MIKE PASTORE

TORII FOR EMBER.JS

WHY ARE WE HERE?

- Authentication is a common problem, more-so with SPAs.
- Tough to tackle for SPA and Ember.js newbies.
- Developers want social media login, custom OAuth2 providers, **and** traditional username/password login!

WHO AM I?

- A former web developer…
- SysAdmin, DevOps, Data Engineering for past 13 years.
- Currently building a social media analytics (stealth-mode startup) front-end with Ember.js.
- All this new technology is making me feel… quixotic.

TORII IS A SET OF CLEAN ABSTRACTIONS FOR AUTHENTICATION IN EMBER.JS APPLICATIONS.

vestorly.github.io/torii

TORII IS...

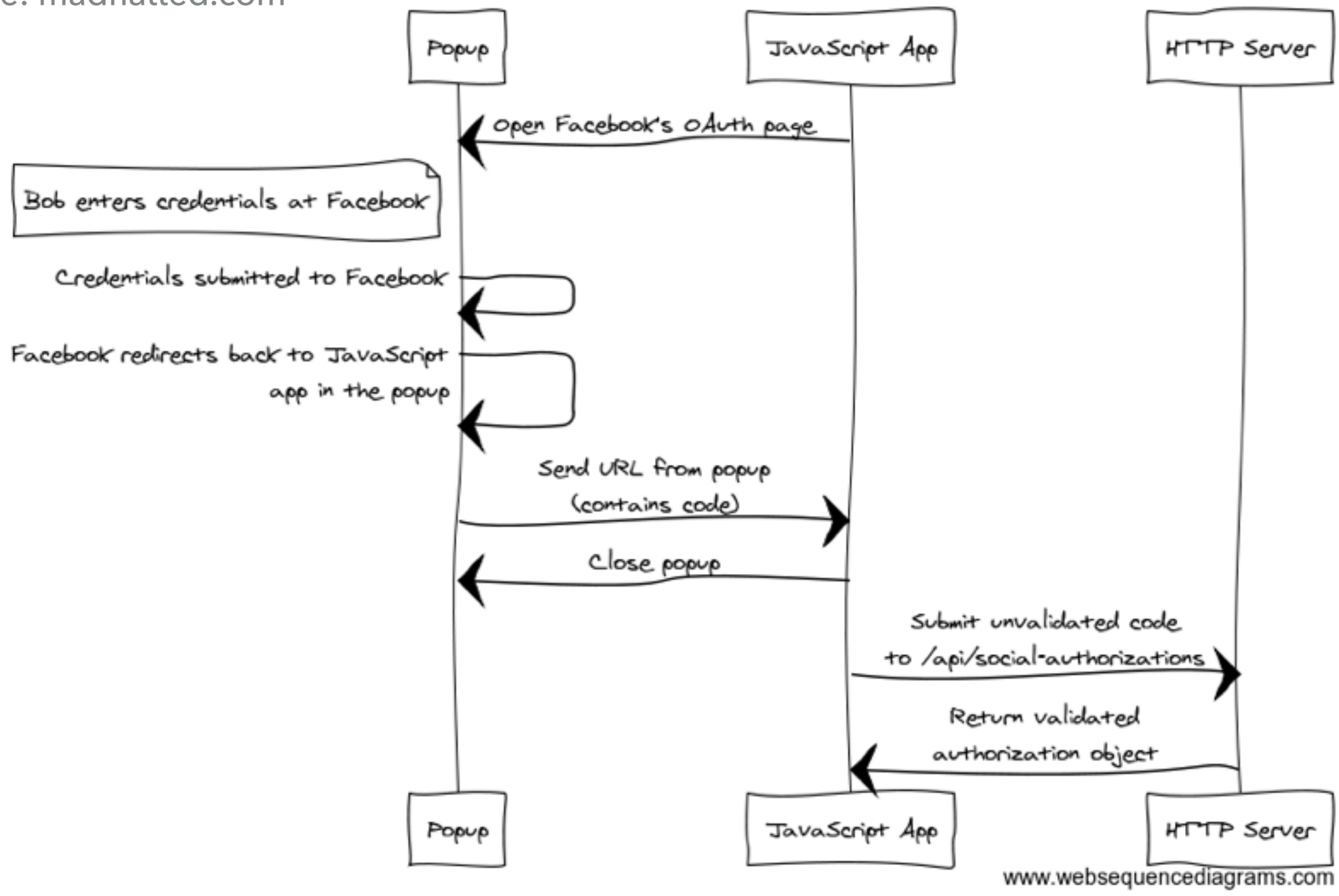
- Relatively un-opinionated about your "session" persistence strategy.
- Not limited to login, e.g. can provide Stripe services to your application, or share to Facebook or Twitter, or...
- Able to be used with or without Ember Simple Auth (note that Torii does not provide an Authorizer).

TORII GIVES YOU...

- a state machine: unauthenticated ↠ authenticating ↠ authenticated
- a method of protecting certain “authenticated” routes
- stock providers for Facebook, LinkedIn, GitHub, etc.
- base classes for defining custom providers (e.g. in-house OAuth2 or username/password login)
- “magic” for implementing OAuth1.0a (which takes place primarily server-side)
- a very simple opt-in session store (e.g. current user ID or username)

CLIENT-SIDE OAUTH2 LOGIN FLOW

Source: madhatted.com



TODAY'S SOLUTION PATTERN

- Torii (standalone) with social media login
- JWT, Cookies*, HTTPS
- XSS and XSRF prevention
- Ember Data optional
- Concepts, not code today

JSON WEB TOKENS (JWT)

- Pronounced “jot”
- Evolved from Session Tokens
- Persisted on client in cookie or localStorage key, similar to a session token
- Cryptographically signed (not encrypted)
- Securely identifies a user and encodes “session” metadata (e.g. TTL)
- Do not store sensitive information in the payload!
- For more information: http://jwt.io

> IF YOU THINK TECHNOLOGY CAN SOLVE YOUR SECURITY PROBLEMS, THEN YOU DON'T UNDERSTAND THE PROBLEMS AND YOU DON'T UNDERSTAND THE TECHNOLOGY.
>
> Bruce Schneier

CROSS-SITE SCRIPTING (XSS) PREVENTION

- Definition: Malicious code injected into web site, potentially exposing other users' private data.
- Prevention: Escape user-submitted data before display.
- For more information:
  - https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
  - https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

WORDPRESS IS AN UNAUTHENTICATED REMOTE SHELL THAT, AS A USEFUL SIDE FEATURE, ALSO CONTAINS A BLOG

azonenberg (bash.org)

CROSS-SITE REQUEST FORGERY (XSRF OR CSRF) PREVENTION

- Definition: Requests submitted on behalf of authenticated users from other web sites.
- Prevention: Synchronizer token pattern (double submit cookies).
- For more information:
  - https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)
  - https://www.owasp.org/index.php/CSRF_Prevention_Cheat_Sheet

WE HAVE ONLY TWO MODES: COMPLACENCY AND PANIC.

James R. Schlesinger

CROSS-ORIGIN RESOURCE SHARING (XORS OR CORS)

- Don’t stress out about content security!
- Use “ember server --proxy“ in development
- Mount Ember.js app at e.g. “/“ and API at e.g. “/api“ of same domain (and protocol) in production (follow ember-cli-deploy model)
- Add resources and statements for external login providers to “ENV.contentSecurityPolicy” as needed (per the console warnings)
- For more information:
  - https://www.youtube.com/watch?v=QZVYP3cPcWQ
  - http://ember-cli.github.io/ember-cli-deploy

PROGRAMMING —

@kyleve (Twitter) [gunshowcomic.com]

LOCALSTORAGE AND HTTP HEADER, OR COOKIES?

	localStorage & HTTP header	Cookies
XSS	Vulnerable	Protected (with “httpOnly” option)
XSRF	Protected	Vulnerable
Responsible for storing and attaching to outbound requests	Application	Browser
At risk from…	Buggy application code	Buggy browsers
Token leakage?	Vulnerable	Protected (with “secure” option)

COMPONENTS

- “torii” service
- “session” service (opt-in)
- “isWorking” property
- “isAuthenticated” property
- “currentUser” property
- custom properties

COMPONENTS

- “authenticatedRoute” DSL
  - “accessDenied()” callback

COMPONENTS

- providers – talk to authentication sources
  - built-in social media providers
  - subclass e.g. OAuth2 provider to implement your own
  - OAuth1.0a “magic”

COMPONENTS

- adapters – talk to your backend API
  - “open()” method
  - “fetch()” method
  - “close()” method
- Pro-Tip: ApplicationAdapter with provider-specific subclasses

RESOURCES

- Code walkthrough and demo Git repo: http://perlkour.pl/2015/09/easy-login-sessions-with-ember-js-and-torii
- Matthew Beale (@mixonic)'s introduction to Torii: http://madhatted.com/2014/6/17/authentication-for-single-page-apps
- Torii itself: http://vestorly.github.io/torii https://github.com/vestorly/torii

FUTURE TALKS

- Developing a backend for Torii
- OAuth 1.0a and Torii
- Torii for non-login operators (e.g. sharing)
- Writing custom Torii providers