Object.observe

Implementation report

Take-Aways

- Results are encouraging
- Two Google projects are planning to deploy run-time support
- Use cases argue for WeakRef
- Worth considering special treatment of Array mutations

V8 Implementation Status

- Spec fully implemented (behind a run-time flag)
- Mostly self-hosted: changeRecord allocation, enqueuing, and delivery happens in JS
- Mutation sites of observed objects de-optimize; observed arrays are always "slow mode"
- Biggest perf bottleneck is freezing changeRecords; plan to speed up Object.freeze()

Spec changes (since acceptance in Oct '12 to ES7)

- Object.deliverChangeRecords() continues delivery until pending records are cleared
- Added { type:'prototype' .... } changeRecord which reports changes to __proto__
- Minor changes to changeRecord generation to enforce consistency invariants

ChangeSummary JS Library

- Supports dirty-checking and Object.observe (polyfill to fast/safe object observation)
- Exposes semantics for a "diffing" summary (T_n -> T_{n+1} changes)
- Observe
  - "Values at a path", e.g. observe.observePath(foo, 'bar.baz');
  - Entire objects
  - Array "splice" mutations e.g. { index: i, removed: [...], addedCount }
- Prototype support for abstract "bindings", read-synchronous "computed properties" using dependency tracking.
- Lack of WeakRefs forces dispose() pattern

Perf Analysis (dirty-checking vs. Object.observe)

- Results not surprising (this is good!)
- *nothing-changed* case is overwhelming win.
  - Discovering that nothing changed only incurs the cost of de-optimizing observed objects.
- Object.observe case never needs to keep a copy of observed data.
- Object.observe appears to become slower than dirty-checking when between 10~50% of observed properties have changed (depending on observation "type")
- Arrays encourage "hidden" everything-changed cases e.g. unshift()

Adding support for Array "splice mutations"

- Report on changes to Array elements (isUInt32(propertyName)) as "splice mutations"
- Degrade to "normal" property mutations if...
  - !Array.isArray(array)
  - array has non-data index properties
  - array has non-writable or non-configurable index properties
  - array has index properties visible on proto chain
  - operation will affect indices > UInt32