Privowny

Les WebExtensions, du rêve et du cauchemar

Daniel Glazman
Co-CTO & VP Engineering

TL; DR

Pourquoi ?

- Un de nos produits est une extension navigateurs
- Elle utilise presque toute l'API WebExtension disponible
- Auparavant, nous avions une extension XUL pour Firefox
- Nous jonglons au quotidien avec :
  - Les différences d'implémentation
  - Les absences d'implémentation
  - Les limitations parfois terribles de l'API

- On voulait vous raconter ☺

Le rêve…

- Des Extensions pour tous les navigateurs
- Un modèle complet et sécurisé
- Des API interopérables

On revenait de loin…

- Premières extensions à Internet Explorer 5 en 1999
  - Évidemment, on est dans de la DLL…
- Netscape 6 avec les XPI, XUL et compagnies dès 2000
  - Firefox/Thunderbird en ont hérité
- Opera 10 en 2009
- Chrome 4 en 2010
- Safari 5 en 2010
  - Mais la notion d'extension chez Apple est… comment dire…
- Edge en 2016
  - Mais ce n'est pas encore au niveau de Chrome et Firefox

Si vous avez écrit des extensions XUL

- Dialecte XML dédié UI avec un <tree> très puissant
- Overlays, XBL, L10N, styles natifs ou *native-like*
- Tout dans le navigateur est modifiable
- Le code s'exécute au niveau de privilèges « chrome »
- Accès aux interfaces natives du navigateur
- Aucune limitation d'accès ou presque aux fenêtres, onglets, documents, etc.

➢Le rêve absolu pour développer une browser extension…
  ➢Explosion phénoménale de `addons.mozilla.org`

Et puis tout est parti en vrille…

- Abandon progressif de XUL et XBL par Mozilla
  - Ah bah oui mais est-ce que HTML est au même niveau ?
    - Bah non, évidemment, eh !
- Dominance totale du marché par Google Chrome
  - Compatibilité cruciale avec les extensions Chrome pour les autres
- Chrome 1-Process-per-Tab, Firefox Electrolysis
  - Accès directs au DOM impossible sans *proxies* coûteux
- Sécurité
  - Trajectoire similaire aux sécurités aéroportuaires…

En gros…

- **Un manifeste**
  - Un « background »
    - Browser Action
    - Page Action
    - Main popup en html
    - Dialogues en html
  - Des scripts et CSS injectés dans les pages
    - Sélection des pages
    - Frames ou pas
    - Liste des ressources accessibles
  - Des permissions
- **WebExtension API**

Le megafail des permissions

- La plupart des API WebExtension sont soumises à permission(s)
- Une extension un peu costaud va demander **plus d'une dizaine de permissions** (18 pour Privowny)
- Impossible pour l'usager de sélectionner individuellement les permissions accordées
- La plupart des Extensions disponibles n'ont pas de fallback
- Les usagers ne regardent plus…
- Cf. Android ou iOS

- Modèle foireux et sous-optimal…

WebExtension, la base

- Jeu de méthodes toutes asynchrones
  - Promises sur Firefox, callbacks sur Chrome…
- Pas d’accès aux internes du navigateur
- Sandboxing
- Interopérabilité
  - De plutôt haut niveau entre Google Chrome et Mozilla Firefox
  - De niveau franchement moyen pour Microsoft Edge
  - De pas de niveau du tout pour Apple Safari…
  - De niveau quasi-nul sur mobile (Firefox pour Android)…

La « spécification » WebExtension

- https://browserext.github.io/browserext/

- Démarrée dans un W3C Community Group par Microsoft
- Toujours en Draft
- Dernière MàJ en Juillet… 2017 !!!
- Le contributeur principal a jeté l'éponge et ne répond plus
- Le Community Group est en hibernation, voire en coma dépassé…
- Pas de standardisation de l'objet global !
- Pas de standardisation des URLs dans le scope de l'extension

- Qui a dit « Standard Interopérable » ???

State of the onion

alarms 100 100 0	bookmarks 95 86 27	browserAction 77 100 72	browserSettings 0 100 0	**clipboard** **0 100 0**
commands 60 100 0	contentScripts 0 100 0	contextualIdentities 0 100 0	cookies 100 100 77	dns 0 100 0
downloads 100 88 0	events 100 33 0	**extension** **100 61 38**	extensionTypes 30 100 20	find 0 100 0
history 91 100 0	i18n 100 100 80	identity 100 100 0	**idle** **100 100 50**	**management** **100 46 0**
menus 75 100 75	**notifications** **90 80 90**	omnibox 100 100 0	pageAction 81 100 72	permissions 100 77 0
pkcs11 0 100 0	privacy 58 100 0	proxy 0 100 0	**runtime** **97 80 47**	search 0 100 0
sessions 43 100 0	sidebarAction 0 100 0	storage 100 100 83	**tabs** **83 83 40**	theme 0 100 0
topSites 100 100 0	types 100 50 0	webNavigation 100 100 85	**webRequest** **81 100 76**	**windows** **100 100 87**

UI (élements)

- Il manque tellement de choses à html pour la UI…
  - Rien qu'un `select`/`optgroup`/`option` est un cauchemar…
- Quasi-impossible de matcher la UI système
- Exigences des Stores sur les librairies/frameworks
- Librairies/frameworks faites pour le Web et pas pour les sandboxes des extensions
- Pas de tooltip système au-delà de @title !

- Énorme régression par rapport à XUL ☹

UI (fenêtres)

- Impossible de spécifier des Window Features !
- Pas de dialogues modaux
- Pas de fenêtres non-retaillables
- `moz-extension://foobar` dans les titres !
- Edge incapable de fermer une fenêtre par programme !
- Aucun passage de paramètre d'aucune sorte !
- Notifications système trop faibles sur Firefox

- Retour en arrière de 15 ans par rapport à `window.openDialog()`

Performances, système

- La « main popup » est un document indépendant
- Memory Footprint très supérieure aux Overlays de XUL

- Pas d’accès par exemple au lecteur d’empreintes
- Qui a dit « touchbar » ? 😎

Localisation

- Un seul fichier L10N `messages.json` par locale
- (Très) pénible à gérer pour les gros projets

Styles

- Les CSS injectées ont le même poids dans la cascade que celles de la page
  - Aucun « scoping »
  - Collisions majeures avec les styles de la page
  - Extrêmement compliqué d'injecter de la UI correcte partout
  - Rien de nouveau sous le soleil mais avec les WebExtensions, on est obligé d'injecter…
- Très gros soucis de z-index

Communication

- Des messages JSON dans tous les sens ☹
- Une frame injectée dans une page ne peut communiquer avec cette page qu'au travers du background !
- La Background Page n'est pas accessible aux scripts de page

- Attention au nombre de messages envoyés !
  - Message « Waiting for FooBar extension »

Startup, login, logout, etc.

- Injection des scripts et CSS dans tous les onglets
  - Pratique mais…
  - … potentiellement TRÈS lent et coûteux en performances

- Ou

- Injection uniquement dans l'onglet « actif »
  - Bien plus rapide mais…
  - Rien dans le manifest pour ça, c'est à vous de le gérer manuellement

- L'API `idle` est pour le système, pas pour le navigateur…

nativeMessaging

- Possibilité de communiquer avec une appli indépendante

`port.postMessage()`	→	`stdin`
`port.onMessage()`	←	`stdout`

- Lève les limitations d'API, en particulier l'accès aux système de fichiers (on est vraiment en 2018 ?)
- Intégration bien meilleure au Window Manager

IPR

- La seule protection (faiblarde) des assets est *l'uglification*
- Or les uglifiers travaillent rarement sur plusieurs fichiers…
- … et génèrent toujours un seul fichier uglifié

- Or limite de taille de fichier imposée par les Stores !

Paramètres

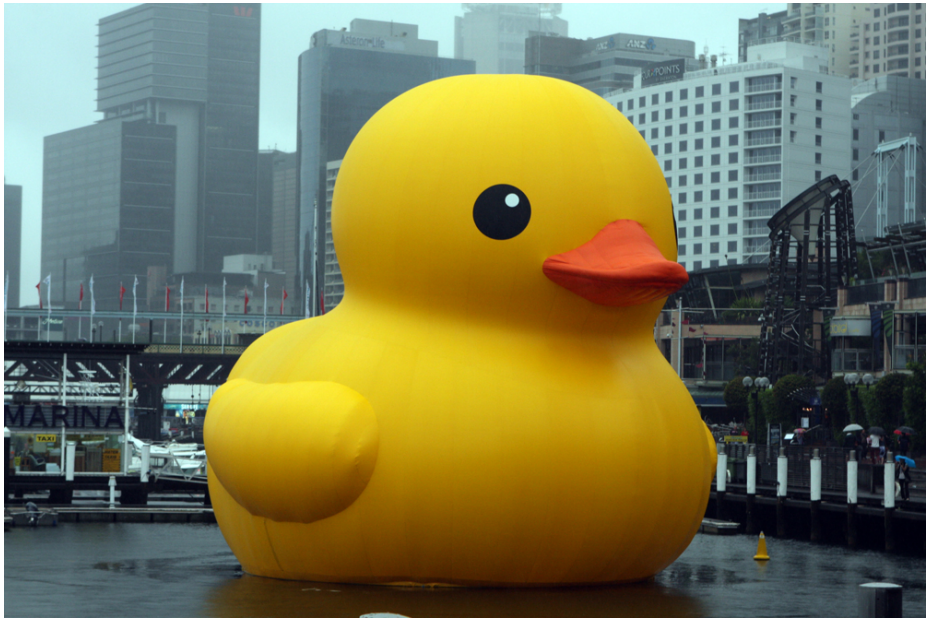
- Le manque d'interop ne se limite pas à « implémenté ou pas »
- Il descend au niveau paramètres…
- Certains paramètres cruciaux (genre `frameId`) peuvent manquer
- Le manque de standardisation est visible, sensible et cruel pour nous les développeurs

Le feeling général

- Bordelus Maximus…
- Après le PGCD et le PPCM, nouvelle notion du PPCD (sic)
- C'est pas beau, pas complet, pas interopérable
- La progression est lente et surtout opaque
- Aucun workaround possible, évidemment
- Apple, Apple, Apple…
- **Mais pourquoi ceci n'est-il pas un vrai Standard W3C, édité par un Working Group, publié comme REC ?**

- Et comme d'habitude, presqu'aucune compétence disponible sur le marché de l'emploi…

Merci



Eva Rinaldi - Rubber Duck - CC BY-SA 2.0

Des questions ?

http://beta.privowny.com