Enumeration is loosely specified

“The mechanics and order of enumerating the properties (step 6.a in the first algorithm, step 7.a in the second) is not specified. Properties of the object being enumerated may be deleted during enumeration. If a property that has not yet been visited during enumeration is deleted, then it will not be visited. If new properties are added to the object being enumerated during enumeration, the newly added properties are not guaranteed to be visited in the active enumeration.

“Enumerating the properties of an object includes enumerating properties of its prototype, and the prototype of the prototype, and so on, recursively; but a property of a prototype is not enumerated if it is ‘shadowed’ because some previous object in the prototype chain has a property with the same name.”
Firefox 4 betas

- Andreas’s “fastiterators” patch
- Breaking change: eagerly compute property set up front
- Compromise: suppress *some* deleted properties
- Too ad-hoc to standardize
Why standardize?

- Portability: enumeration order inconsistent across browsers
- MOP: unspecified behavior leaks into enumerate trap
- Performance: worthwhile breaking changes?
“Concurrent” modification

Some alternatives:

● Codify quasi-intersection of existing engines’ behavior

● Ignore all modifications (eager snapshot)

● After modification, throw at next iteration (dirty bit)
“Concurrent” modification, ctd.

Consequences of disallowing/ignoring:

- Eager snapshot expensive for large property sets?
- Precludes iterating on manual work-lists
- Portability hazard anyway; better expressed via iterators
Lexicographic ordering (most significant to least):

1. Prototype chain order
2. Index order
3. Creation order
Summary

- Harmony would benefit from tighter enumeration semantics
- Simplified behavior of concurrent modifications
- Standardized enumeration order