Skipping iterator protocol allocations

Disclaimer: I have not touched V8 since before the introduction of Torque, so I have no idea if this is actually feasible. I haven't touched the other engines at all.

Currently all engines create `{ done, value }` pairs when consuming any iterator, even built-in iterators, except in a few cases where they have specific optimizations such as for the array iterator. This is wasteful, and that waste will be more acutely felt as iterator helpers are adopted. Here is a plan to avoid that:

- For all built-in iterators, factor out an underlying implementation which returns `value | THE_HOLE` (or some other sentinel value). Have a wrapper which calls the underlying implementation, checks for THE_HOLE, and creates the appropriate `{ done, value }` pair. This should have no effect on anything.
- On such wrapped iterators, add a private symbol or something which exposes the underlying implementation to engine code (but not userland).
    - It should be possible to do this for userland generators too. Those are just another kind of built-in iterator.
- In places which are doing IteratorStep/IteratorValue, first check for that symbol. If present, call the underlying implementation and check for THE_HOLE directly, instead of calling IteratorStep/IteratorValue.
    - The number of places this would touch right now is large, so first refactor the current callers of IteratorStep/IteratorValue to call IteratorStepValue, as introduced in this PR. After that refactoring this step only needs to touch IteratorStepValue. (I am assuming an internal implementation of IteratorStepValue can return THE_HOLE or a Maybe or something.)
    - A followup optimization in callers which consume the iterator in a loop could be to check for the private symbol only once, outside the loop, and then go down a different path. This would avoid per-iteration lookups of the private symbol and per-iteration branches at the cost of having to duplicate the looping logic. No idea if this is worth it. Probably it is for some callees, like spread syntax.
    - An alternative optimization would to be move the check for the underlying implementation into the place which creates internal Iterator Records, and add an extra "fast iterator function" field to said records, and then have IteratorStepValue check for and call this value if present. This avoids the per-iteration private symbol lookups from the naive approach, although it does keep the per-iteration branch which would be eliminated by the optimization strategy in the previous bullet (but would not require touching sites which are doing iteration the way that one would).
- Maybe expose the private symbol to embedders, so HTML/Node/etc can write their own iterators in the same way.