Staging of WebAssembly Features

(externally visible, non-confidential)

Author: ahaas@, mstarzinger@
Status: Draft | **Final**
Tracking Bug: https://crbug.com/v8/9601
Created: 2019-08-07 / **Updated:** 2019-08-23

Name	**Write (not) LGTM in this row**
hablich	LGTM
machenbach	LGTM
mstarzinger	LGTM
adamk	LGTM
Your ldap?	

History

2019-08-07	Initial document
2019-08-08	• Add background information about --harmony. • Mention --es-staging.
2019-08-09	• Add adamk's reason for not sharing flags with JavaScript in the Alternatives section.
2019-08-23	Change status to Final

Overview and Goals

For WebAssembly proposals implemented in V8 we currently don't have an established path for when/how to enable them. Here is what is currently missing:

- Ensuring fuzzing coverage before a feature is enabled by default: Currently all feature flags are prefixed with "--experimental-wasm-foo", explicitly excluding them from fuzzing. Many of our fuzzers are JS-based, but we would still benefit from fuzzing the JS/Wasm bindings layer that way. Additionally our dedicated WebAssembly fuzzers should also be enabled. This should also cover potential interaction between multiple features enabled at the same time that our tests are missing.
- Some way for early adopters to experiment with implementations of upcoming features by explicitly enabling them via chrome://flags/ or similar.
- Goal post for feature owners/developers of when to consider a feature done/stable in V8. This has the risk of being an artificial goal post, because the proposal can always evolve and change afterwards. But at the minimum the feature implementation would be in a self-consistent state, despite being out of sync with the proposal at that point.

To this end we propose to classify the features in V8 into two groups, "experimental" and "staged". The "experimental" group reflects the state most of our features are in now. The "staged" group would fill the gap mentioned above. When moving a feature to the "staged" group all of the above should happen automatically.

Ideally the decision of when to move a certain feature from "experimental" to "staged" could be done mostly based on implementation status in V8 (e.g. stability, completeness, need for additional coverage/feedback) and be somewhat decoupled from the proposal stages (modulo being cognizant of expected substantial changes to the proposal).

The goal of this is to make enabling new feature by default less risky (by having an intermediate step) and also to provide an additional goal post for feature owners to reach when enabling a feature by default is blocked by progress on the proposal itself.

Proposal

We have to do the following changes:

- Split the current list of WebAssembly features (wasm-feature-flags.h) into three lists (CL):
  - Experimental
  - Staged
  - Shipped
- Introduce a new --wasm-staging flag in flag-definitions.h, and add implications from that flag to all flags listed in the “staging” list in wasm-feature-flags.h.
- Add a new flag in chrome://flags/ which allows to enable the --wasm-staging flag.
- On our try-bots and fuzzers, enable --wasm-staging where --es-staging ~~--harmony~~ is also enabled.
- Enable --wasm-staging in the libfuzzer fuzzers. Set the flag in the generated tests.

- Add the proposal-spec tests to our tests.

At the moment we start new features “foo” with a --experimental-wasm-foo flag. The “experimental” prefix keeps ClusterFuzz from using the flag. I don’t really see us changing the flag name once we move a feature from experimental to staging. However, this means that even a staged feature has the --experimental-wasm-foo flag. I’m open to suggestions here.

Alternatives

- Do not introduce a new --wasm-staging flag but re-use --harmony or --future for this.
    - Sharing a flag only makes sense from an inside-V8 point of view. For users it may look surprising. WebAssembly is sufficiently different to JavaScript to justify an additional flag. Users who want to experiment with new JavaScript features may not be the same as those that experiment with WebAssembly features. JavaScript also does not share feature flags with other WebPlatform features.
    - Background information about JavaScript flags:
        - --future is for staging VM backend features, not language features
        - --harmony: Experimental/Staged JavaScript language features
            - Background Information:
                - jkummerow@: ECMAScript Harmony was the early codename for what later became ES6. --harmony was the umbrella flag for ES6 features. After ES6 we kept the flag.
                - The individual features had flags named --harmony-feature-name, see doc. --harmony-feature-name flags are ignored by ClusterFuzz, the same as --experimental-flag-name.
            - --harmony does not sound like an intuitive name for a staging flag for WebAssembly features. Also ClusterFuzz does not use --harmony but --es-staging. There is an implication from --es-staging to --harmony.
            - I don’t know where “--harmony” is coming from, or if we could re-use the flag for WebAssembly.
        - --es-staging: The flag for enabling ECMAScript features in ClusterFuzz.
            - The name does not sound appropriate for WebAssembly.
            - We could introduce a --staging flag with implications to both --es-staging and --wasm-staging. --staging can then be used in ClusterFuzz. Alternatively we can also just add --wasm-staging everywhere in ClusterFuzz where --es-staging exists already.
- Streamline the naming
    - I’m open to better names.