Extensions Manifest v3

http://go/extensions-manifest-v3
WIP list of potential breaking changes we might want to include in manifest v3.

Deprecated extension/app features

- The "experimental" permission
- chrome://favicon (bug another bug apparently which I can't access)
- "persistent" key of background pages no longer optional (bug).
- remove chrome.Event (bug)
- chrome.app.window deprecation / behavioural changes:
    - …
- chrome.pushMessaging (replaced by chrome.gcm)
- chrome.socket (replaced by chrome.sockets)
- move captureVisibleTab into chrome.tabCapture, require active tab for it
- file_handlers: “title” attribute

Deprecated web features

- Sync XHR (bug)
-

Tightened security

- Strictest CSP, extensions can't override (e.g. no more re-enabling eval) like apps.
- For extensions to declare all subresources via web_accessible_resources, not just top-level resources. This would break an accessible “page.html” from including “image.img” without listing it in web_accessible_resources. Important for reasoning about security properties.
- Only allow remote script injection from URLs that match a 'script-src' that it declares in its manifest (meaning, among other things, https only)
-
-

Untriaged

- Incognito split mode should be the default for event pages.
    - Haven’t fully thought this through... but having 2 incognito modes is a pain, and split is more flexible than spanning.

    - Yet spanning is more likely what developers actually want, so maybe we should only support that.
- Fix the permissions mess: rather than a permissions array, just use toplevel manifest keys. e.g.
    - “permissions”: [“storage”] becomes “storage”: true
    - “permissions”: [{“socket”: [“tcp-connect”]}] becomes “socket”: [“tcp-connect”]
    - “permissions”: [“*://*.google.com”] becomes “hosts”: [“*://*.google.com”]
    - bluetooth permissions become sane
    - https://code.google.com/p/chromium/issues/detail?id=62898 also has stuff on this
    - http://crbug.com/329303: Host permissions allow a path, but treat it as "/*". We should make specifying a path into an error.
    - something similar with file system permissions perhaps, which are weird compared to other permissions.
- TODO: Ask security folks for suggestions.
- Block localStorage/indexeddb/etc from accessing anything but the extension’s security origin. In practice this basically involves blocking it from content scripts. Why? to minimise the chance of an extension interfering with the page in unexpected ways (and I can’t think of any legitimate reason for an extension to do it). For content script storage, extensions should use the storage API.
- Execute extension JS with ‘use strict’ mode.
    - probably need an option to turn this off for libraries, in the same way that we have for eval currently
- Enforced stricter CSP that prevents remote JS from being executed
    - Maybe define an extension-library mechanism which must be uploaded to the store, and make extensions declaratively depend on it.
- Think up a more generic name than “chrome extensions” that encompasses apps too, and start using that? E.g. don’t run at chrome-extensions:// URLs, run at... chrome-platform or something.
- Promise-style APIs? https://github.com/slightlyoff/Promises/
- unmix URLs permissions and features permissions (https://docs.google.com/a/google.com/document/d/1y9FU7pwLroxA8QF-bBNKenHZ5s0MhGGU70wg-vBN4vE/edit)
- fine-grained host permissions; separate content script permissions from background permissions; while it might not mean anything in practice (extensions can still execute scripts on pages to do privileged XHR; anything they can do on-page is likely doable via XHR from a background page) it may still be valuable for static analysis.
    - also consider a cookie-less version of host permissions, though I doubt this is much value.
- fix "port madness in URL patterns" (we either do, or don't, support ports; this came from the link below. needs further investigation)
- disable 'code' from tabs.executeScript
- Get rid of chrome.runtime.lastError, make error part of the callback - this would imply fixing bugs like this.

- APIs with proper web-like Events?
- Remove app and extension namespace. The distinction between apps and extensions is getting blurry, it would good to not have it baked into the APIs as we have more flexibility then.
- Split chrome.app.runtime.onLaunched into separate events for open, launch with file, launch with URL, launch with media gallery.
- Make specifying plugin requirement without a plugin a load error (crbug.com/359238)
- Require extensions using NaCl to declare this in a permission. (bradnelson)
    - Possibly require all architectures in nmfs loaded from such apps?
    - CONs: Requires developer action without adding much benefit for them
    - PROs: Makes malware analysis (assuming nacl is specifically interesting) to find nacl apps more easily. Provides forcing mechanism for developers to support arm (good for chromebook users).
- Similarly, require declaration for Flash usage

Long ago we had written up a list of breaking changes we wanted to do at:
http://dev.chromium.org/developers/design-documents/extensions/proposed-changes/extension-system-changes/breaking-changes
Some/many of those may have been done already for manifest v2, but it's worth taking a look again to see if there are remaining items

Infeasible, not high enough value, further discussion needed, etc

- Any script executed in a page (<script> tags, javascript: URLs, etc) execute in the content script context, not the page's. I.e. extensions should have no way of executing code in the page's context.
    - or just disallow in-page script execution entirely (script tags, javascript: URLs)
    - this may be too harsh and kill too many extensions, sadly, but see comments about potentially providing an alternative. Either way it's likely too much work to block manifest v3.
- Use HTML-y APIs, like chrome.browserAction.addEventListener('clicked', …) rather than chrome.browserAction.onClicked.addListener(...).
- Make background script to be always background.js (saves 7 lines in manifest.json)
- localStorage (and sessionStorage?)