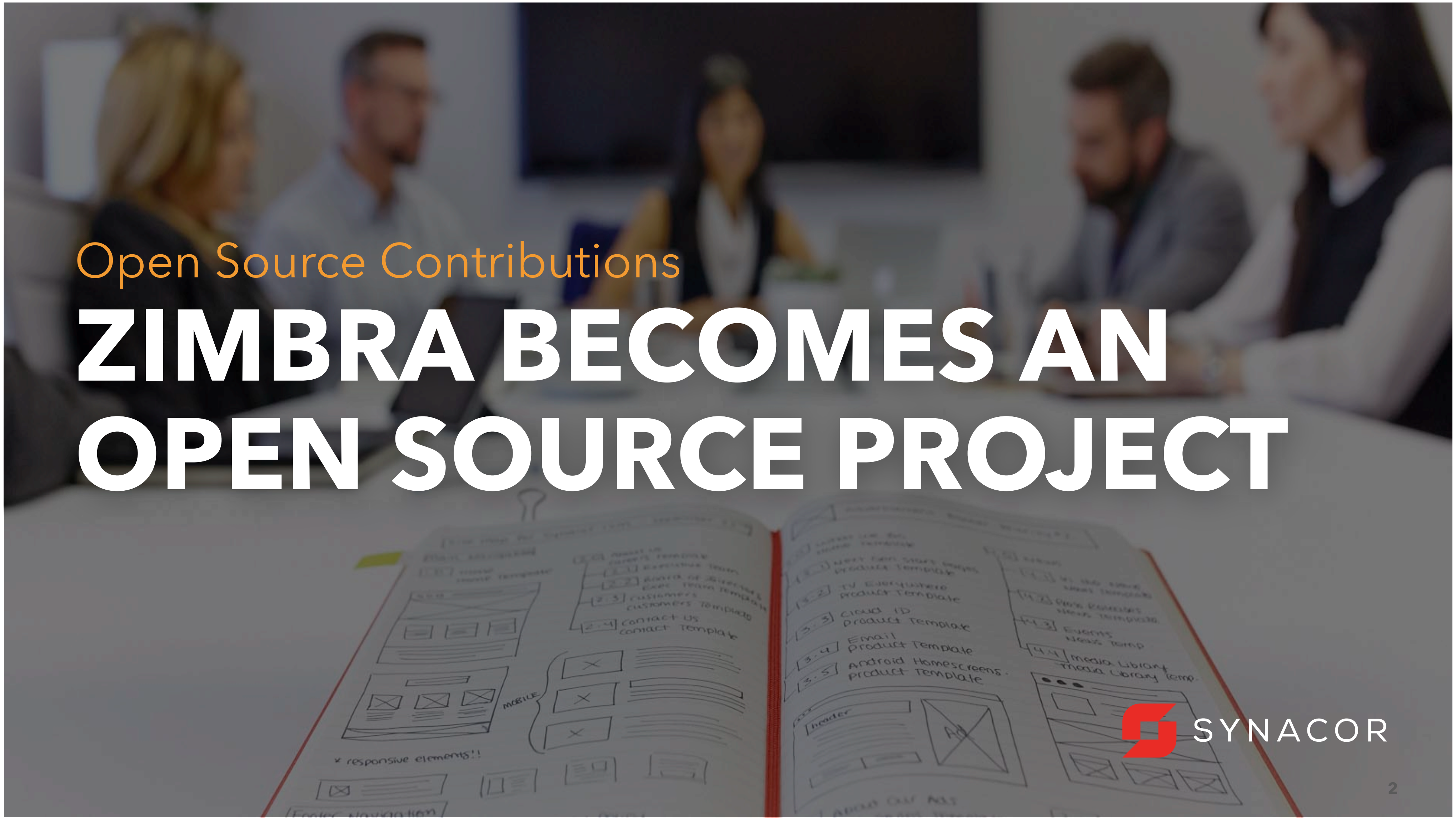
Synacor Presents

Zimbra APxJ Partner Summit 2017

ANANTARA SIAM BANGKOK HOTEL
BANGKOK, THAILAND.
AUGUST 3 & 4 2017

Zimbra: From available source to Open Source Project

OPEN SOURCE CONTRIBUTIONS



Open Source Contributions

ZIMBRA BECOMES AN OPEN SOURCE PROJECT

SYNACOR

ZIMBRA SOURCE CODE AVAILABLE

- Zimbra has been open source since its inception

But …

- Source code was difficult to get access to
  - Product was difficult to build even if you got access to it
- Contribution Agreement was difficult
- Contributions were often ignored in Bugzilla
  - No internal processes for handling contributions

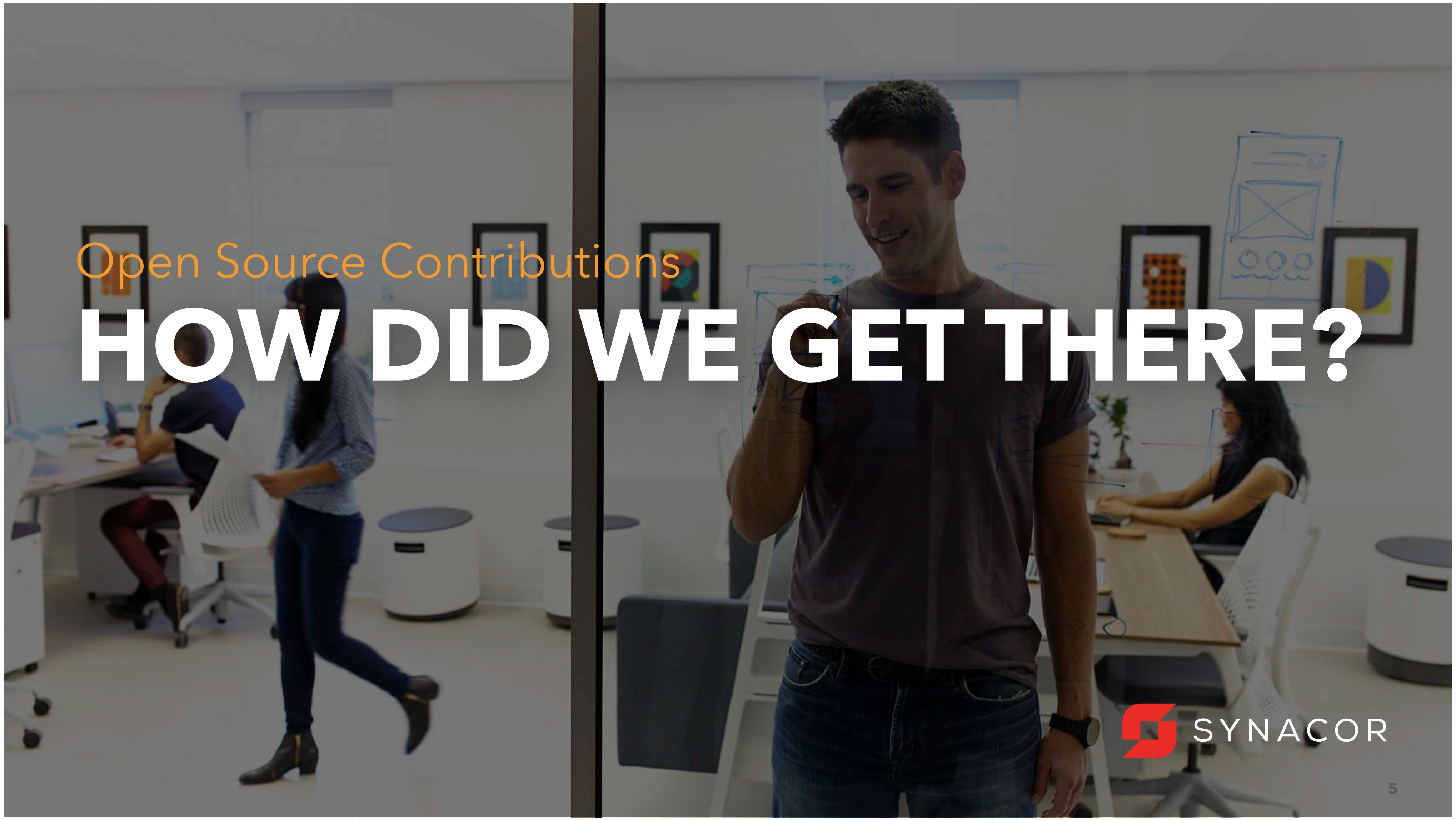
- Zimbra might have been ***open***, but development was ***closed***

All of this together created a large barrier for community members

PROJECT TRANSFORMATION

- Zimbra has transformed
  - Code is easily available and is easy to build
  - Barriers to contributions are gone
  - Contributions
    - Easy to see what is out-standing
    - Built in code review (with automation)
      - Clear interaction with development team
      - Clear understanding of when contributions are included
    - Can be seem by everyone
  - Internal processes in place to handle contributions

Zimbra is now an Open Source Project



Open Source Contributions

HOW DID WE GET THERE?

SYNACOR

MOVING FROM PERFORCE...

- For a long time, Zimbra utilized Perforce internally for version control
    - A copy of Perforce data was publicly exposed
    - Later moved to being mirrored via Git protocol
        - Very often this crashed and was not accessible or just did not work
    - Community members did not use same source code to build FOSS version
        - This caused many, many problems

Perfoce's GitFusion product was used to expose a view of the source code via Git

… TO GITHUB

- Zimbra code base is now hosted on GitHub (utilizing Git for Source Control)
    - GitHub is one of the largest Open Source repository hosting sites
    - Expected by most Open Source developers
    - Code is made highly available.
    - Contributions tied to developers community profile (contributions benefit the contributor)
    - Synacor Engineering team uses the same repositories for all development

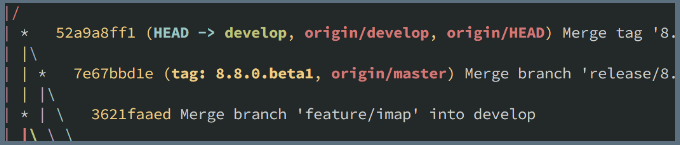
Zimbra began the process of moving to GitHub after v8.7.0 (June 2016)

Version 8.7.1 was the last version created out of Perforce

CODE BRANCHES

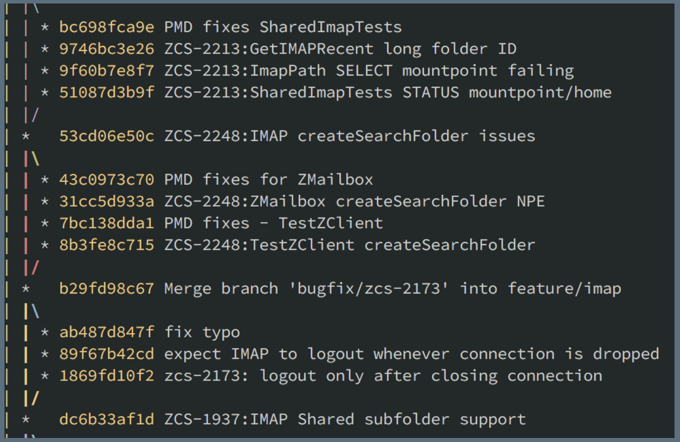
- Each repository has several branches associated with it
    - “master” branch represents the most recent released code
    - “develop” branch is code staged for next release
    - “feature” and “bug” branches represent on-going development
- Easy to track what is in a release, what is about to be released and what is in development
- After integration, easy to see “change sets”

master and development



```
|/
| *   52a9a8ff1 (HEAD -> develop, origin/develop, origin/HEAD) Merge tag '8.
| |\
| | *   7e67bbd1e (tag: 8.8.0.beta1, origin/master) Merge branch 'release/8.
| | |\
| * | \   3621faaed Merge branch 'feature/imap' into develop
| |\ \ \
```

Integrated feature and bug branches



```
| |\
| | * bc698fca9e PMD fixes SharedImapTests
| | * 9746bc3e26 ZCS-2213:GetIMAPRecent long folder ID
| | * 9f60b7e8f7 ZCS-2213:ImapPath SELECT mountpoint failing
| | * 51087d3b9f ZCS-2213:SharedImapTests STATUS mountpoint/home
| |/
| *   53cd06e50c ZCS-2248:IMAP createSearchFolder issues
| |\
| | * 43c0973c70 PMD fixes for ZMailbox
| | * 31cc5d933a ZCS-2248:ZMailbox createSearchFolder NPE
| | * 7bc138dda1 PMD fixes - TestZClient
| | * 8b3fe8c715 ZCS-2248:TestZClient createSearchFolder
| |/
| *   b29fd98c67 Merge branch 'bugfix/zcs-2173' into feature/imap
| |\
| | * ab487d847f fix typo
| | * 89f67b42cd expect IMAP to logout whenever connection is dropped
| | * 1869fd10f2 zcs-2173: logout only after closing connection
| |/
| *   dc6b33af1d ZCS-1937:IMAP Shared subfolder support
| |\
```

REFACTORING MONOLITHIC CODEBASE

- Zimbra has a very large code base
- Perforce repository exposed through Git was approx. 17GB
- Code was monolithic and made up of many inter-related packages
    - Hard to navigate and comprehend

Building Zimbra required downloading ~17GB of source code and binaries

REFACTORING MONOLITHIC CODEBASE - CONTINUED

- With the move to GitHub, this single, monolithic code base was been broken up into components
- Goal is to have each repository represent a single package/library
- Each repository has a unique purpose
- Contributions to one repository does not mean entire product changes
    - Future benefit for upgrades

Building based on new repositories only requires ~1GB

CHANGES IN PROCESS

- Engineering team moved from a single, large team executing against (multi) year-long targets to smaller Agile-based (SCRUM) teams delivering on a predictable cycle
- Moving to Git meant the team was no longer tied to Perforce's archaic process models
    - The team has implemented a clear branching model which allows them to control *when* something is added to the code base
    - No more half-committed features/fixes
- Contributions are packaged as independent "change sets" and can be merged when appropriate
- Internal development and Community contributions follow the same review and merge process
- Isolated changes can be independently tested/verified
    - Mandated by process to occur before inclusion

NEW RELEASE CYCLES

- Zimbra was previously released after *long* development cycles
- Zimbra is now released from Engineering **every 2 weeks**
  - Zimbra Product team still responsible for product versions and when official builds are produced

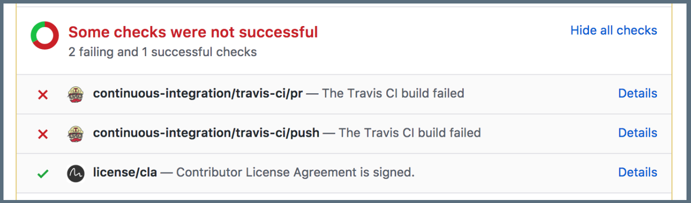
Shorter release cycle means:

community contributions are incorporated rapidly

new features/bug fixes make it to the community faster

AUTOMATION

- Automation has been introduced at almost every level of development
- Pull-requests (both internal and community contributions) are statically analyzed and built with Continuous Integration tools which test the code
    - Results of checks are added as comments in the code review
- Contribution Agreement acceptance is integrated into pull-requests. “One-click”, online acceptance as part of pull-request.
- Has lead to obvious quality improvements in code



Some checks were not successful
2 failing and 1 successful checks
Hide all checks

× **continuous-integration/travis-ci/pr** — The Travis CI build failed Details
× **continuous-integration/travis-ci/push** — The Travis CI build failed Details
✓ **license/cla** — Contributor License Agreement is signed. Details

Automated checks as part of code review

Open Source Contributions

HOW DOES IT WORK?

SYNACOR

CONTRIBUTIONS AND COMMUNITY PULL REQUESTS

- Contributions can be easily made by submitting a pull-request on GitHub
- GitHub's code review tools are utilized to review the contribution (both automated and manually)
- The code is merged into a "bug" or "feature" branch and then merged into "develop" for inclusion in a release
- The next release, after inclusion in "develop", will include the contribution

Low-friction contributions and a well-defined process mean quicker inclusion of contributions

WHERE DO I GO?

Zimbra on GitHub: https://github.com/Zimbra

To build Zimbra Open Source: https://github.com/Zimbra/zm-build

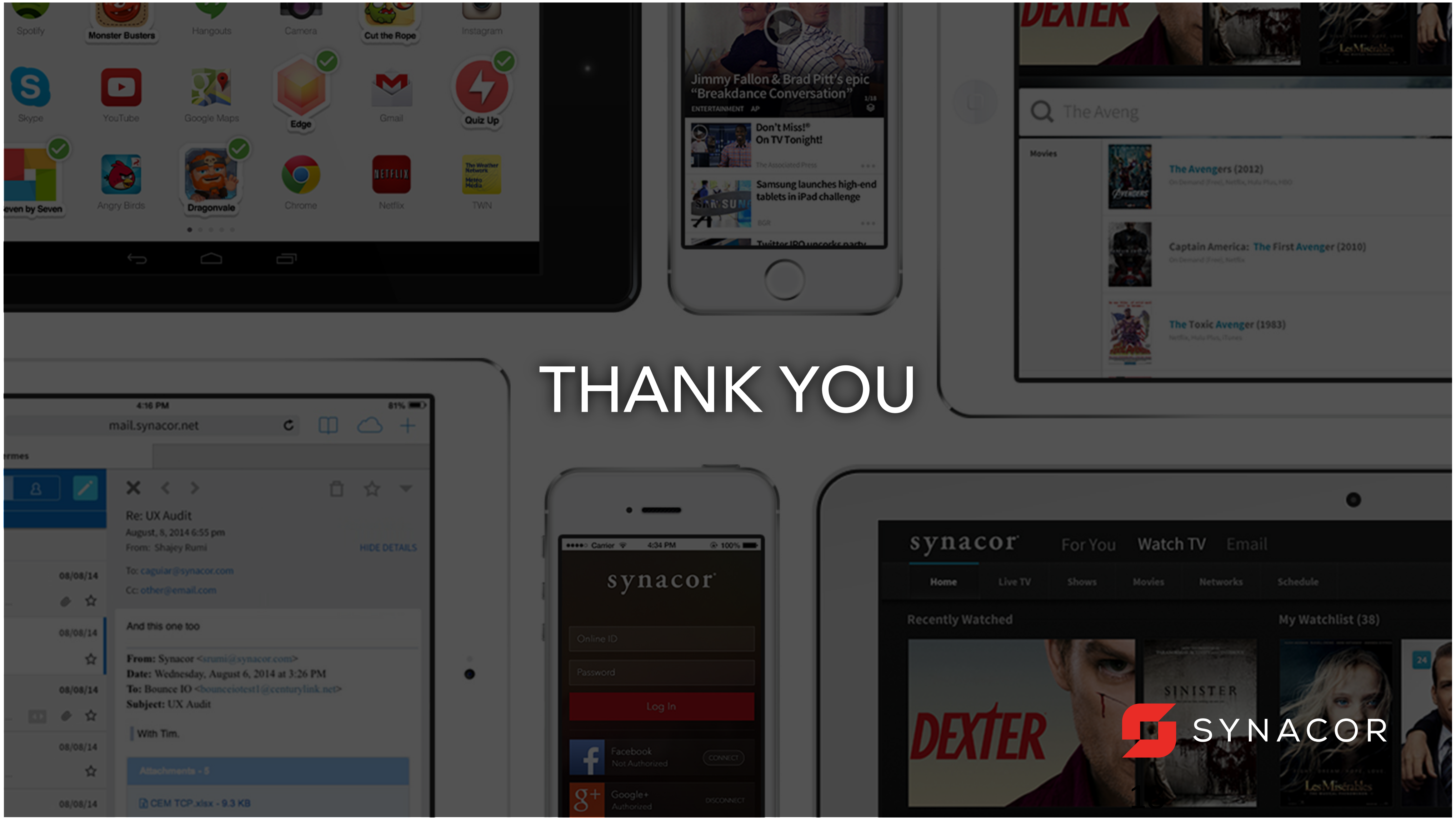
Static Analysis: https://www.codacy.com/app/Synacor/REPOSITORY/dashboard
Where REPOSITORY is the GitHub repository name
Ex: https://www.codacy.com/app/Synacor/zm-mailbox/dashboard

Zimbra Documentation is Open Source too! [CC BY-SA 4.0]
Installation Guides: https://github.com/Zimbra/installguides
https://zimbra.github.io/installguides/ (for online reading)
Administrators Guide: https://github.com/Zimbra/adminguide
https://zimbra.github.io/adminguide/ (for online reading)

Open Source Contributions

QUESTIONS?

SYNACOR



THANK YOU

SYNACOR

Synacor Presents

Zimbra APxJ Partner Summit 2017

ANANTARA SIAM BANGKOK HOTEL
BANGKOK, THAILAND.
AUGUST 3 & 4 2017

- Tarang Khandelwal

EXTENSIBILITY IN ZIMBRA

Agenda

- What is Extensibility?
- Extending Zimbra with Zimlets & Extensions.
- What is a Zimlet?
- What all can a Zimlet do?
- Benefits
- Examples
- Limitations

Why Software products needs to be extensible?

A truly Extensible Software allows 3rd Party/External developers to extend and enhance it further even after they have been compiled and released.

SYNACOR

Add/modify/remove features with ease

FLEXIBILITY & EXTENSIBILITY

SYNACOR

Zimlets, Admin & Server Extensions.

EXTENDING ZIMBRA

SYNACOR

WHAT IS A ZIMLET?

SYNACOR

WHAT CAN A ZIMLET DO?

SYNACOR

THINGS THAT CAN BE DONE USING ZIMLETS

- Create new Tab/Application.
- Hide/Disable unwanted UI elements.
- Add buttons/menu to toolbar.
- Add items to context menus.
- Pattern matching in email content while sending.
- Define & use custom widgets.
- Drag & Drop items into zimlet panel.
- Integrating with 3rd party APIs.
- Displaying content into widgets from outside APIs.

EXAMPLES: ADD ON FEATURES AS ZIMLETS/EXTENSIONS

- Integrate with other products by fetching data through API into Widgets.
- Right click a flight number to see the status of the flight arrival time.
- Right click on a name, address, or phone number to update your address book.
- Right click on a date to schedule a meeting.
- Mouse-over a date or time, and see what's in your calendar.
- Mouse-over a physical address, and see a map or even driving directions and estimated arrival time.
- Mouse-over a customer email address or case tracking number, and see its status.
- Mouse-over an purchase order, see its status, approve/reject and so on.

BENEFITS OF ZIMLETS

SYNACOR

FEATURES & BENEFITS OF ZIMLETS

- Allows extending the existing platform/product with new features.
- Offers flexibility to customize the platform as per need.
- Easy integration with external products.
- Allows offering features to selected users/groups.
- Easily pluggable to existing ZCS installations.
- Zimlets have full access to Core Zimbra JS.
- Easy to Use API hooks.
- Zimlets can be made mandatory (feels like core feature to end user).
- Zimlets can easily be upgraded.
- Zimlets have very small learning curve.

KINDS OF API’S

1. **Zimlet Framework APIs**: APIs for creating Zimlet.
    - XML APIs: Help describe Zimlet & basic functionalities.
    - JavaScript APIs: Helps implement advanced functionalities.

2. **Zimbra SOAP/JSON/REST APIs**: APIs to make calls to Zimbra Server & extend Zimbra.

3. **Zimbra Widget APIs**: JavaScript API to create widgets like buttons, menus, dialog boxes etc.

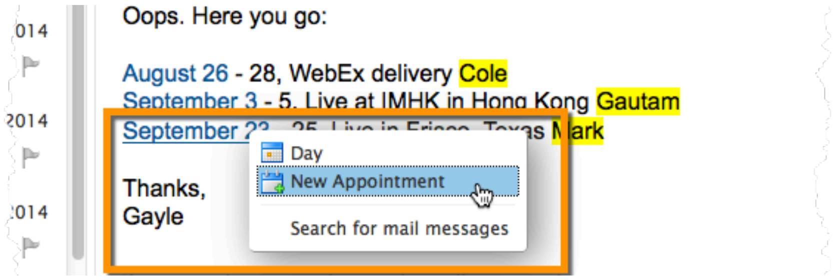
Pluggable features in Zimbra Web Client

ZIMBRA DEFAULT FEATURES AS ZIMLETS

SYNACOR

ZIMBRA DEFAULT FEATURES AS ZIMLETS

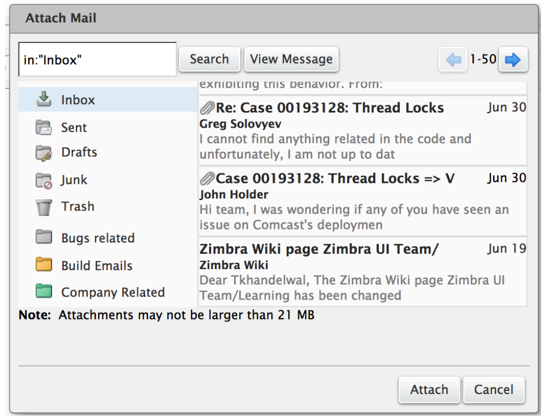
- Undo Send, cancel sending of message.
- Attach Mail/Contacts while composing email.
- Archive Mail feature.
- Schedule & join WebEx meetings.
- Search terms highlighter in emails.
- Highlighting dates & previous appointments.
- Creating appointments directly from dates highlighted in emails.
- Preview associated contact details for an email address.



Oops. Here you go:

August 26 - 28, WebEx delivery Cole
September 3 - 5, Live at IMHK in Hong Kong Gautam
September 22 - 25, Live in Frisco, Texas Mark

Thanks,
Gayle



Attach Mail

in:"Inbox" Search View Message 1-50

Inbox
Sent
Drafts
Junk
Trash
Bugs related
Build Emails
Company Related

exhibiting this behavior. From:

Re: Case 00193128: Thread Locks Jun 30
Greg Solovyev
I cannot find anything related in the code and unfortunately, I am not up to dat

Case 00193128: Thread Locks => V Jun 30
John Holder
Hi team, I was wondering if any of you have seen an issue on Comcast's deploymen

Zimbra Wiki page Zimbra UI Team/ Jun 19
Zimbra Wiki
Dear Tkhandelwal, The Zimbra Wiki page Zimbra UI Team/Learning has been changed

Note: Attachments may not be larger than 21 MB

Attach Cancel

ZIMLETS DEMO

SYNACOR

Which way to implement a feature?

CORE VS ZIMLETS

SYNACOR

LIMITATIONS WITH ZIMLETS

- Not always upgrade safe.
- Load Timing for a zimlet can limit customizations at times.
- Executes after default packages are loaded.
- Cannot affect the default loading view. (Possibly implement at Skin level.)
- Certain features can turn out to be difficult implementing the zimlet way.

ZIMLET 101

Getting started with Zimlet development

SYNACOR

Agenda

- Kinds of Zimlets
- Zimlet Structure
- Writing a Zimlet
- Global App Context
- Zimlet User Properties
- Zimlet API & Hooks
- Dialogs
- Managing Zimlets

KINDS OF ZIMLETS

Service Zimlets

Zimlets that connects to some external service and brings/posts information to ZCS

e.g: WebEx, Flight Tracker, Weather Info etc

Extension Zimlets

Zimlets that extends Zimbra’s core features to provide some extra functionality. These don’t deal with external service.

e.g.: Email Attacher, Sticky Notes, Undo Send, Archive Mails etc

Content or Inline Zimlets

Zimlets that scans the message’s or Contact’s content(body) for some words and converts them to links. These links further provides additional functionalities & events.

e.g: Bugzilla, Email, URL links, Highlight Phone No., Date zimlet etc

TYPICAL ZIMLET STRUCTURE

- Zimlet description xml file(e.g. com_zimbra_test.xml)

  Zimlet information, things like: Zimlet name, Zimlet version, user properties
- A JavaScript file (e.g. com_zimbra_test.js)

  This is the main file that does all the zimlet hard work.
- CSS file(e.g. com_zimbra_test.css)

  Style-sheet that you might use to style a Zimlet dialog/form fields or anything.
- config_template.xml file

  Contains configuration information like "allowedomains " property & global configurations

* **Optional**: icon/images, *.jsp file, *.template file, *.properties i18N file

ZIMLET CONFIGURATION FILE

This file should be hardcoded as config_template.xml

```
<zimletConfig name=”com_zimbra_example” version=”1.0”>
       <global>
          <property name=”serverUrl”>example.com</property>
          <property name="allowedDomains">*.google.com,*.example.com</property>
      </global>
</zimletConfig>
```

Properties can be accessed as:

- Within Zimlet definition file: ${config.global.serverUrl}
- Within JavaScript handler file: this.getConfig(“serverUrl”)

What it takes to create one..

WRITING A ZIMLET

SYNACOR

ZIMLET XML VS JS API

Zimlet XML/JS API	• XML API: Use it to descript the Zimlet. • JS API: Use it to implement both basic & advanced functionalities

MINIMAL STEPS TO GET A ZIMLET READY FOR USE

1. Create Zimlet XML configuration file with minimal configuration.
2. Create JavaScript Handler file.
3. Use Zimlet hooks/Api’s and define functions to customize/enhance the behavior in JS handler.
4. Add any other optional resources like stylesheets, images, I18n files etc
5. Create zip package for the zimlet.
6. Deploy using CLI or Zimbra Admin Console.
7. Access the feature using Zimbra Web Client.

Zimbra Global Object

APPCTXT

SYNACOR

APPLICATION CONTEXT

appCtxt (ZmAppCtxt.js) is a global object that provides access to various applications, dialog boxes and also centrally stores information about current application and about current user information, preferences etc.

When an application is loaded and active:

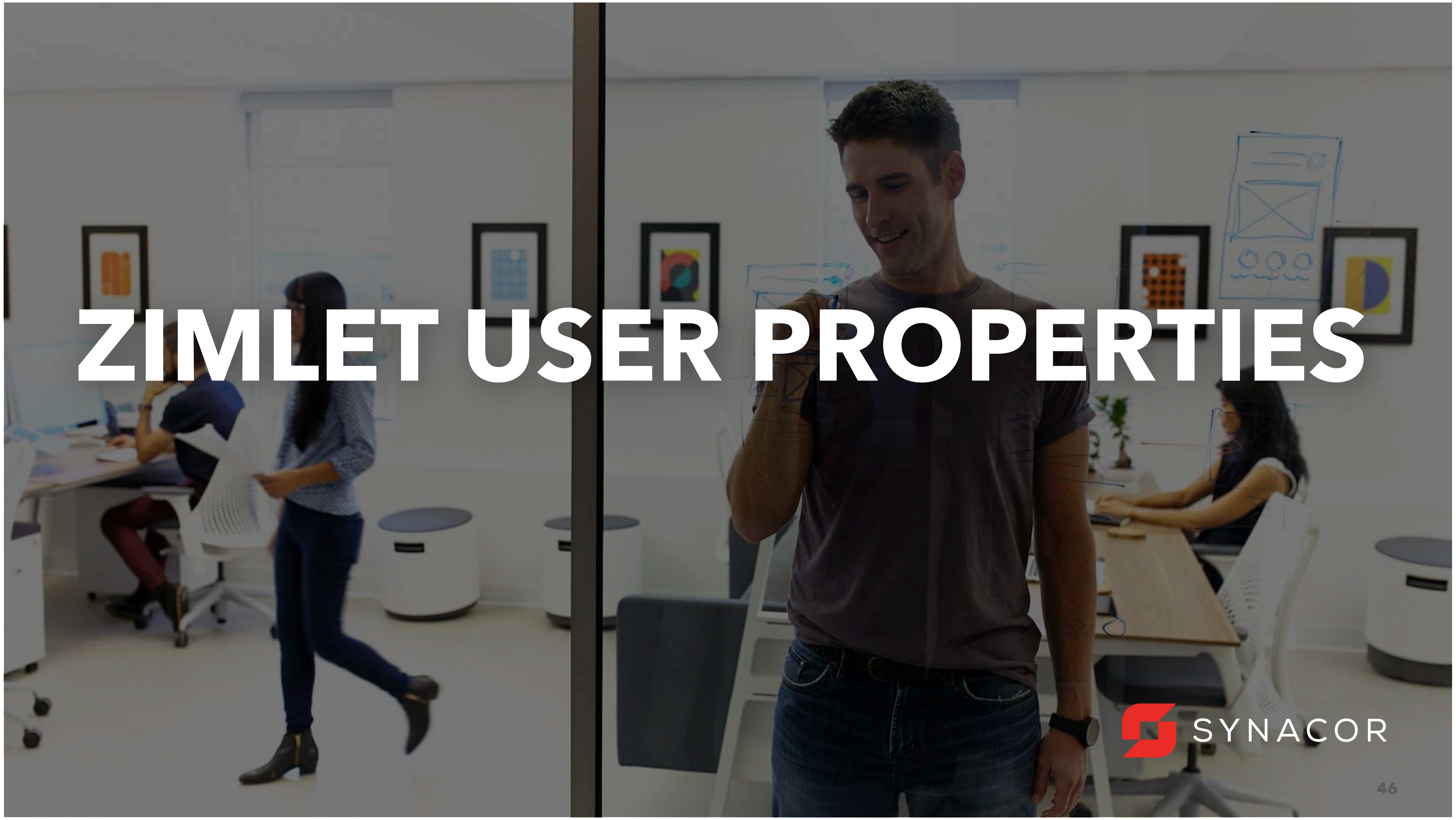
appCtxt.getCurrentApp() – returns current application object
appCtxt.getCurrentAppName() –returns current app's name
appCtxt.getCurrentController() –returns current app's controller

Several Zimbra's widgets also register themselves to appCtxt

appCtxt.getNewFolderDialog().popup() - shows Folder dialog widget
appCtxt.getNewCalendarDialog().popup() – shows New Calendar dialog widget

Access Account information:

appCtxt.getActiveAccount() – returns active account
appCtxt.getActiveAccount().settings –returns all preferences/settings



ZIMLET USER PROPERTIES

SYNACOR

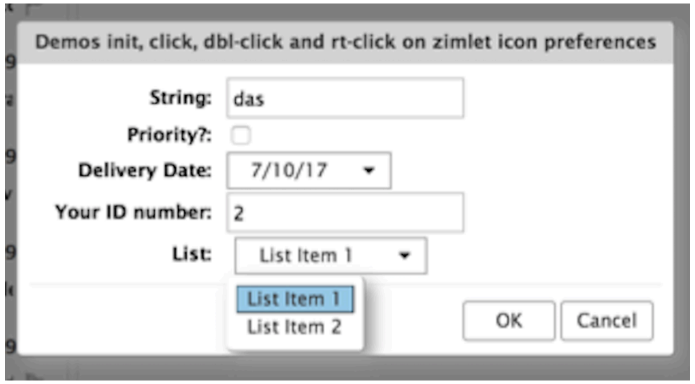
USER PROPERTIES

User properties enable zimlets to store user state/properties with the zimlet.

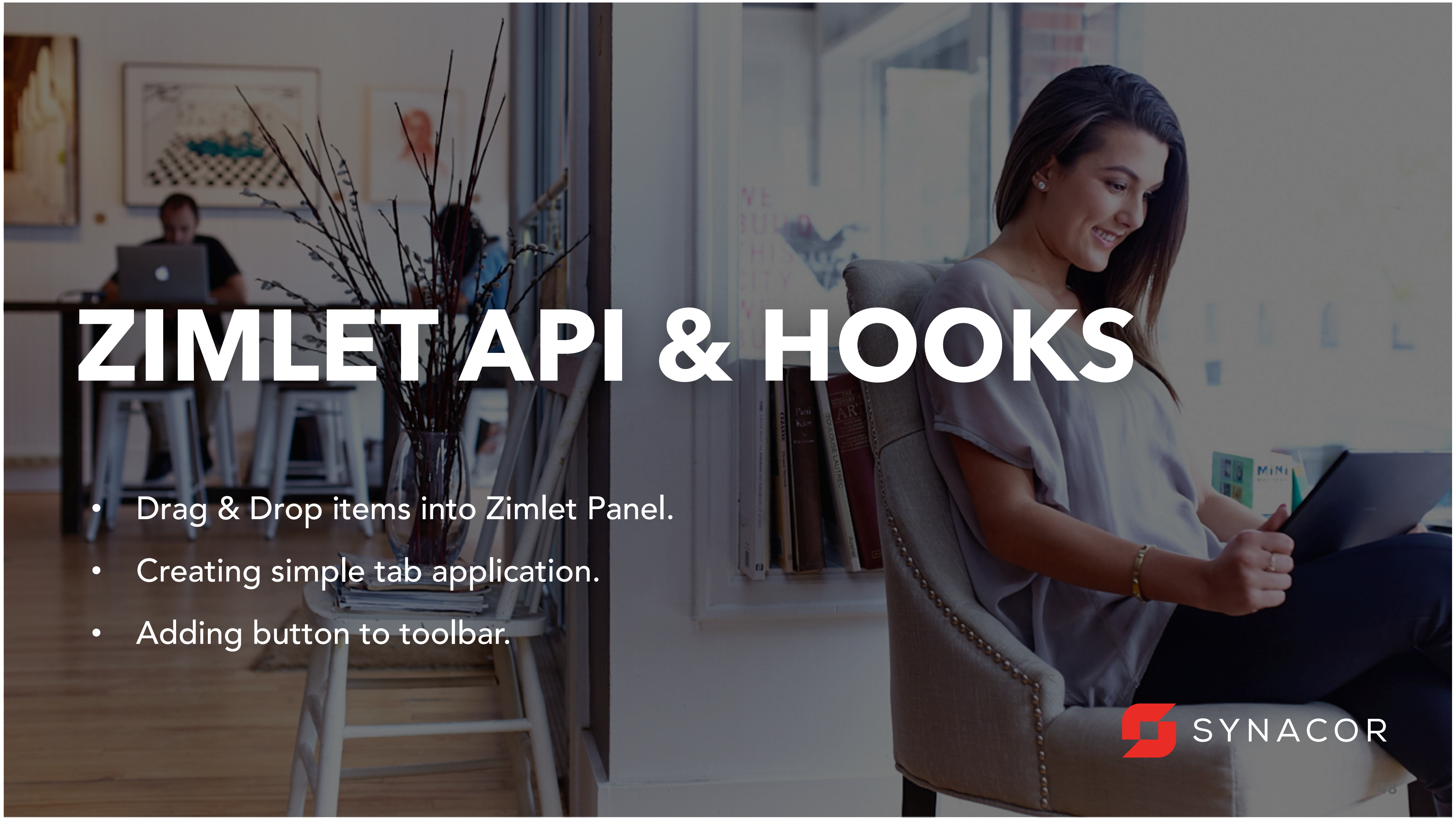
User Properties can be used to:

- Create custom user level preferences stored at server.
- Create feature preferences.
- Store user specific persistent data.

If a zimlet defines user properties and the zimlet has a <**zimbraPanelItem**>, a property editor dialog is presented when the panel item is double-clicked.


Demos init, click, dbl-click and rt-click on zimlet icon preferences

String: das
Priority?: ☐
Delivery Date: 7/10/17
Your ID number: 2
List: List Item 1



ZIMLET API & HOOKS

- Drag & Drop items into Zimlet Panel.
- Creating simple tab application.
- Adding button to toolbar.

SYNACOR

DIALOGS

SYNACOR

WRITING ADMIN EXTENSIONS

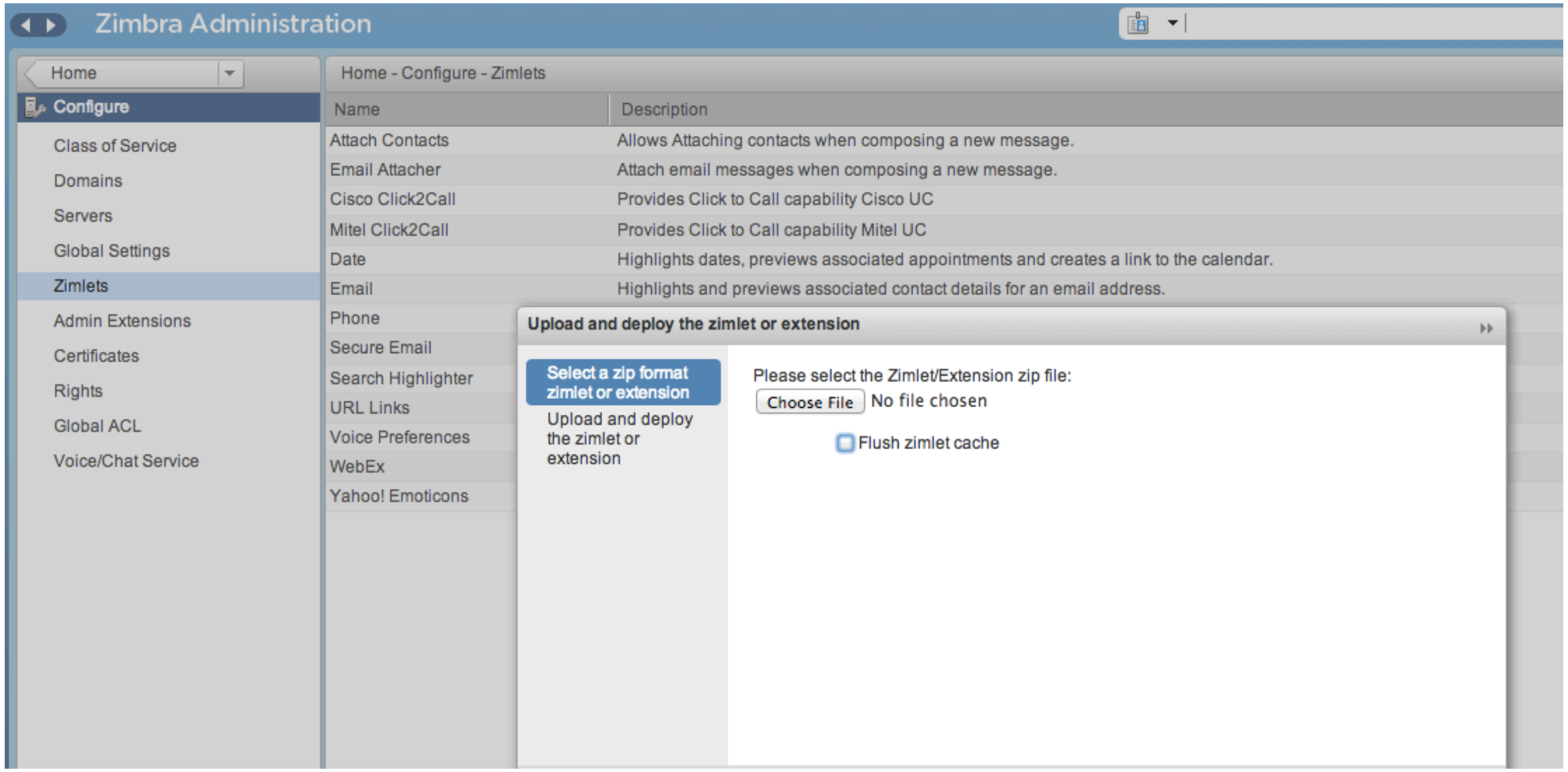
- https://wiki.zimbra.com/wiki/Extending_Admin_UI

MANAGING ZIMLETS

SYNACOR

USING ADMIN CONSOLE

- Login to admin console > Configure > Zimlets > Deploy
- Remember to select Flush Zimlet cache checkbox if you are upgrading
- Best practice: Always uninstall Zimlet before upgrading



Name	Description
Attach Contacts	Allows Attaching contacts when composing a new message.
Email Attacher	Attach email messages when composing a new message.
Cisco Click2Call	Provides Click to Call capability Cisco UC
Mitel Click2Call	Provides Click to Call capability Mitel UC
Date	Highlights dates, previews associated appointments and creates a link to the calendar.
Email	Highlights and previews associated contact details for an email address.
Phone	
Secure Email	
Search Highlighter	
URL Links	
Voice Preferences	
WebEx	
Yahoo! Emoticons	

Please select the Zimlet/Extension zip file:
No file chosen

☐ Flush zimlet cache

MANAGING ZIMLETS USING CLI

zmzimletctl - This command is used to manage Zimlets and to list all zimlets on the server.

su – zimbra - Run as Zimbra

zmzimletctl deploy <path to zimlet.zip>

-This will registers zimlet information in LDAP, installs the zimlet, adds the zimlet to default COS and turns on the zimlet. It adds the Zimlet files to zimlets-deployed(opt/zimbra/zimlets-deployed <zimletName>) folder

zmzimletctl undeploy <just the zimlet name>

-This will undeploy the Zimlet

MANAGING ZIMLETS (CONTD.)

zmzimletctl listZimlets

-Lists all available Zimlets in 3 sections: Host, LDAP and COS .

-Host: indicates all those Zimlets that are installed (not necessarily deployed (in ldap)) and basically means, they are physically present in zimlets-deployed folder.

-LDAP: Shows the list of Zimlets that actually registered in LDAP

-COS <cosname>: Shows the list of Zimlets that are turned ON

Notes:

- For Zimlet to work, it must be listed on all three areas.
- In multimode environment, you will see these three sections for each node.

MANAGING ZIMLETS (CONTD.)

Example:

Installed Zimlet files on this host:

com_zimbra_date
com_zimbra_email
com_zimbra_attachcontacts
com_zimbra_oldzimlet

Installed Zimlets in LDAP:

com_zimbra_date
com_zimbra_email
com_zimbra_attachcontacts

Available Zimlets in COS:

default:

com_zimbra_date
com_zimbra_email

In the above example, users on 'default' COS:

-Have access to com_zimbra_date and com_zimbra_email.

-Don't have access to com_zimbra_attachcontacts zimlet (its deployed but is turned OFF)

MANAGING ZIMLETS (CONTD.)

zmprov fc zimlet

Flushes Zimlet cache. Use this whenever you want to make sure to clear stale zimlet information.

zmzimletctl listPriority

List the priorities of all Zimlets.

zmzimletctl setPrority

Set a Priority to a Zimlet. Priorities can be set to Zimlets to ensure that a particular Zimlet to be used over another similar zimlet. Like: You might have two different phone zimlets using two different services. You can use priority to make sure which one should be used.

For all other zmzimletctl commands:

zmzimletctl help

Zimlet Development

CONFIGURING A DEVELOPER ENVIRONMENT

SYNACOR

DEVELOPER ENVIRONMENT FOR ZIMLETS

_dev folder for Zimlets

- Create a folder _dev under /opt/zimbra/zimlets-deployed folder and keep the zimlet under development within that. For example: /opt/zimbra/zimlets-deployed/_dev/com_zimbra_test
- Any Zimlet under _dev folder will automatically be picked up by Zimbra without actual deployment.
- Any changes to this file will be automatically reflected when the browser is refreshed.
- Files are not obfuscated and merged with other Zimlet files
- JSP files can be placed within the Zimlet folder. It will be compiled every time browser is refreshed
- Jar files should be manually copied to /opt/zimbra/jetty/webapps/zimlet/WEB-INF/ folder
- If .class file is used, it must be manually copied to /opt/zimbra/jetty/webapps/zimlet/WEB-INF/<path of the package>/<dot-class file>

DEPLOYING ZIMLET USING CLI

This is mainly useful when ZCS Server is remote or inside a Virtual Machine.

- Zip all the files inside the zimlet in a zip file. (Zimlet files should directly be inside zip). Ex: *zip –r com_zimbra_test.zip* *
- Optionally you can do zmzimletctl fc zimlet to flush zimlet cache from Zimbra Server.
- zmzimletctl deploy path-to-your-zimlet/com_zimbra_test.zip

DEVELOPER MODE

?dev=1 (developer mode for Core-Zimbra Web Client)

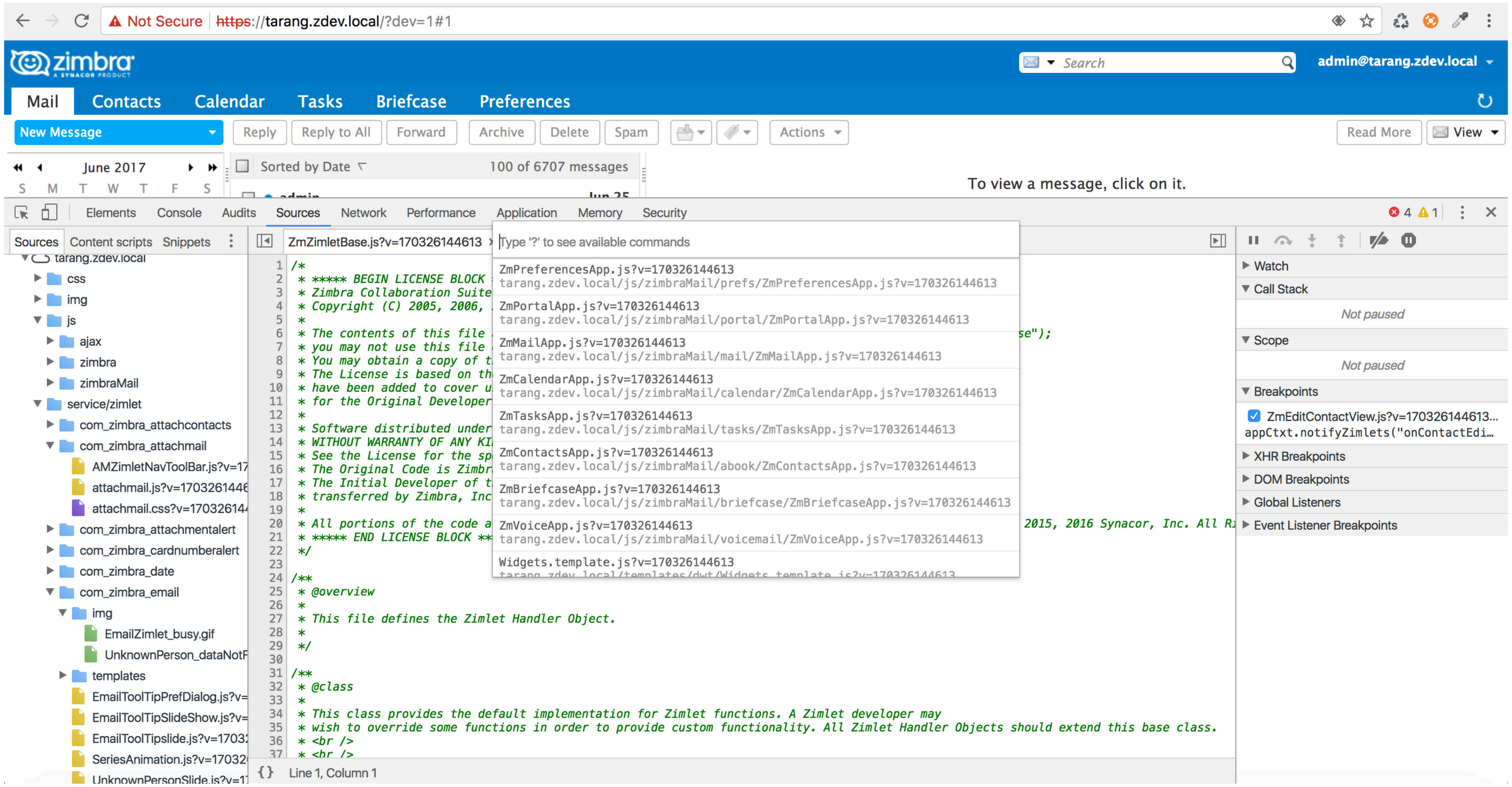
- Open Zimbra Web Client with ?dev=1 attribute.
  E.g.: *http://www.zimbramail.com/zimbra/?dev=1*

- Zimbra loads all JavaScript files individually and doesn't obfuscate them.

- Zimbra loads JavaScript files of all applications even if they are not going to be used (no lazy loading).

- Much **slower** but helps in debugging

DEVELOPER MODE



Not Secure | https://tarang.zdev.local/?dev=1#1

```
tarang.zdev.local
  css
  img
  js
    ajax
    zimbra
    zimbraMail
  service/zimlet
    com_zimbra_attachcontacts
    com_zimbra_attachmail
      AMZimletNavToolBar.js?v=17
      attachmail.js?v=17032614
      attachmail.css?v=17032614
    com_zimbra_attachmentalert
    com_zimbra_cardnumberalert
    com_zimbra_date
    com_zimbra_email
      img
        EmailZimlet_busy.gif
        UnknownPerson_dataNotF
      templates
      EmailToolTipPrefDialog.js?v=
      EmailToolTipSlideShow.js?v=
      EmailToolTipslide.js?v=1703
      SeriesAnimation.js?v=17032
      UnknownPersonSlide.js?v=1
```

```
/*
 * ***** BEGIN LICENSE BLOCK
 * Zimbra Collaboration Suite
 * Copyright (C) 2005, 2006,
 *
 * The contents of this file                                     se");
 * you may not use this file
 * You may obtain a copy of t
 * The License is based on th
 * have been added to cover u
 * for the Original Developer
 *
 * Software distributed under
 * WITHOUT WARRANTY OF ANY KI
 * See the License for the sp
 * The Original Code is Zimbr
 * The Initial Developer of t
 * transferred by Zimbra, Inc
 *
 * All portions of the code a                                    2015, 2016 Synacor, Inc. All Ri
 * ***** END LICENSE BLOCK **
 */

/**
 * @overview
 *
 * This file defines the Zimlet Handler Object.
 *
 */

/**
 * @class
 *
 * This class provides the default implementation for Zimlet functions. A Zimlet developer may
 * wish to override some functions in order to provide custom functionality. All Zimlet Handler Objects should extend this base class.
 * <br />
 * <br />
```

```
Type '?' to see available commands
ZmPreferencesApp.js?v=170326144613
tarang.zdev.local/js/zimbraMail/prefs/ZmPreferencesApp.js?v=170326144613
ZmPortalApp.js?v=170326144613
tarang.zdev.local/js/zimbraMail/portal/ZmPortalApp.js?v=170326144613
ZmMailApp.js?v=170326144613
tarang.zdev.local/js/zimbraMail/mail/ZmMailApp.js?v=170326144613
ZmCalendarApp.js?v=170326144613
tarang.zdev.local/js/zimbraMail/calendar/ZmCalendarApp.js?v=170326144613
ZmTasksApp.js?v=170326144613
tarang.zdev.local/js/zimbraMail/tasks/ZmTasksApp.js?v=170326144613
ZmContactsApp.js?v=170326144613
tarang.zdev.local/js/zimbraMail/abook/ZmContactsApp.js?v=170326144613
ZmBriefcaseApp.js?v=170326144613
tarang.zdev.local/js/zimbraMail/briefcase/ZmBriefcaseApp.js?v=170326144613
ZmVoiceApp.js?v=170326144613
tarang.zdev.local/js/zimbraMail/voicemail/ZmVoiceApp.js?v=170326144613
Widgets.template.js?v=170326144613
```

Line 1, Column 1

```
Watch
Call Stack
    Not paused
Scope
    Not paused
Breakpoints
    ☑ ZmEditContactView.js?v=170326144613…
    appCtxt.notifyZimlets("onContactEdi…
XHR Breakpoints
DOM Breakpoints
Global Listeners
Event Listener Breakpoints
```

PRODUCTION MODE

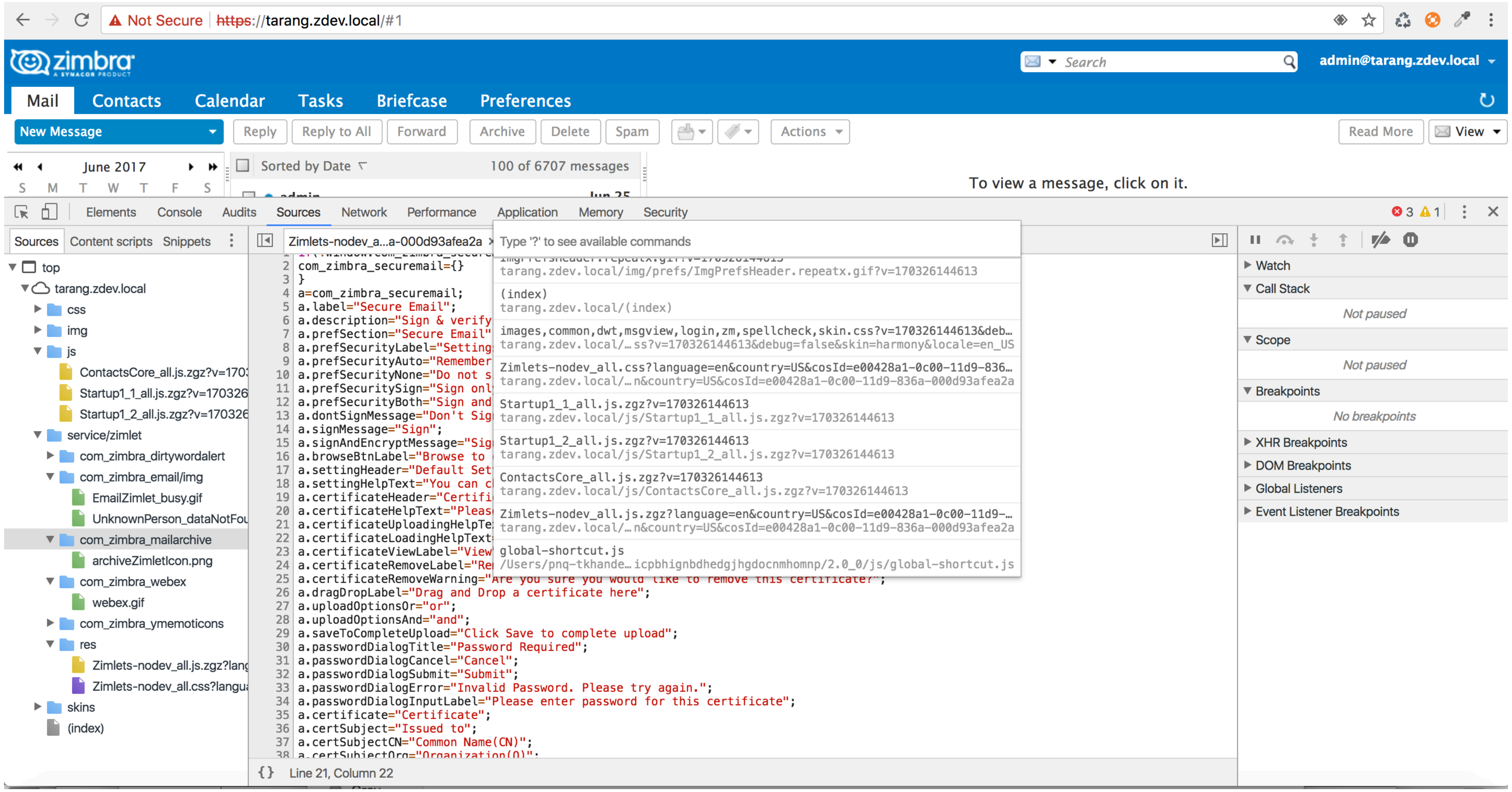
Lazy loading is ON. i.e. Zimbra loads only those packages that are necessary.

- For example: Zimbra loads all JS files for Mail App, but not Calendar App. It will lazily loads JS required for Calendar App only when the user actually clicks on Calendar tab.

We merge, zip and obfuscate (remove comments, shorten function names etc) to reduce JS footprint in order to quickly load Zimbra.

- For example: SendEmail(param1, param2) might look like A(p1, p2)
- All deployed Zimlets are merged in Zimlets-nodev_all.js

PRODUCTION MODE



```
com_zimbra_securemail={}
}
a=com_zimbra_securemail;
a.label="Secure Email";
a.description="Sign & verify
a.prefSection="Secure Email"
a.prefSecurityLabel="Setting
a.prefSecurityAuto="Remember
a.prefSecurityNone="Do not s
a.prefSecuritySign="Sign onl
a.prefSecurityBoth="Sign and
a.dontSignMessage="Don't Sig
a.signMessage="Sign";
a.signAndEncryptMessage="Sig
a.browseBtnLabel="Browse to
a.settingHeader="Default Set
a.settingHelpText="You can c
a.certificateHeader="Certifi
a.certificateHelpText="Pleas
a.certificateUploadingHelpTe
a.certificateLoadingHelpText
a.certificateViewLabel="View
a.certificateRemoveLabel="Re
a.certificateRemoveWarning="Are you sure you would like to remove this certificate?";
a.dragDropLabel="Drag and Drop a certificate here";
a.uploadOptionsOr="or";
a.uploadOptionsAnd="and";
a.saveToCompleteUpload="Click Save to complete upload";
a.passwordDialogTitle="Password Required";
a.passwordDialogCancel="Cancel";
a.passwordDialogSubmit="Submit";
a.passwordDialogError="Invalid Password. Please try again.";
a.passwordDialogInputLabel="Please enter password for this certificate";
a.certificate="Certificate";
a.certSubject="Issued to";
a.certSubjectCN="Common Name(CN)";
```

```
Type '?' to see available commands
tarang.zdev.local/img/prefs/ImgPrefsHeader.repeatx.gif?v=170326144613
(index)
tarang.zdev.local/(index)
images,common,dwt,msgview,login,zm,spellcheck,skin.css?v=170326144613&deb…
tarang.zdev.local/…ss?v=170326144613&debug=false&skin=harmony&locale=en_US
Zimlets-nodev_all.css?language=en&country=US&cosId=e00428a1-0c00-11d9-836…
tarang.zdev.local/…n&country=US&cosId=e00428a1-0c00-11d9-836a-000d93afea2a
Startup1_1_all.js.zgz?v=170326144613
tarang.zdev.local/js/Startup1_1_all.js.zgz?v=170326144613
Startup1_2_all.js.zgz?v=170326144613
tarang.zdev.local/js/Startup1_2_all.js.zgz?v=170326144613
ContactsCore_all.js.zgz?v=170326144613
tarang.zdev.local/js/ContactsCore_all.js.zgz?v=170326144613
Zimlets-nodev_all.js.zgz?language=en&country=US&cosId=e00428a1-0c00-11d9-…
tarang.zdev.local/…n&country=US&cosId=e00428a1-0c00-11d9-836a-000d93afea2a
global-shortcut.js
/Users/pnq-tkhande…icpbhignbdhedgjhgdocnmhomnp/2.0_0/js/global-shortcut.js
```

ADDITIONAL TIPS

- You can use _dev Zimlet mode while you are still using Zimbra's Production mode (i.e. no ?dev=1).
    - In this case, JavaScript files that are part of Zimlet are not obfuscated while that of core-Zimbra and all other Zimlets (that are deployed) will be.
    - This *greatly* helps to cut down development time since Zimbra loads much faster (obfuscates all core JavaScript files).
- If you are calling some functions say from Contacts or Calendar etc in your Zimlet, make sure to load that package because it might work in Dev mode but would break in Production mode.
- Make sure to not have **debugger** statements in production mode.
- **Always make sure to test your Zimlets in production mode.**

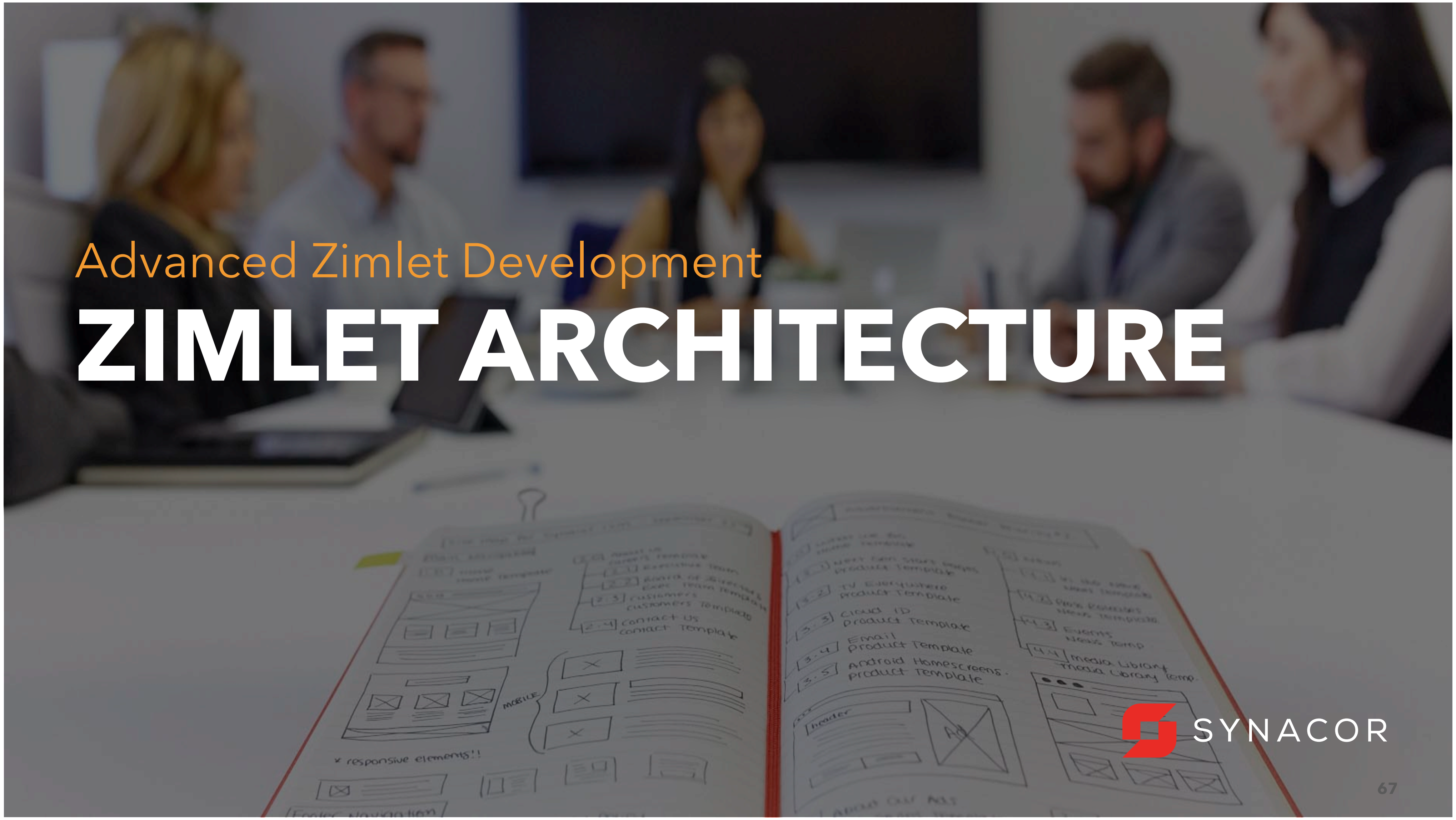
Debugging/Development

CLIENT SIDE DEBUGGING

SYNACOR

THINGS TO USE WHILE DEBUGGING CLIENT SIDE CODE

- Preferably use dev mode i.e. ?dev=1.
- JavaScript Debug point & track the call stack .
- XHR breakpoints
- Conditional Breakpoints
- DOM breakpoints
- Event listener breakpoints.



Advanced Zimlet Development

ZIMLET ARCHITECTURE

SYNACOR

TYPICAL ZIMLET STRUCTURE

- Zimlet description xml file(e.g. com_zimbra_test.xml)

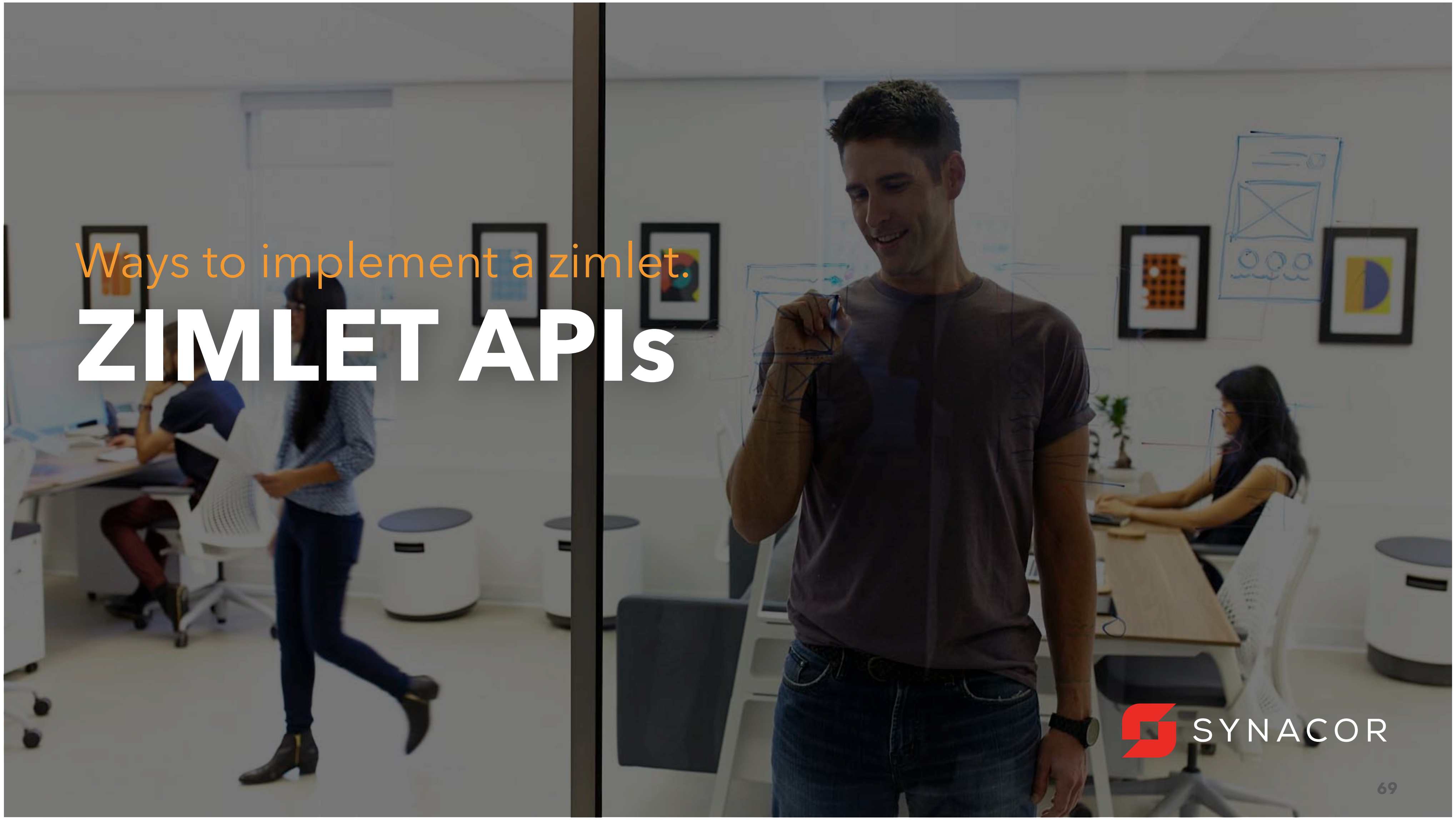
  Zimlet information, things like: Zimlet name, Zimlet version, user properties
- A JavaScript file (e.g. com_zimbra_test.js)

  This is the main file that does all the zimlet hard work.
- CSS file(e.g. com_zimbra_test.css)

  Style-sheet that you might use to style a Zimlet dialog/form fields or anything.
- config_template.xml file

  Contains configuration information like "allowedddomains " property & global configurations.

  * **Optional**: icon/images, *.jsp file, *.template file, *.properties i18N file



Ways to implement a zimlet.

ZIMLET APIs

SYNACOR

WAYS TO IMPLEMENT A ZIMLET

Using only XML API

- You can do some of the basic actions using just the xml-api although enough for few content Zimlets
- Limited set of features(e.g. you can't design your own dialog box or create a Zimbra entity(like a folder, tag etc)
- No debugging & almost impossible to write Extension Zimlets.

Using JavaScript & XML API

- You can write some really complex Zimlets with relative ease (having Zimbra written in pure JavaScript.)
- Browsers provide excellent debugging tools. (Ex: Chrome DevTools for Chrome)
- You can still use the xml api for certain purposes like creating a panel item, context-menus etc. (i.e. a mix of the XML and JS APIs)

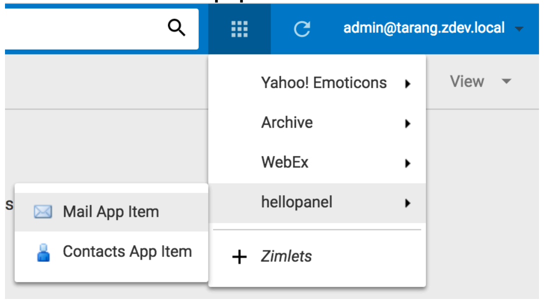
ZIMLET XML API

SYNACOR

EXAMPLE: ZIMLET DEFINITION FILE

Overview Panel Zimlet with click, double click and menu selected options (com_zimbra_hellopanel)

```
<zimlet name="com_zimbra_hellopanel" version="0.1” label=“Hello Panel Zimlet”
description="Demos init, click, dbl-click and rt-click on zimlet icon">
  <include>hellopanel.js</include> //include JS file to handle click, dblclick etc
  <includeCSS>hellopanel.css</includeCSS> // include CSS file to store icon’s CSS
  <handlerObject>com_zimbra_hellopanel</handlerObject> // include main Zimlet JS class name
  <zimletPanelItem label="hellopanel" icon="hellopanel-panelIcon"> //Add a panel icon(CSS name)
   <toolTipText>Click,dblclick and rtclick on me</toolTipText> // tooltip for panel icon
                 <contextMenu>  //add context menu with two items
                        <menuItem label="Mail App Item" icon="MailApp" id="hellopanel_mailAppId"/>
                        <menuItem label="Contacts App Item" icon="ContactsApp"
id="hellopanel_contactsAppId"/>
                </contextMenu>
  </zimletPanelItem>
</zimlet>
```



BASIC ZIMLET DEFINITION XML TAGS: <ZIMLET>

It is the enclosing element in the definition file.

Attributes:

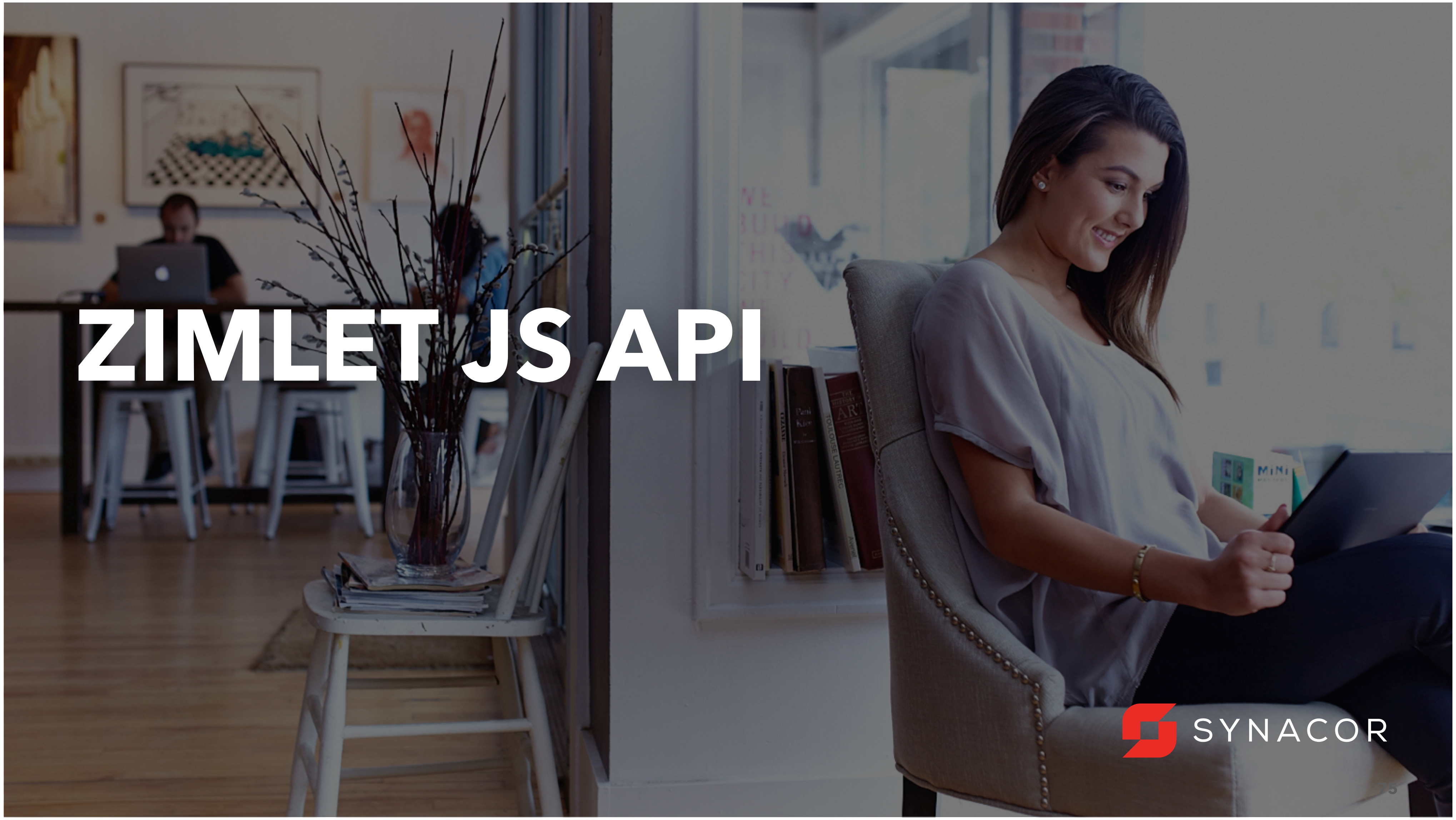
- *name (required)*: The name attribute is the Zimlet's name. The Zimlet name is required to be unique within a deployment & should match with XML file name.
- version (required): Specifies the Zimlet's version.
- Description: This attribute provides a short (approximately one line) description used in mouse-overs and dialogs.
- label: Specifies the display name for the zimlet. (Ex: Name in preferences).
- **target:** Used to specify the view target where zimlet should be loaded.
    - *main* (default): Main Zimbra Web Client Application.
    - *compose-window*: Compose mail in new window.
    - *view-window*: Viewing mail in new window.

ZIMLET DEFINITION FILE: ADDITIONAL TAGS

- <**include**>: Indicates JavaScript files used by the Zimlet. The listed JavaScript files are automatically loaded by the Zimlet framework in the order specified in the definition file.
- <**includeCSS**>: Indicates CSS style sheet files used by the Zimlet. The listed files are automatically loaded by the Zimlet framework in the order specified.
- <**resource**>: Indicates additional resource files, such as GIF or JPEG images.
- <**handlerObject**>: Name of the top level JavaScript object. The object must declare ZimletBase as the prototype. The object is automatically instantiated by Zimlet framework.
- <**zimletPanelItem**>: Creates an entry in the Zimlet panel area.

  Supported attributes: label & icon.

  May contain following elements: toolTipText, dragSource, contextMenu.
- <**userProperties**>: Lists per-user properties used by the Zimlet using <property>.



ZIMLET JS API

SYNACOR

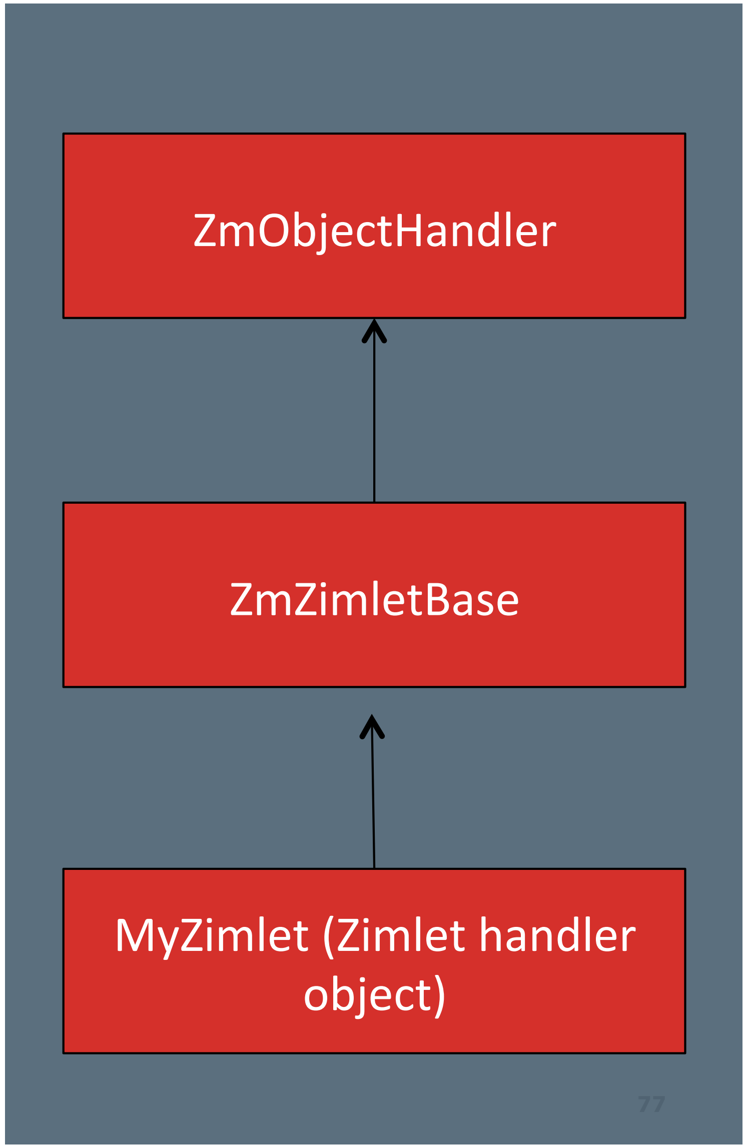
HANDLER OBJECT JS FILE

```
//Constructor.
function com_zimbra_myzimlet_HandlerObject() {
};

com_zimbra_myzimlet_HandlerObject.prototype = new ZmZimletBase();
com_zimbra_myzimlet_HandlerObject.prototype.constructor =
com_zimbra_myzimlet_HandlerObject;

//Simplify handler object notation.
var MyZimlet = com_zimbra_myzimlet_HandlerObject;

//Initializes the zimlet.
MyZimlet.prototype.init = function() {
   // do something...
};
```

ZIMLET JS CLASS HIERARCHY

- The Zimlet handler function extends the ZmZimletBase JS class.
  - **init()** called only once during initialization.
  - Include the JS file for the Handler Object.
  - Specify the Zimlet JS Handler Object in the Zimlet Definition File.

```
<zimlet name="com_zimbra_myzimlet" version="1.0”
description=”My Zimlet">
    <include>com_zimbra_myzimlet.js</include>
    <handlerObject>com_zimbra_myzimlet_HandlerObject</
handlerObject>
</zimlet>
```



SINGLE-CLICK COMPARISON

XML API

```
<zimlet name="com_zimbra_hello4" version="1.0"
description="com_zimbra_hello4: Panel Item with Clicks">
  <zimletPanelItem label="Test Panel Item">
    <onClick>
      <canvas type="window" width="300" height="300" />
      <actionUrl method="get"  target="http://maps.google.com" />
    </onClick>
  </zimletPanelItem>
</zimlet>
```

JS API

```
<zimlet name="com_zimbra_clicks" version="0.1" label="Clicks" description="Shows a
single-click in JS.">
<include>com_zimbra_clicks.js</include>
<handlerObject>com_zimbra_clicks_HandlerObject</handlerObject>
<zimletPanelItem label="Clicks">
<toolTipText>Click in JS</toolTipText>
</zimletPanelItem>
</zimlet>
```

```
// Called on single click.
ClicksZimlet.prototype.singleClicked = function() {
     window.open ("http://maps.google.com", ”google_maps”,
       "width=300,height=300");
};
```

Zimbra Global Object

APPCTXT

SYNACOR

APPLICATION CONTEXT

appCtxt (ZmAppCtxt.js) is a global object that provides access to various applications, dialog boxes and also centrally stores information about current application and about current user information, preferences etc.

When an application is loaded and active:

appCtxt.getCurrentApp() – returns current application object
appCtxt.getCurrentAppName() –returns current app's name
appCtxt.getCurrentController() –returns current app's controller

Several Zimbra's widgets also register themselves to appCtxt

appCtxt.getNewFolderDialog().popup() - shows Folder dialog widget
appCtxt.getNewTagDialog().popup() – shows New Tag dialog widget

Access Account information:

appCtxt.getActiveAccount() – returns active account
appCtxt.getActiveAccount().settings –returns all preferences/settings

https://files.zimbra.com/docs/zimlet/zcs/8.6.0/jsapi-zimbra-doc/symbols/ZmAppCtxt.html

ZIMLET USER PROPERTIES

SYNACOR

USER PROPERTIES

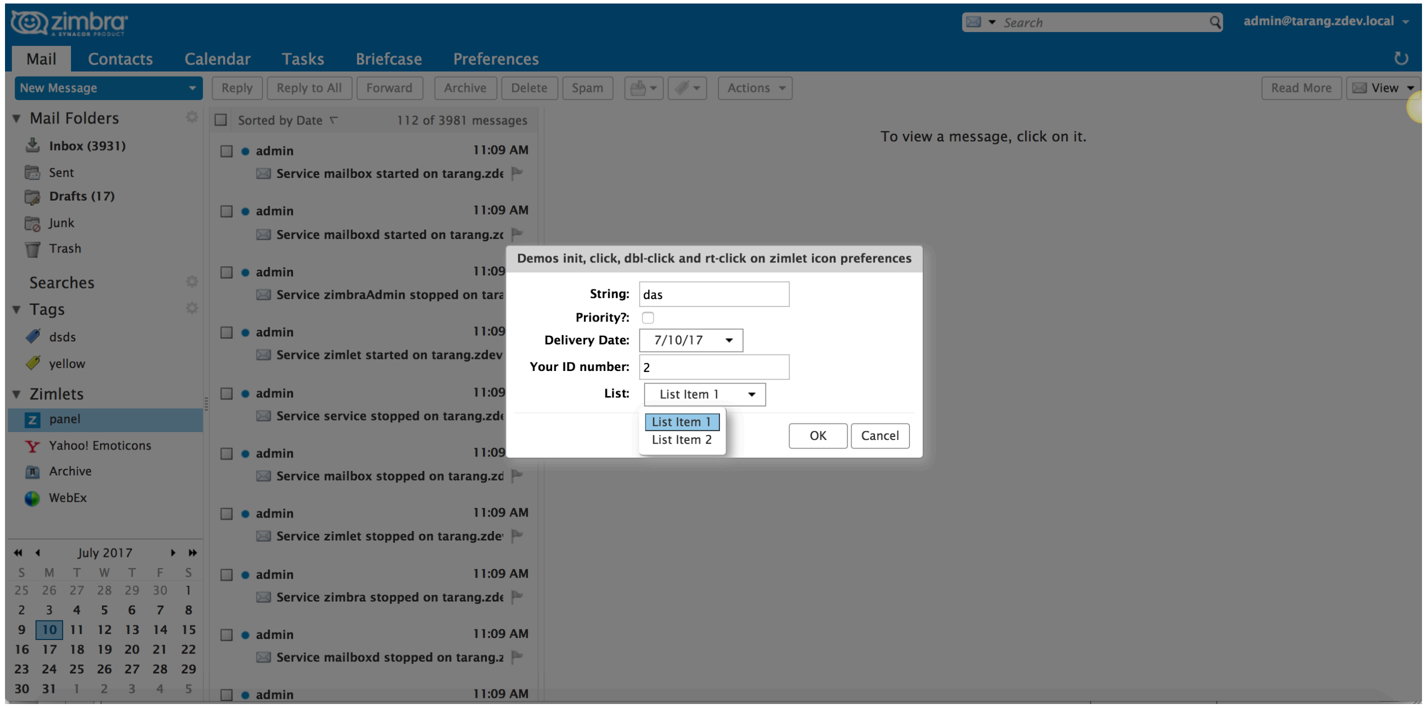
In Zimlet Definition XML file:

```
<userProperties>
   <property type="string" name="my_string" minLength="3" maxLength="3" label="String"/>
   <property type="boolean" label="Priority?" name="my_test_priority" />
   <property type="date" label="Delivery Date" name="my_test_delivery_date" />
   <property type="number" label="Your ID number" name="my_test_id_num" />
   <property type="enum" name="my_enum" label="List" value="LIST_OF_STUFF">
     <item label="List Item 1" value="LIST_ITEM_1"/>
     <item label="List Item 2" value="LIST_ITEM_2"/>
   </property>
</userProperties>
```

For More Details on attributes and schema structure: Refer DwtPropertyEditor.js

PROPERTY EDITOR (DWTPROPERTYEDITOR)

By default, when double-clicked on the Zimlet Panel Item, the property editor dialog is launched.



Demos init, click, dbl-click and rt-click on zimlet icon preferences

String:	das
Priority?:	☐
Delivery Date:	7/10/17
Your ID number:	2
List:	List Item 1

USER PROPERTIES: JS API

- getUserProperty API: Returns the value of the property

  Usage: **this.getUserProperty(“my_property”)**
- setUserProperty API: Used to set/save user properties.
  - **this.setUserProperty(“my_property”, “text”);** ← This simply sets user property in memory.
  - **this.setUserProperty(“my_property”, “text”, true);** ← This sets the user property and immediately saves the properties and won’t callback.
- saveUserProperties API: Saves all properties and returns control to the callback.

  Usage: this. saveUserProperties(callback)

  Example:

  this.saveUserProperties(new AjxCallback(this, this._propSavedHandler));

USER PROPERTIES: JS API

Can intercept properties prior to saving by implementing:

```
/**
 * This method is called by the zimlet framework prior to user properties being saved.
 *
 * @param {array} props    an array of objects with the following properties:
 *
 * props[...].label {string} the property label
 * props[...].name {string} the property name
 * props[...].type {string} the property type
 * props[...].value {string} the property value
 *
 * @return {boolean}  <code>true</code> if properties are valid; otherwise,
<code>false</code> or {String} if an error message will be displayed in the standard
error dialog.
 */
ZmZimletBase.prototype.checkProperties = function(props) { ... };
```

ZIMLET API HOOKS

SYNACOR

WHAT DOES ZIMLET HOOK MEAN ?

Hook denotes a place in Zimbra code where you dispatch an event of certain type, and if this event was registered by any Zimlet, then it would be handled by this registered function, otherwise nothing happens. This allows extensibility without exposing the core code.

Example:

- initializeToolbar
- onMsgView
- onContactEdit
- emailErrorCheck
- addCustomMimeHeaders

Create buttons, menus, dialogs etc.

ZIMLET WIDGETS

SYNACOR

GENERIC STEPS TO ADD A DWT WIDGET

1. Create a DOM element(usually DIV, sometimes TD) with a unique ID.
2. Create the widget.
3. Do document.getElementById(“id”).appendChild(widget.getHtmlElement());

Step1: A line from custom Dialog’s view

```
html[i++] = "<DIV id='hellowidget_button1'></div>";
```

Step2: Create button widget

```
var btn = new DwtButton({parent:this.getShell()});
btn.setText("Simple Button1");//button name
btn.addSelectionListener(new AjxListener(this, this._buttonListener, "Simple Button1"));
```

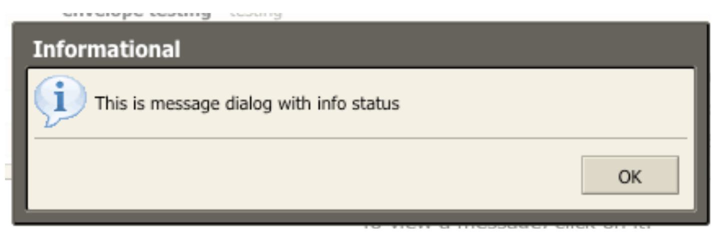
Step3: Append the widget to the DOM

```
document.getElementById("hellowidget_button1").appendChild(btn.getHtmlElement());
```

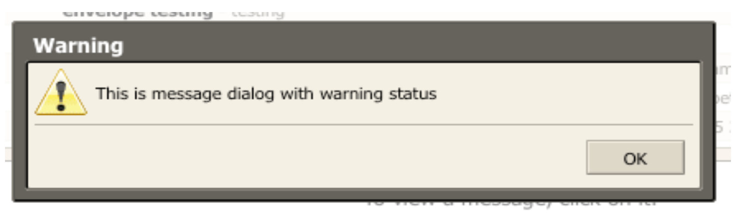
DIALOGS

- Message Dialog:
  appCtxt.getMsgDialog()
  this._dialog.setMessage(msg, style);
  Styles:
  - DwtMessageDialog.INFO_STYLE
  - DwtMessageDialog.WARNING_STYLE
  - DwtMessageDialog.CRITICAL_STYLE
- Error Dialog: appCtxt.getErrorDialog()
- Yes/No Dialog: appCtxt.getYesNoMsgDialog()
- Yes/No/Cancel Dialog: appCtxt.getYesNoCancelMsgDialog()

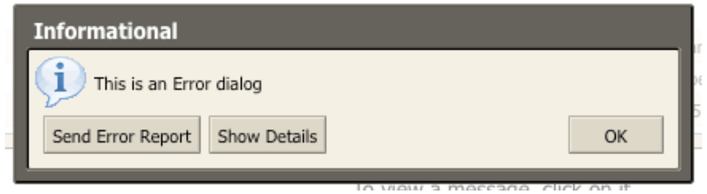
You need to call dialog.popup() to show any dialog.


Informational

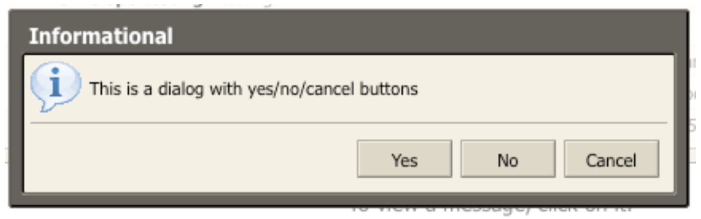
This is message dialog with info status


Warning

This is message dialog with warning status

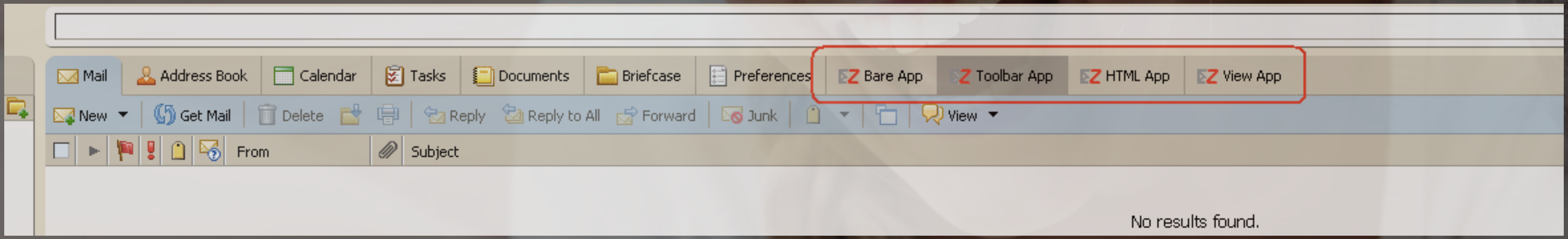

Informational

This is an Error dialog


Informational

This is a dialog with yes/no/cancel buttons

CREATING ZIMLET APPLICATION



SYNACOR

ZIMLET APPLICATION

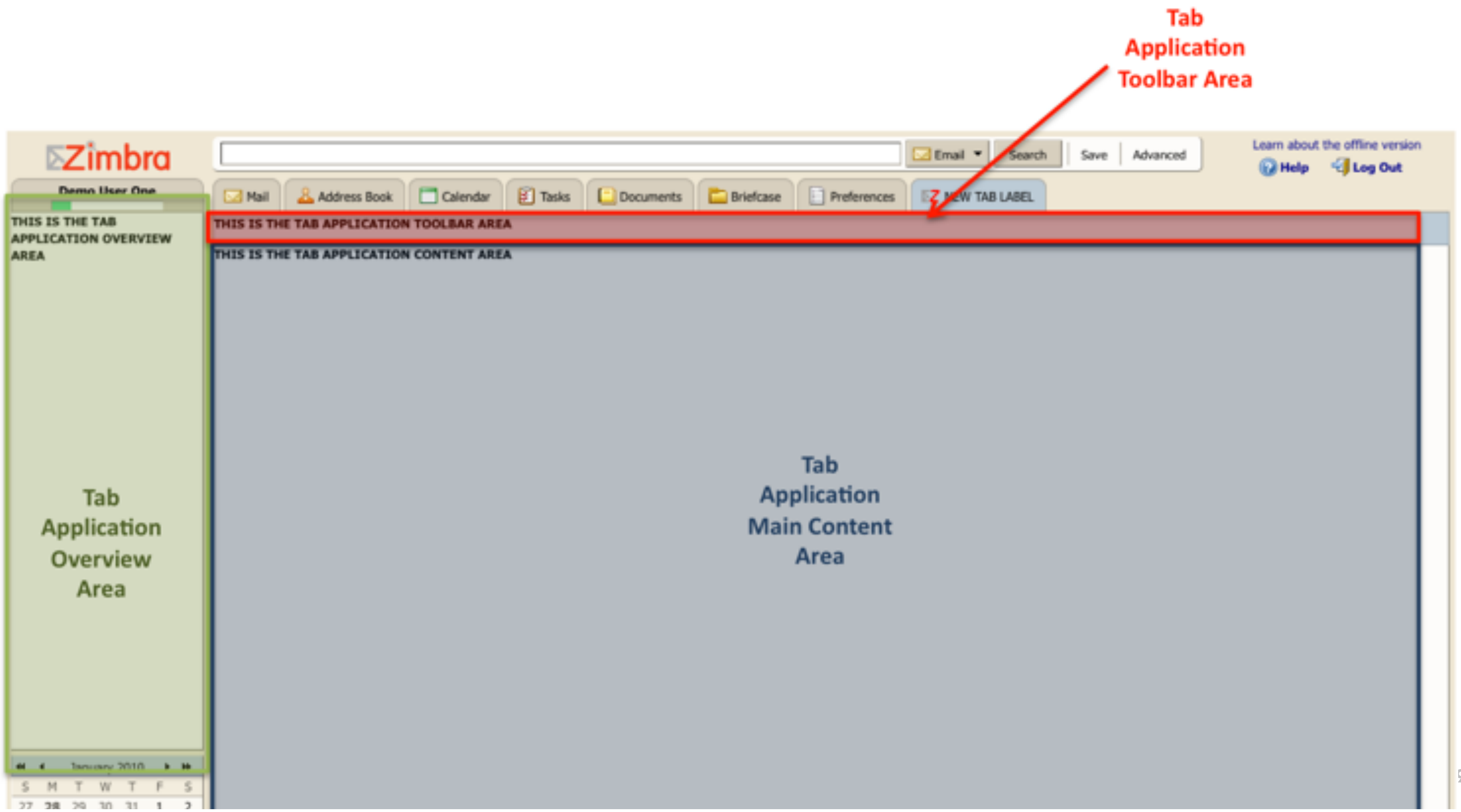
- Typical way to create a Zimlet application is by using createApp API
- [returns appIdName] = this.createApp(<visible name>, <icon>, <tooltop>);
- Since we have to show the app when user logs in to Zimbra, we call this API in "init" function

```
    com_zimbra_zimletAppExample.prototype.init = function() {
            this. _tabAppName= this.createApp("Bare App", "zimbraIcon", "Basic app in a new tab");
    }     // optional 4th argument to set the position of the tab, default last
```

After creating a zimlet application, the Zimlet handler object receives following events:

- ZmZimletBase.appActive()
- ZmZimletBase.appLaunch()

TAB AREAS: MAIN CONTENT, TOOLBAR, OVERVIEW



Tab Application Toolbar Area
Zimbra
Demo User One
Email
Search
Save
Advanced
Learn about the offline version
Help
Log Out
Mail
Address Book
Calendar
Tasks
Documents
Briefcase
Preferences
NEW TAB LABEL
THIS IS THE TAB APPLICATION OVERVIEW AREA
THIS IS THE TAB APPLICATION TOOLBAR AREA
THIS IS THE TAB APPLICATION CONTENT AREA
Tab Application Overview Area
Tab Application Main Content Area
January 2010

ZIMLET APPLICATION: APPACTIVE API

When the user clicks on the app’s tab (makes the app active) or when the user moves away from the app (makes app inactive), appActive API is called by Zimbra. The purpose is to allow us to initialize some objects or to disable some stuffs.

appActive(appName, active)
@appName : Name of the current App or Name of the app that the user is switching to.
@active: boolean indicating if the app is active or not

```
com_zimbra_zimletAppExample.prototype.appActive = function(appName, active) {
        if(!active) {
            appCtxt.setStatusMsg(['Hiding app ', appName].join("")));
        } else {
            appCtxt.setStatusMsg(['Showing app ', appName].join(""));
        }
```

ZIMLET APPLICATION: APPLAUNCHAPI

AppLaunch API is called when the app is first launched. It is called after appActive is called and is called only once. Since this API is called only once, it can be used to create all the widgets and other html contents to create the app.

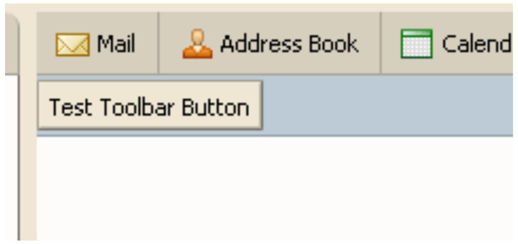
appLaunch(appName, [params])

@appName: appNameId – a unique id that's been given to the app by Zimbra

@params: Usually some search callback. You can ignore this.

<u>Example: Adding a toolbar button</u>

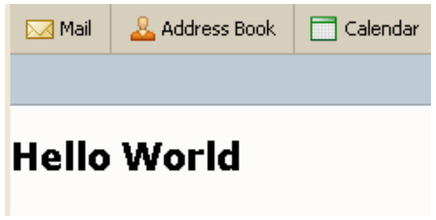
```
var app = appCtxt.getApp(appName); //get access to app from appName
var toolbar = app.getToolbar(); //returns the toolbar for the current app
toolbar.createButton("TEST", {text:"Test Toolbar Button"});
toolbar.addSelectionListener("TEST", new AjxListener(this, this._handleToolbarButton));
```



ADDING WIDGETS TO ZIMLET APP

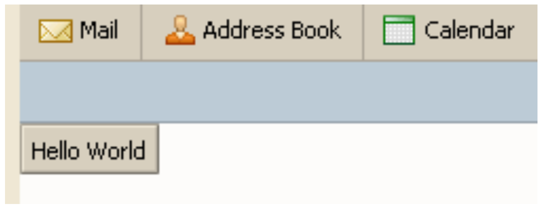
Adding Html content to the Zimlet App:

```
app.setContent("<h1>Hello World</h1>");// Used in Social Zimlet
```

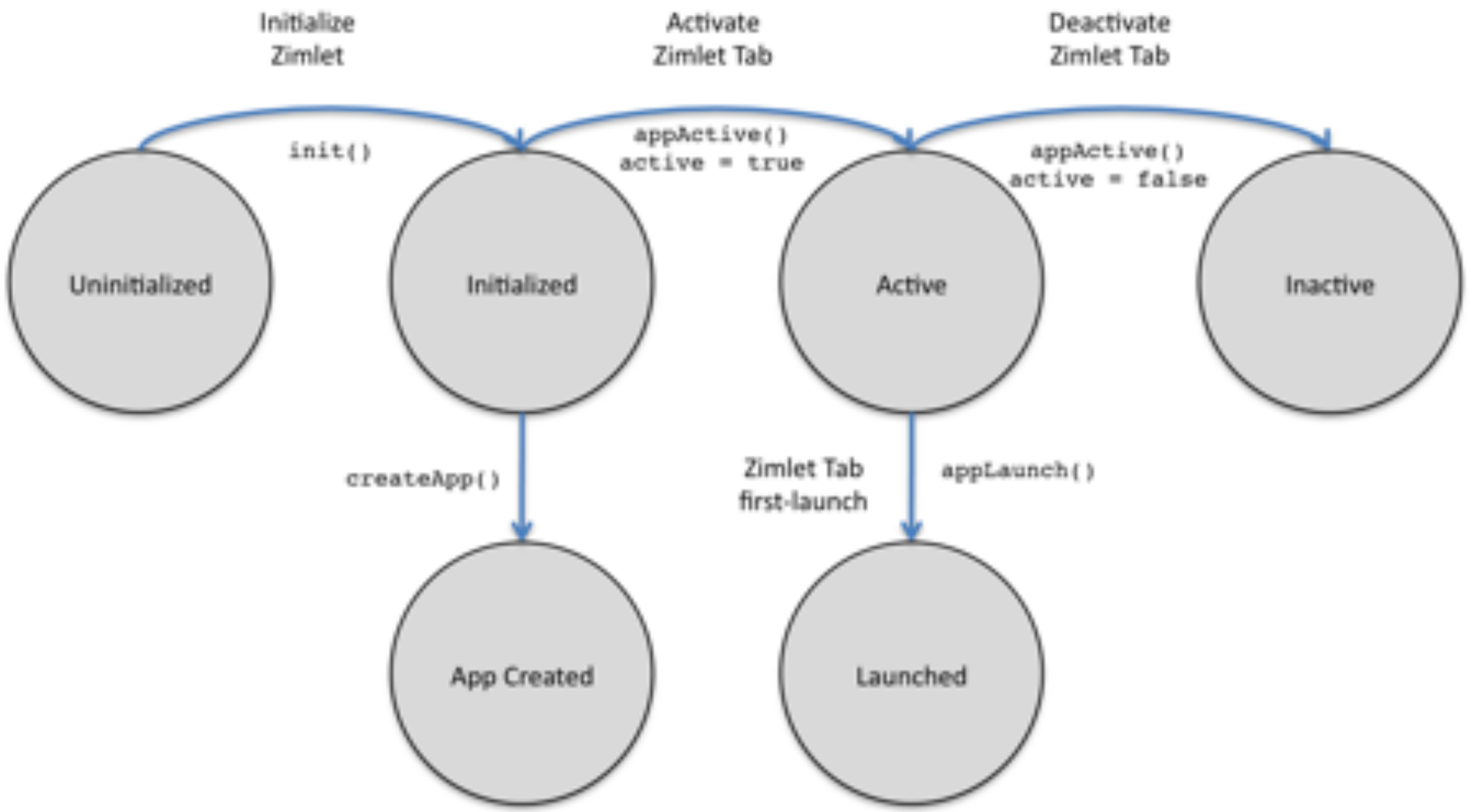


Adding Dwt Widget to the View area:

```
var button = new DwtButton({parent:DwtShell.getShell(window)});
button.setText("Hello World");
app.setView(button);
```



ZIMLET TAB LIFECYCLE



ZIMBRA SOAP/JSON API

SYNACOR

ZIMBRA SOAP API FORMATS

- **SOAP** (Simple Object Access Protocol)
  - SOAP XML formatted envelopes
  - Heavier & slower than JSON

```
<GetAccountInfoRequest
xmlns="urn:zimbraAccount">
        <account by="name">user1</account>
</GetAccountInfoRequest>
```

- **JSON** (JavaScript Object Notation)
  - Data is sent in JavaScript object form.
  - JSON requests are lightweight.

```
GetAccountInfoRequest: {
        _jsns: "urn:zimbraAccount",
        account: {
                _content: "user1",
                by: "name"
        }
}
```

TIPS/TRICKS

SYNACOR

MONKEY PATCHING: OVERRIDING UTILITY

```
com_zimbra_example.prototype.overrideAPI = function(object, funcname, newfunc) {
    newfunc = newfunc || this[funcname];
    if (newfunc) {
        var oldfunc = object[funcname];
        object[funcname] = function() {
            newfunc.func = oldfunc; // (A) can be called as arguments.callee.func from inside new function.
            return newfunc.apply(this, arguments);
        }
        object[funcname].func = oldfunc; //(B) called as object[funcname].func  from other functions.
    }
};
```

UTILITY FUNCTIONS AVAILABLE

- Enable JSP support in Zimlets: (false by default)

```
zmprov ms hostname zimbraZimletJspEnabled TRUE
```

- Reloading Zimbra Web Client

```
window.onbeforeunload = null;
var url = AjxUtil.formatUrl({});
ZmZimbraMail.sendRedirect(url);
```

- Get logged in username:

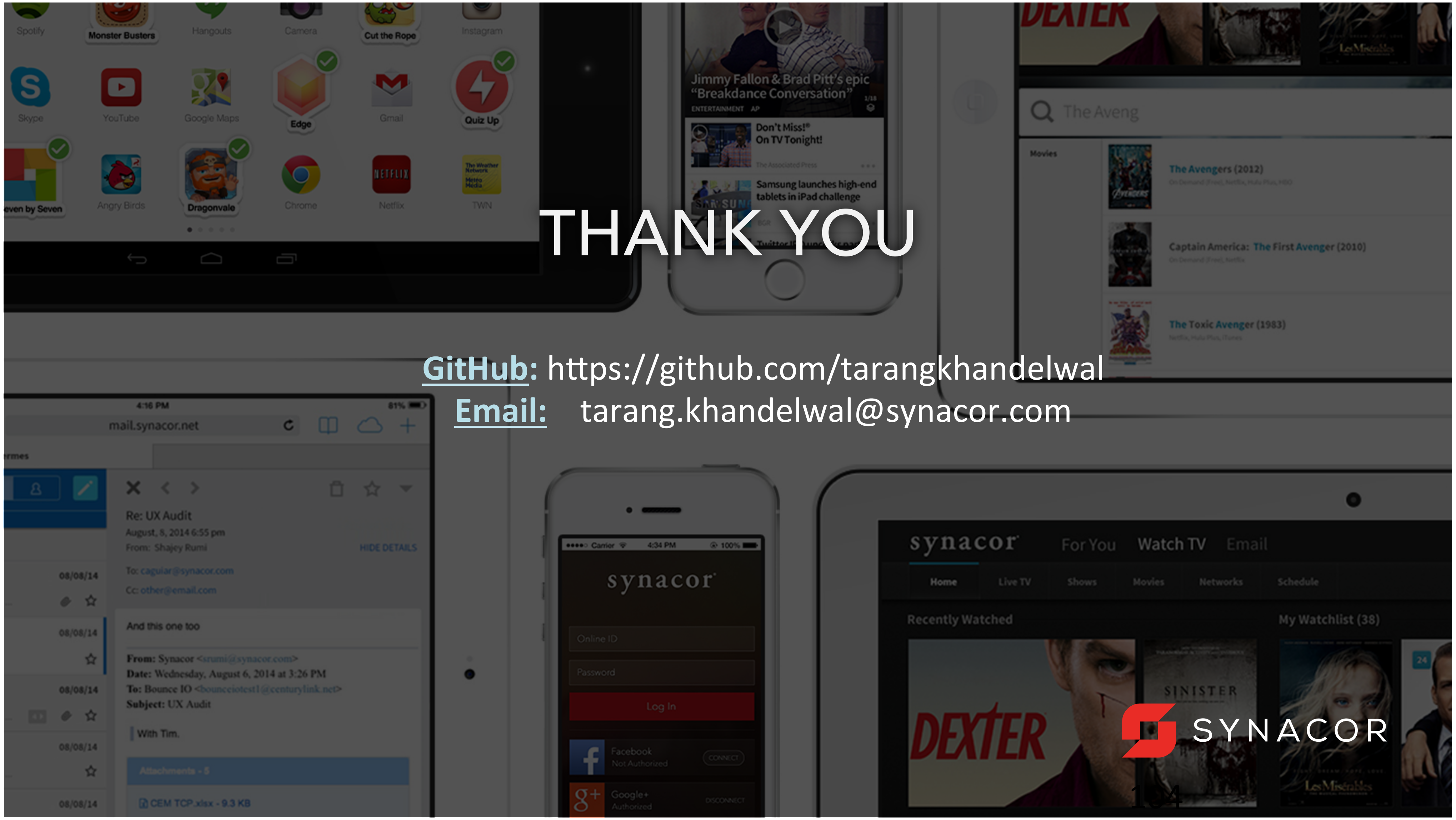
  this. getUsername() (ZmZimletBase)

- Get logged in user's zimbra internal user id:

  this. getUserID()   (ZmZimletBase)

USEFUL LINKS

- Zimbra Forums: https://forums.zimbra.org/
- Zimlet Gallery: https://zimbra.org/extend/
- Zimbra Wiki: https://wiki.zimbra.com
- Zimlet Developer Guide:
  https://wiki.zimbra.com/wiki/Zimlet_Developers_Guide:Introduction
- Zimbra JavaScript API
  https://wiki.zimbra.com/wiki/Zimlet_Developers_Guide:Zimbra_JavaScript_API_Reference
- Zimbra Community : https://github.com/Zimbra-Community



THANK YOU

GitHub: https://github.com/tarangkhandelwal
Email: tarang.khandelwal@synacor.com

SYNACOR