`Array.isSparse(array)`

Detecting if an array is dense

Addressing a array performance cliff and correctness edge case

Champion: `Ruben Bridgewater and Jordan Harband`
Author: `Ruben Bridgewater`
July 2025

What is a Sparse Array?

An array a is sparse if there exists an integer index i such that:

- 0 <= i < a.length
- a does not have an own property at index i

That is, it has at least one "hole" in the array indices.

Motivation

Sparse ("holey") arrays cause correctness pitfalls and potentially major slow-downs across engines.

Today, developers reinvent fragile checks or pay runtime cost scanning arrays, in case they case about correctness.

We propose Array.isSparse(x) – a non-prototype static method mirroring Array.isArray.

Easy to spec, trivial to implement, and ready to unlock optimizations in user-land code.

Specification

Abstract Operation: Array.isSparse (*value*)

1. If `IsArray(_value_)` is *false*, return *false*.
2. Let *len* be `ToLength(Get(_value_, *"length"*))`.
3. For each integer *i* from `0` to `_len_ - 1`:
    i. If `HasOwnProperty(_value_, ToString(_i_))` is *false*, return *true*.
4. Return *false*.

This specification should easily be optimizable by engines in case no proxy is used.
V8, SpiderMonkey, and JavaScriptCore all have internal representations for sparse arrays (while the current types might not indicate all possible cases right now).

Examples

```
Array.isSparse([])                      // false
Array.isSparse([1, 2, 3])               // false
Array.isSparse(new Array(0))            // false
Array.isSparse({ 0: 'a', length: 1 }) // false (not an array)
Array.isSparse('abc')                   // false (not an array)

Array.isSparse([1, , 3])                // true
Array.isSparse(Array(5))                // true
Array.isSparse(new Array(1))            // true
```

Correctness

```
const a = [1, , 3];           // hole @ index 1
const b = [1, undefined, 3]; // filled with undefined

// Bugs creep in:
a.map(x => x * 2); // → [2, , 6] — hole preserved
b.map(x => x * 2); // → [2, NaN, 6]
```

Holes vs undefined semantics break data-processing pipes & equality checks.

Lack of a standard predicate makes safe branching impossible.

Performance

Accessing a hole slows down the code by not only accessing the prototype chain in that specific case, the code will generally behave slower towards all arrays afterwards.
That should be prevented and there is no way to do that in a performant way right now (checking for own properties will be slow).

See https://v8.dev/blog/elements-kinds

Use cases

- `Data validation`: Ensuring an input array has no holes before processing.
- `Serialization`: Sparse arrays can behave unexpectedly when serialized to JSON.
- `Performance optimization`: Engines may deoptimize when arrays become sparse; libraries may wish to warn or convert. Algorithms can skip checks currently necessary to guard against sparse arrays when working with dense arrays.
- `Correctness`: Developers sometimes unintentionally create sparse arrays, not knowing about the implications. This allows to easily test for that.
- `Security`: If someone were to add an index onto Array.prototype, it would "leak" through the hole.

Usage Examples

- Node.js - all logging, all deep comparison methods
- lodash - dense-array fallback for every iterator
- jest - snapshot & diff logic skips holes
- fast-deep-equal - special-cases sparse vs. dense in deep-compare
- deep-equal - would allow for optimizations
- object-inspect - describing sparse arrays for debugging

Many use this in pattern in polyfilled code. It is possible that this allows *future* polyfills to also be more performant.

Alternatives

It is also possible to do the opposite: `Array.isDense(array)`.

Both is fine. Due to prior art, isSparse was used.

Prior Art

- SciPy scipy.sparse.issparse (Python).
  https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.issparse.html
- Android SparseArray class (Java / Kotlin).
  https://developer.android.com/reference/android/util/SparseArray
- V8 / SpiderMonkey internal helpers / types (IsFastPackedElementsKind, etc.)
  https://source.chromium.org/chromium/v8/v8.git/+/ec37390b2ba2b4051f46f153a8cc179ed4656f5d:src/elements-kind.h;l=14

Summary

Goal: implement this for correctness, performance, and security reasons.

Next steps

- Getting input
- Addressing comments
- Stage 1