$JAVASCRIPT_ENGINE internals

for JavaScript developers

@mathias

JAVASCRIPT_ENGINE='V8'

@mathias

V8 internals

for JavaScript developers

@mathias // @v8js

Elements kinds

in V8

```
const array = [1, 2, 3];
```

@mathias

```
const array = [1, 2, 3];

// elements kind: PACKED_SMI_ELEMENTS
```

```
const array = [1, 2, 3];
// elements kind: PACKED_SMI_ELEMENTS
array.push(4.56);
```

@mathias

```
const array = [1, 2, 3];
// elements kind: PACKED_SMI_ELEMENTS
array.push(4.56);
// elements kind: PACKED_DOUBLE_ELEMENTS
```

@mathias

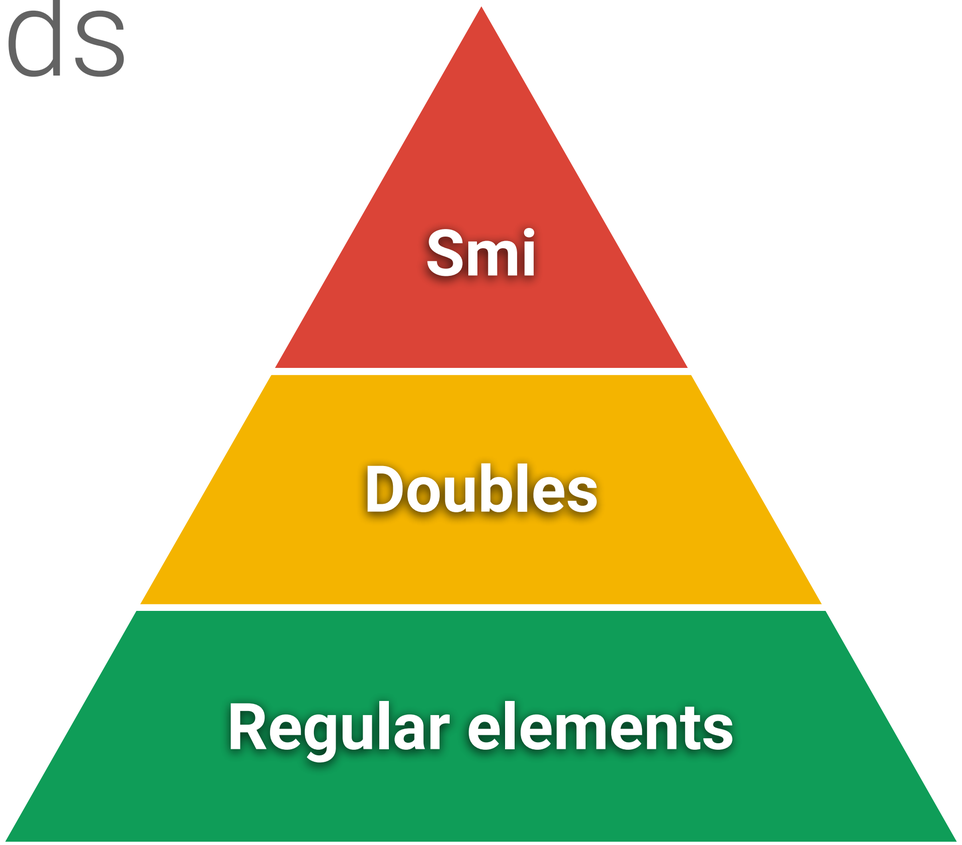
```
const array = [1, 2, 3];
// elements kind: PACKED_SMI_ELEMENTS
array.push(4.56);
// elements kind: PACKED_DOUBLE_ELEMENTS
array.push('x');
```

@mathias

```
const array = [1, 2, 3];
// elements kind: PACKED_SMI_ELEMENTS
array.push(4.56);
// elements kind: PACKED_DOUBLE_ELEMENTS
array.push('x');
// elements kind: PACKED_ELEMENTS
```

@mathias

Elements kinds



```
const array = [1, 2, 3];
// elements kind: PACKED_SMI_ELEMENTS
array.push(4.56);
// elements kind: PACKED_DOUBLE_ELEMENTS
array.push('x');
// elements kind: PACKED_ELEMENTS
```

@mathias

```
array.length; // 5
```

index	0	1	2	3	4
value	1	2	3	4.56	'x'

@mathias

```
array.length; // 5

array[9] = 1;

// array[5] until array[8] are now holes
```

index	0	1	2	3	4	5	6	7	8	9
value	1	2	3	4.56	'x'					1

@mathias

```
array.length; // 5
array[9] = 1;
// array[5] until array[8] are now holes
// elements kind: HOLEY_ELEMENTS
```

index	0	1	2	3	4	5	6	7	8	9
value	1	2	3	4.56	'x'					1

@mathias

```
array[8];

// → ???
```

index	0	1	2	3	4	5	6	7	8	9
value	1	2	3	4.56	'x'					1

@mathias

```
array[8];
// → ??? ❌
```

index	0	1	2	3	4	5	6	7	8	9
value	1	2	3	4.56	'x'					1

@mathias

```
array[8];
// → ??? ❌
8 >= 0 && 8 < array.length; // bounds check
// → true
```

index	0	1	2	3	4	5	6	7	8	9
value	1	2	3	4.56	'x'					1

@mathias

```
array[8];
// → ??? ❌

8 >= 0 && 8 < array.length; // bounds check
// → true ❌
```

index	0	1	2	3	4	5	6	7	8	9
value	1	2	3	4.56	'x'					1

@mathias

```
array[8];
// → ??? ❌

8 >= 0 && 8 < array.length; // bounds check
// → true ❌

hasOwnProperty(array, '8');
// → false
```

index	0	1	2	3	4	5	6	7	8	9
value	1	2	3	4.56	'x'					1

@mathias

```
array[8];
// → ??? ❌

8 >= 0 && 8 < array.length; // bounds check
// → true ❌

hasOwnProperty(array, '8');
// → false ❌
```

index	0	1	2	3	4	5	6	7	8	9
value	1	2	3	4.56	'x'					1

@mathias

```
array[8];
// → ??? ❌

8 >= 0 && 8 < array.length; // bounds check
// → true ❌

hasOwnProperty(array, '8');
// → false ❌

hasOwnProperty(Array.prototype, '8');
// → false
```

@mathias

```
array[8];
// → ??? ❌

8 >= 0 && 8 < array.length; // bounds check
// → true ❌

hasOwnProperty(array, '8');
// → false ❌

hasOwnProperty(Array.prototype, '8');
// → false ❌
```

@mathias

```
array[8];
// → ??? ❌

8 >= 0 && 8 < array.length; // bounds check
// → true ❌

hasOwnProperty(array, '8');
// → false ❌

hasOwnProperty(Array.prototype, '8');
// → false ❌

hasOwnProperty(Object.prototype, '8');
// → false
```

@mathias

```
array[8];
// → ??? ❌

8 >= 0 && 8 < array.length; // bounds check
// → true ❌

hasOwnProperty(array, '8');
// → false ❌

hasOwnProperty(Array.prototype, '8');
// → false ❌

hasOwnProperty(Object.prototype, '8');
// → false ✅
```

@mathias

```
array[8];
// → undefined ✅
8 >= 0 && 8 < array.length; // bounds check
// → true
hasOwnProperty(array, '8');
// → false
hasOwnProperty(Array.prototype, '8');
// → false
hasOwnProperty(Object.prototype, '8');
// → false ✅
```

@mathias

```
packedArray[8];
// → undefined ✅
8 >= 0 && 8 < packedArray.length; // bounds check
// → true ✅
hasOwnProperty(packedArray, '8');
// → true ✅
hasOwnProperty(Array.prototype, '8');
// → false ✅
hasOwnProperty(Object.prototype, '8');
// → false ✅
```

@mathias

```
packedArray[8];
// → undefined ✅
8 >= 0 && 8 < packedArray.length; // bounds check
// → true ✅
has[illegible]nPropert[illegible]packedArr[illegible]'8');
[illegible]e
has[illegible](Array.pro[illegible];
//[illegible]
[illegible]op[illegible]ect[illegible]pe
//[illegible]false ✅
```

@mathias

```
array[0];

// → ???
```

@mathias

```
array[0];
// → ??? ❌
```

@mathias

```
array[0];
// → ??? ❌
0 >= 0 && 0 < array.length; // bounds check
// → true
```

@mathias

```
array[0];
// → ??? ❌

0 >= 0 && 0 < array.length; // bounds check
// → true ❌
```

@mathias

```
array[0];
// → ??? ❌

0 >= 0 && 0 < array.length; // bounds check
// → true ❌

hasOwnProperty(array, '0');
// → true
```

@mathias

```
array[0];
// → ??? ❌

0 >= 0 && 0 < array.length; // bounds check
// → true ❌

hasOwnProperty(array, '0');
// → true ✅
```

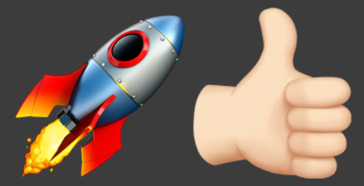
@mathias

```
array[0];
// → 1 ✅

0 >= 0 && 0 < array.length; // bounds check
// → true

hasOwnProperty(array, '0');
// → true ✅
```

@mathias

PACKED > HOLEY

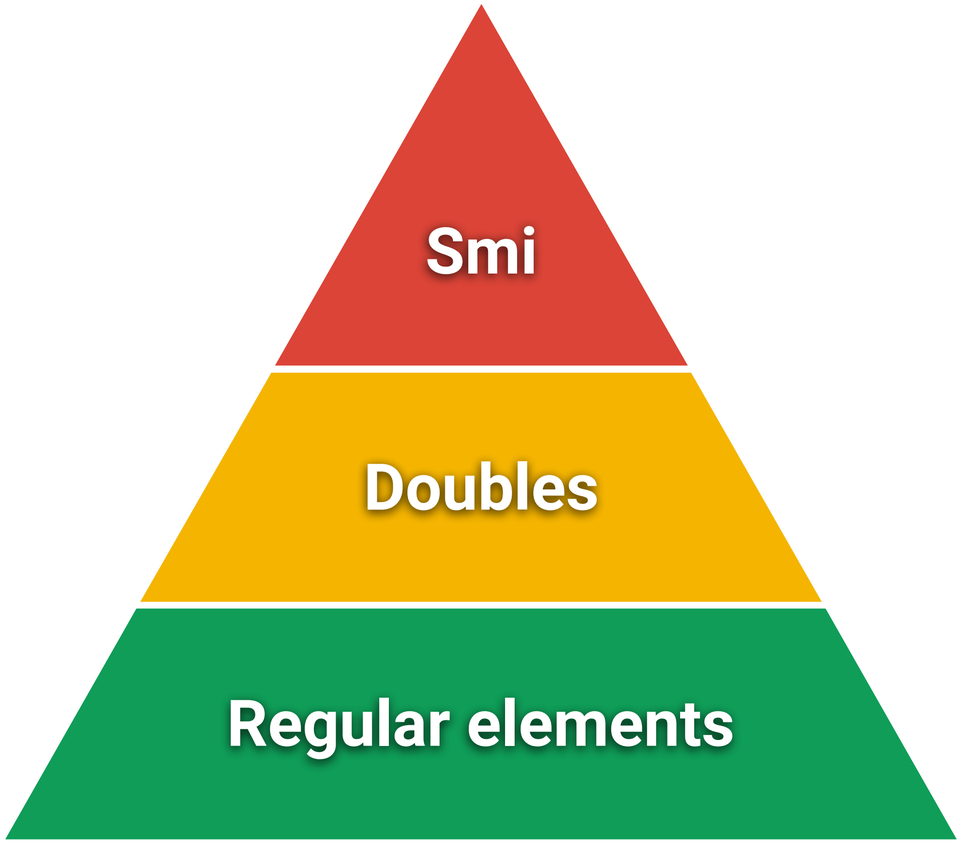
@mathias

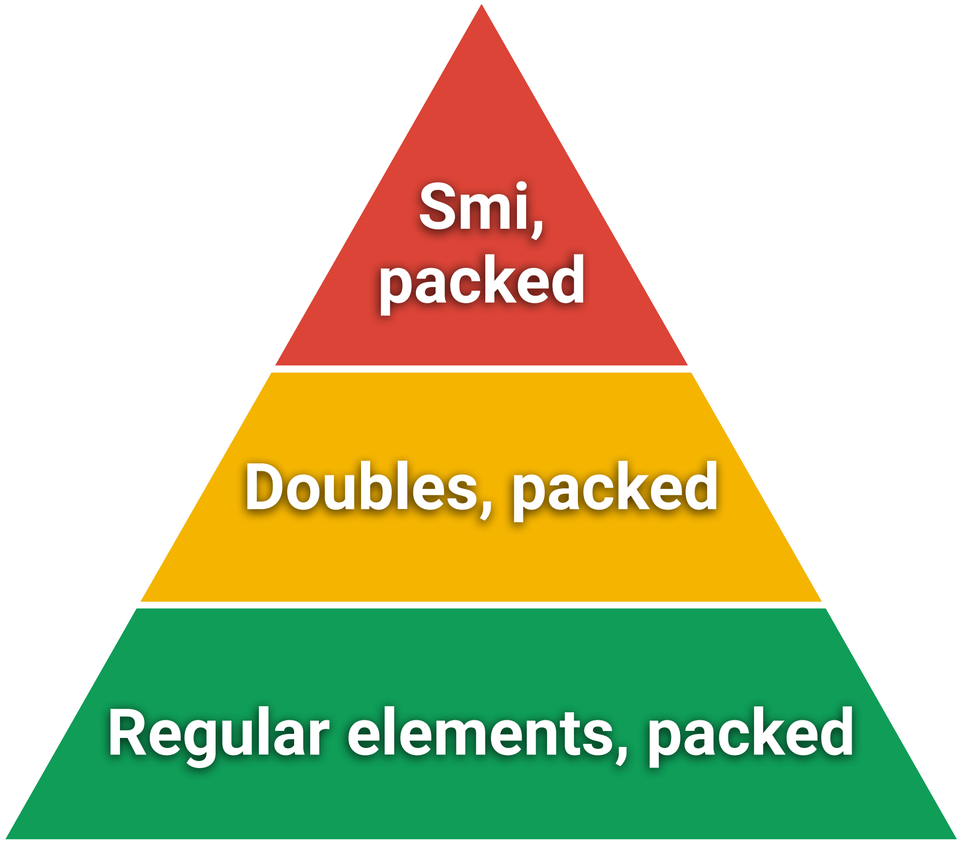


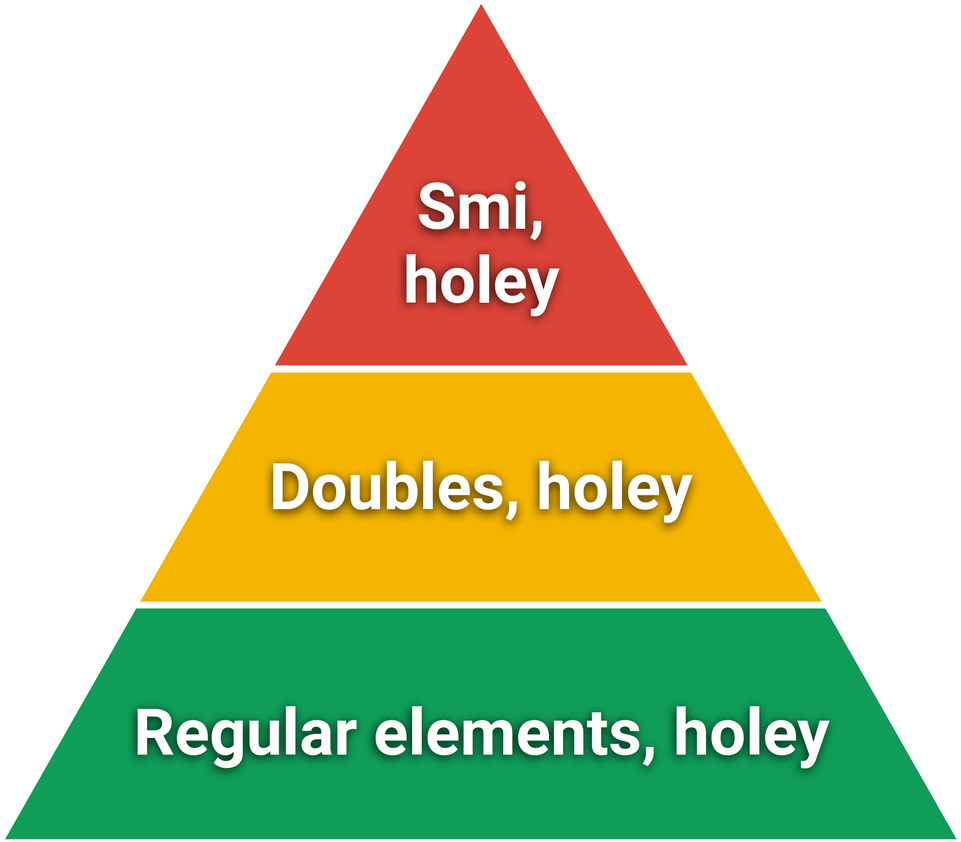
PACKED > HOLEY

@mathias

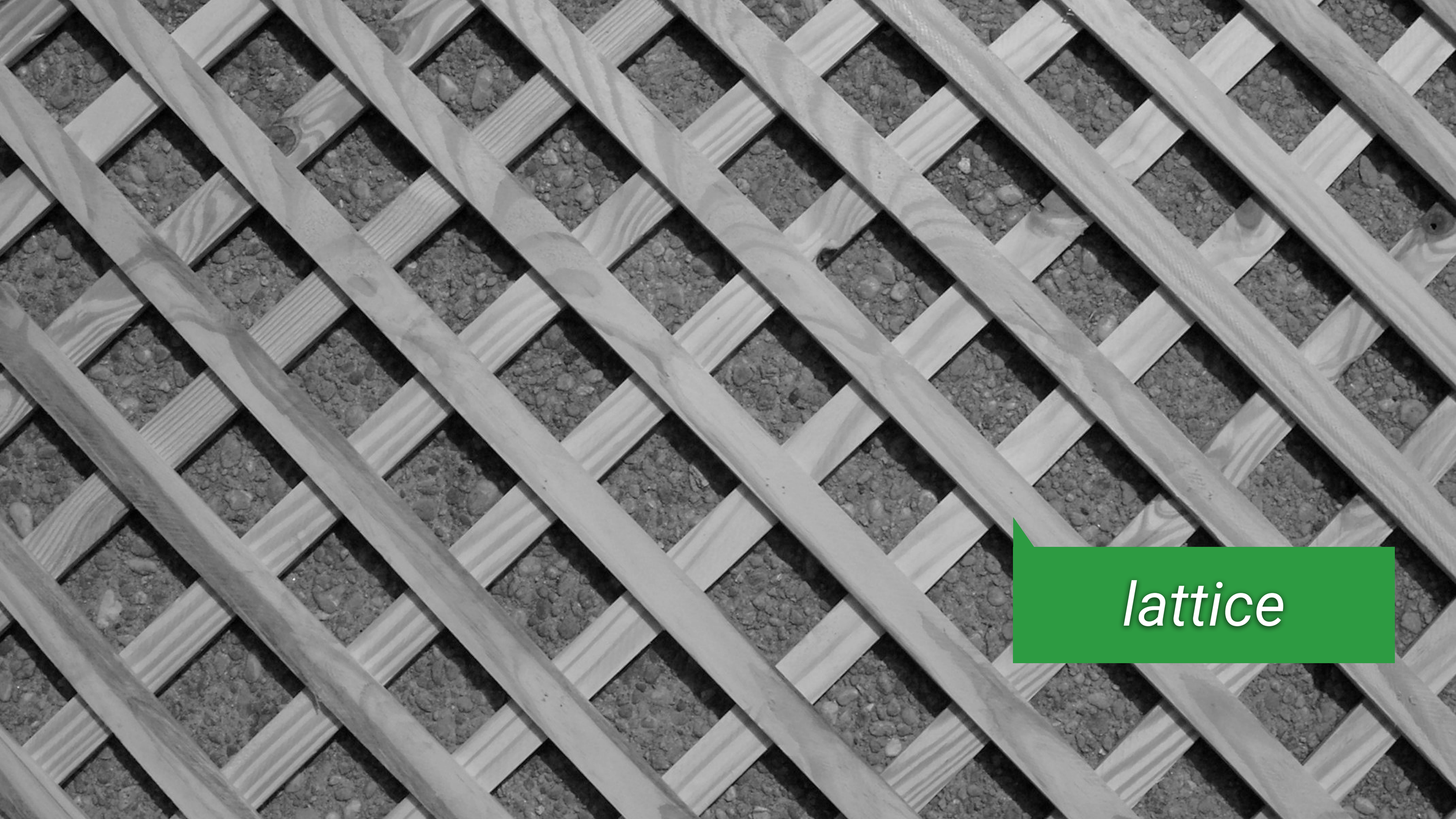
Elements kinds

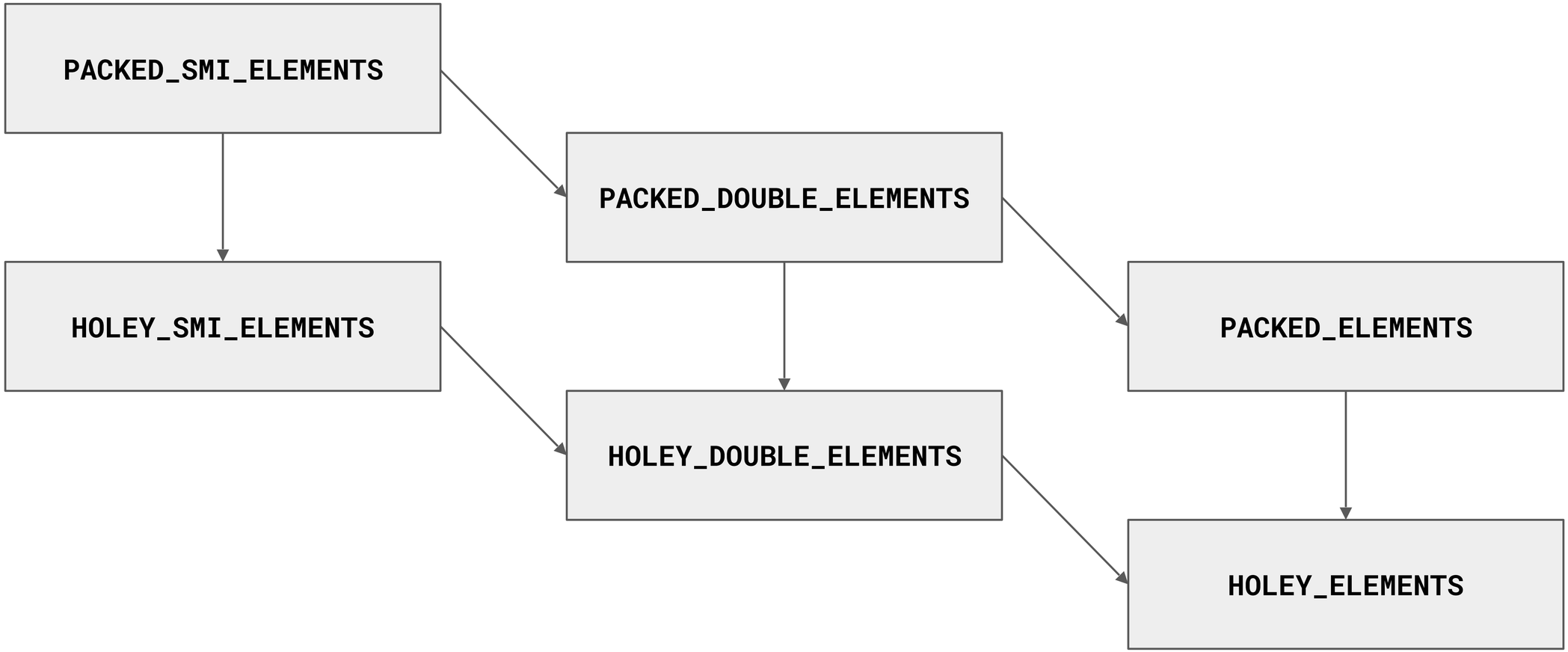



Smi,
packed
Doubles, packed
Regular elements, packed


Smi,
holey
Doubles, holey
Regular elements, holey

@mathias

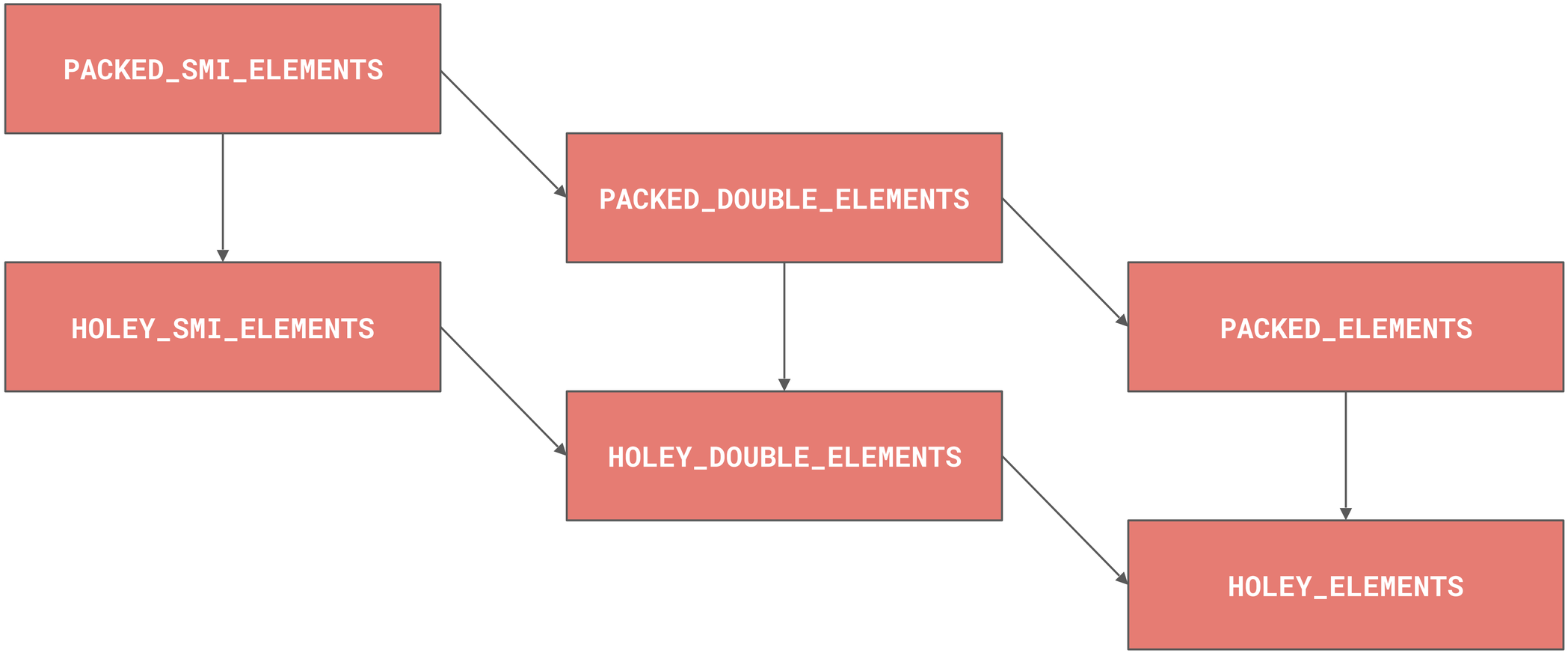

lattice


PACKED_SMI_ELEMENTS
PACKED_DOUBLE_ELEMENTS
HOLEY_SMI_ELEMENTS
PACKED_ELEMENTS
HOLEY_DOUBLE_ELEMENTS
HOLEY_ELEMENTS

@mathias

Chrome 59

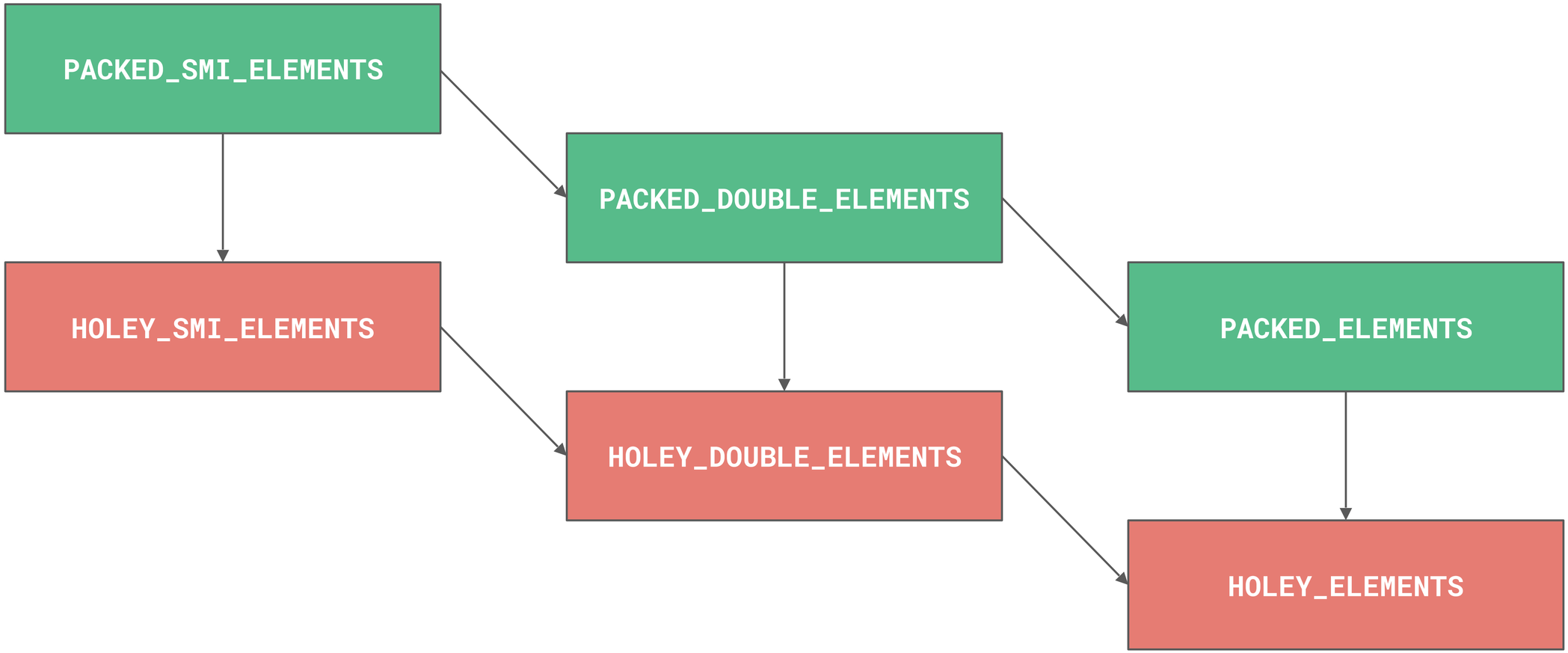
Array.prototype.forEach



@mathias

Chrome 61

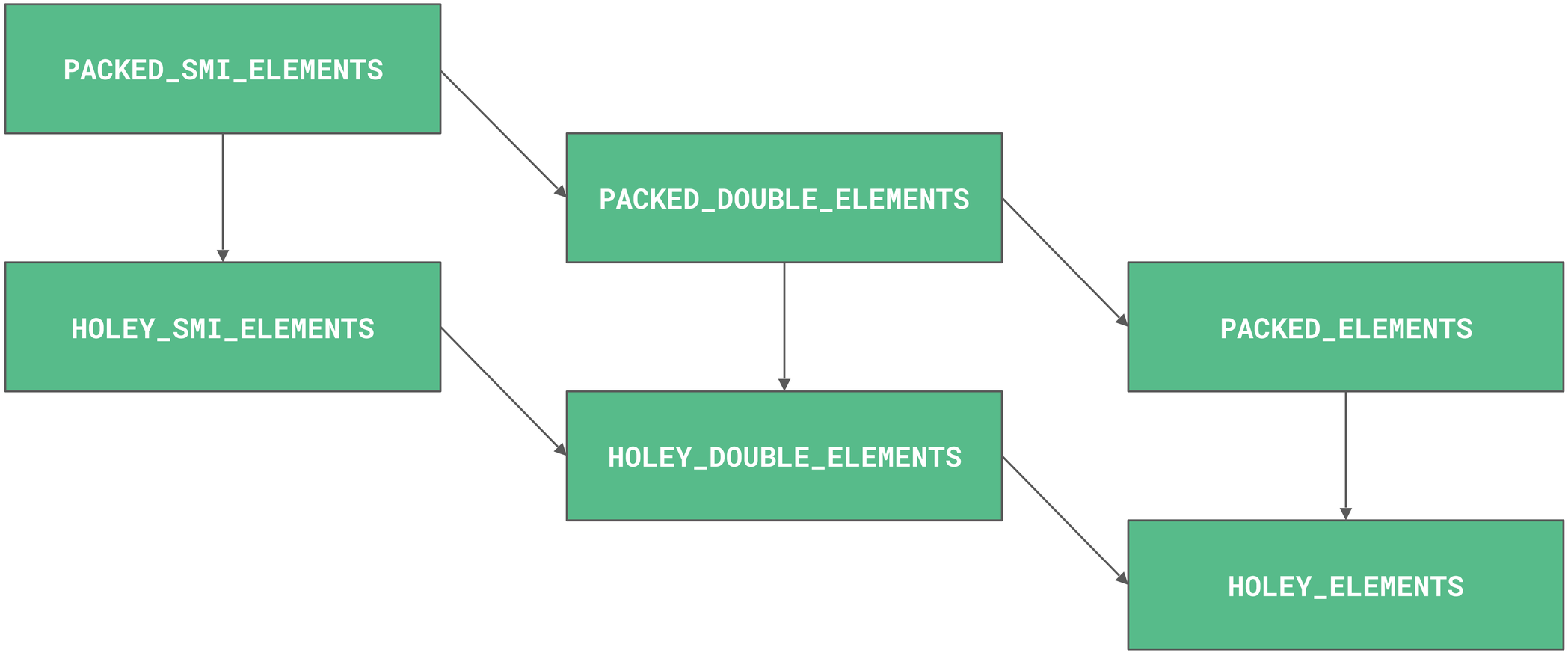
Array.prototype.forEach



@mathias

Chrome 64

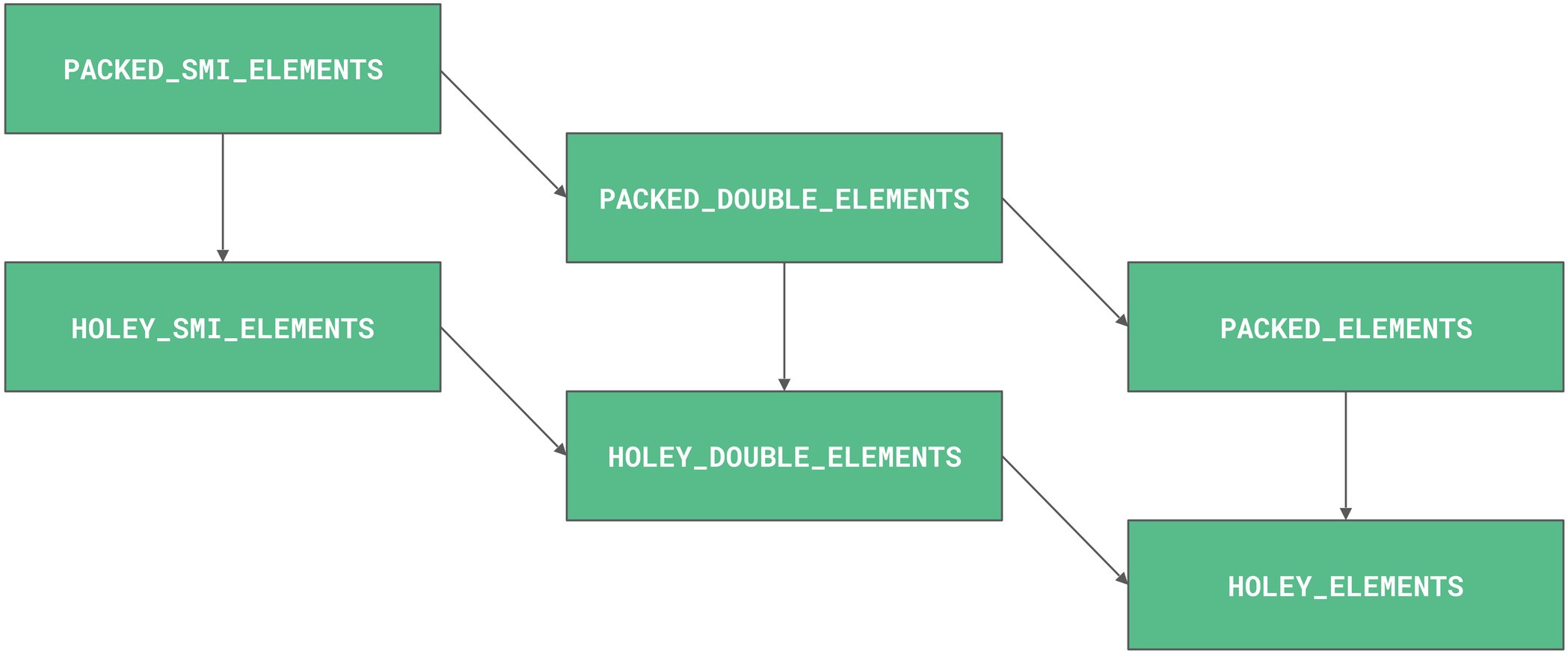
Array.prototype.forEach



@mathias

Chrome 64

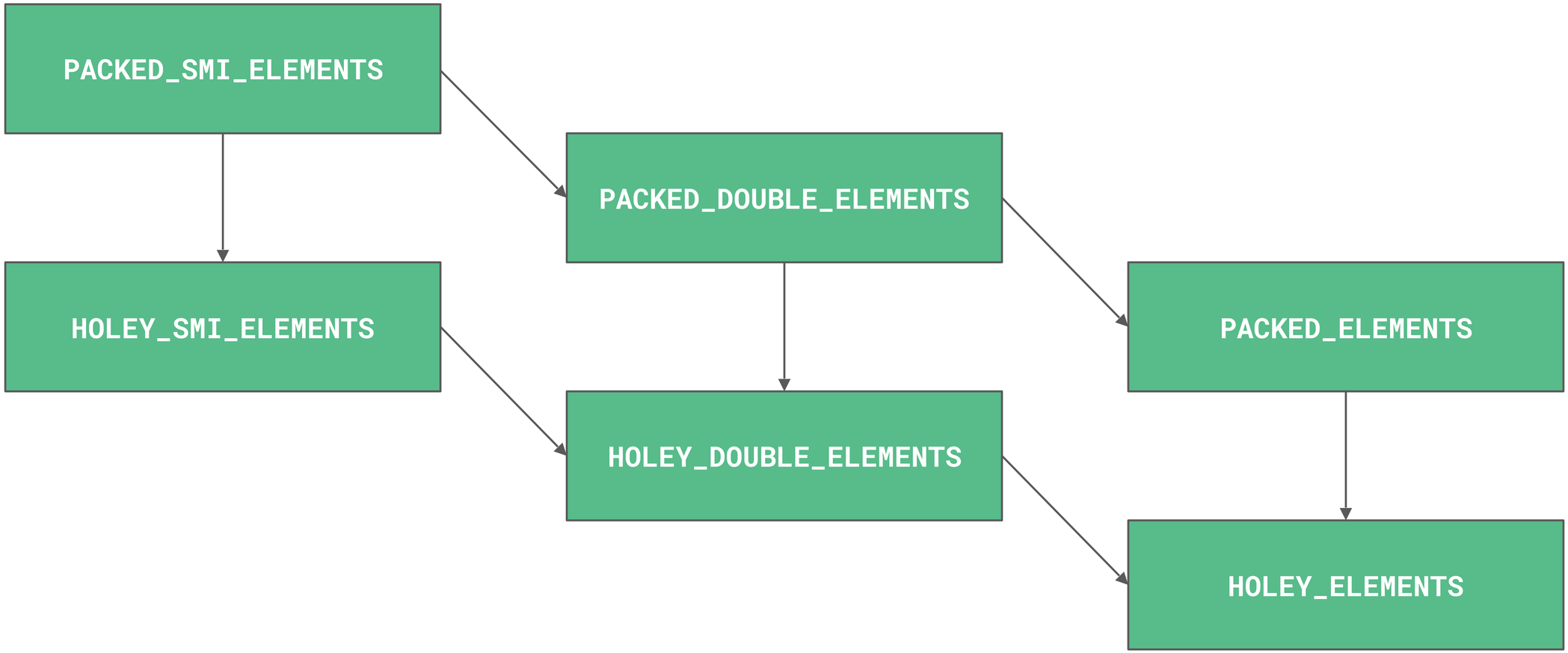
Array.prototype.map



@mathias

Chrome 64

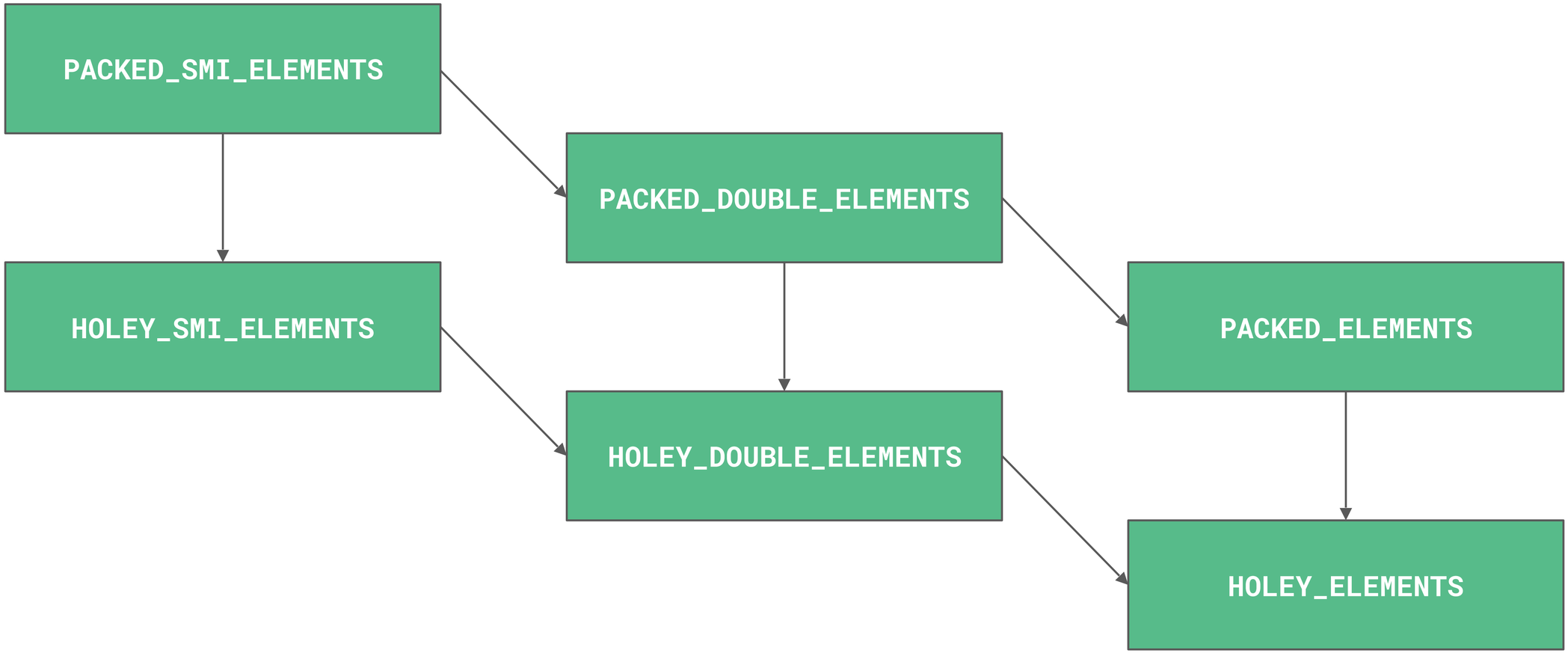
Array.prototype.filter



@mathias

Chrome 64

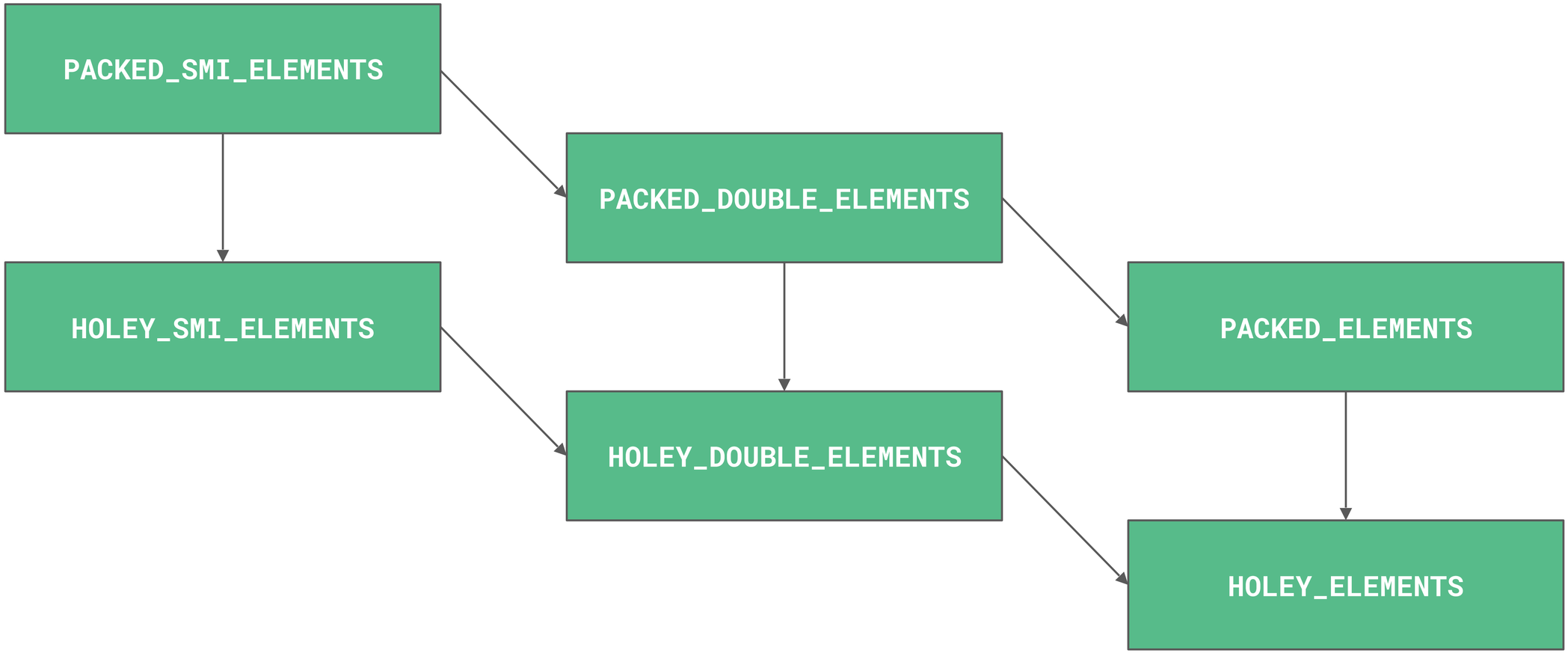
Array.prototype.some



@mathias

Chrome 64

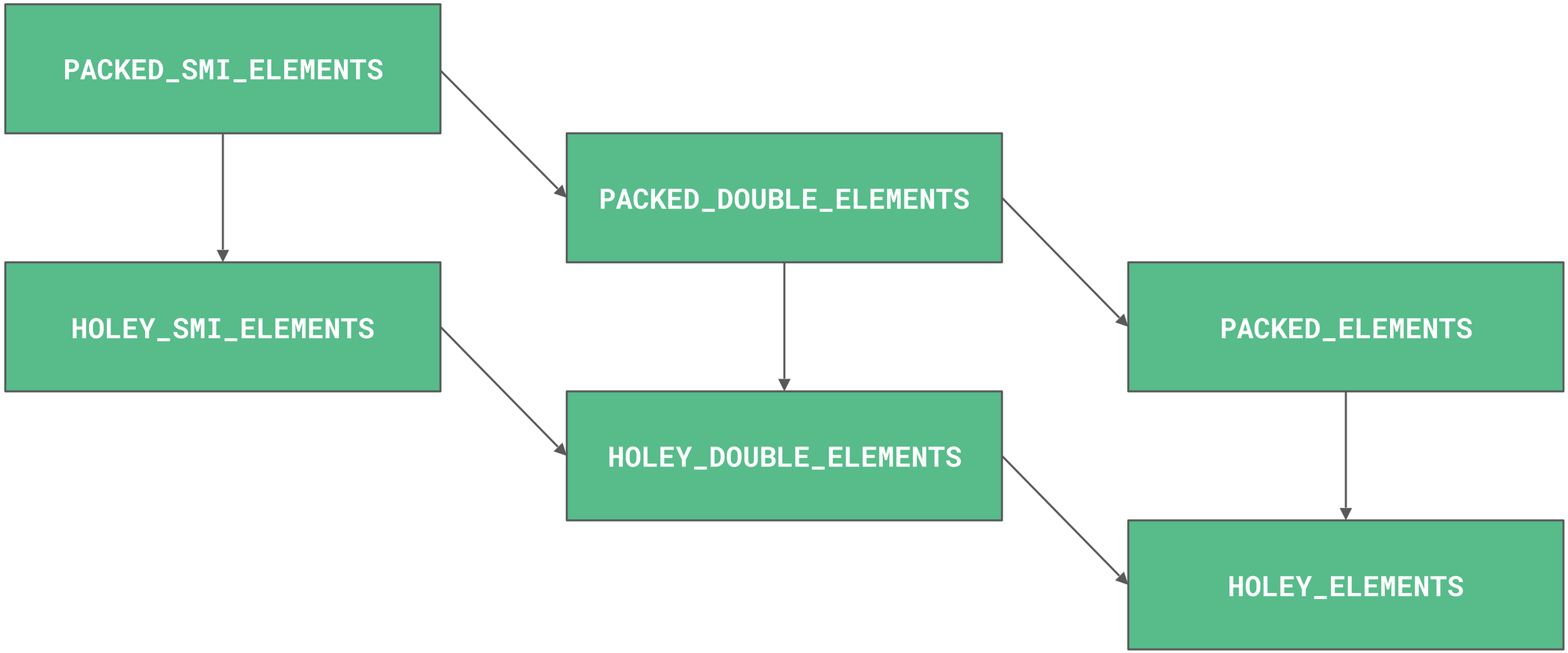
Array.prototype.every



@mathias

Chrome 64

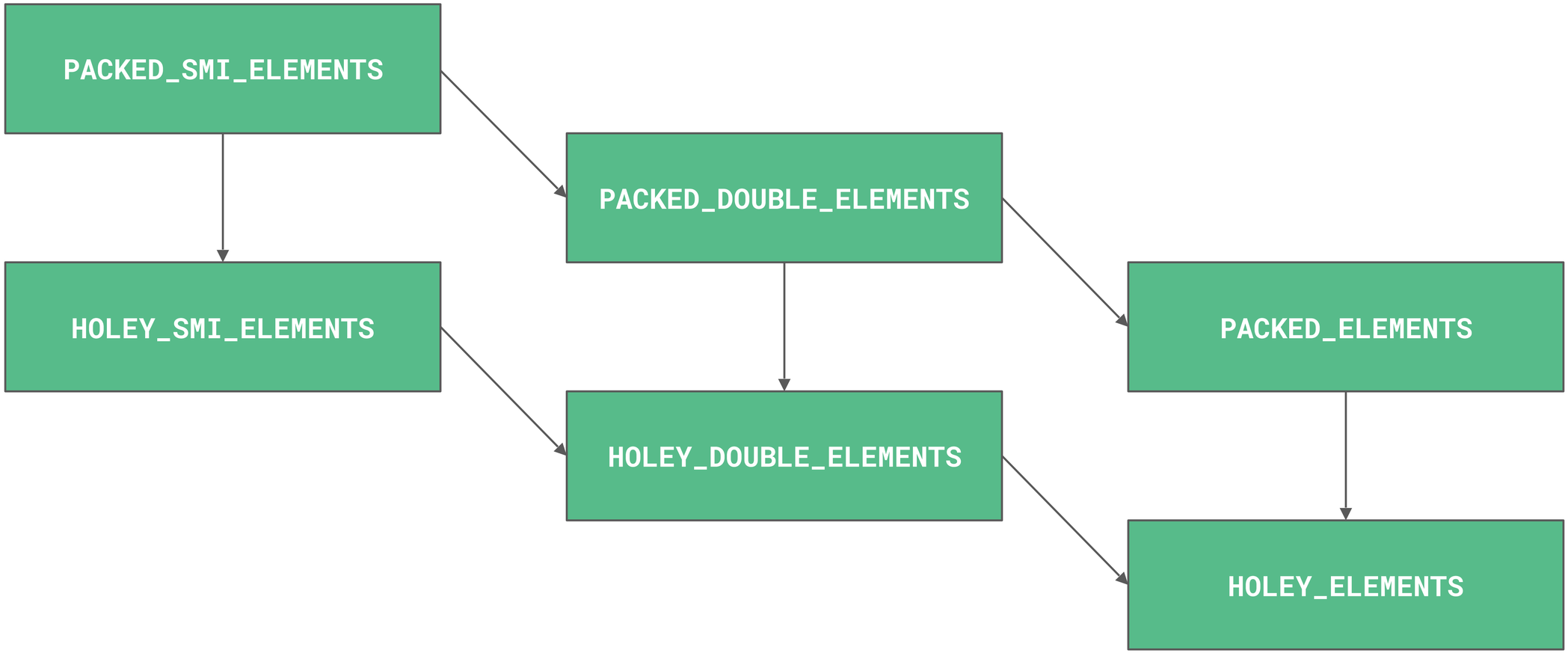
Array.prototype.reduce



@mathias

Chrome 64

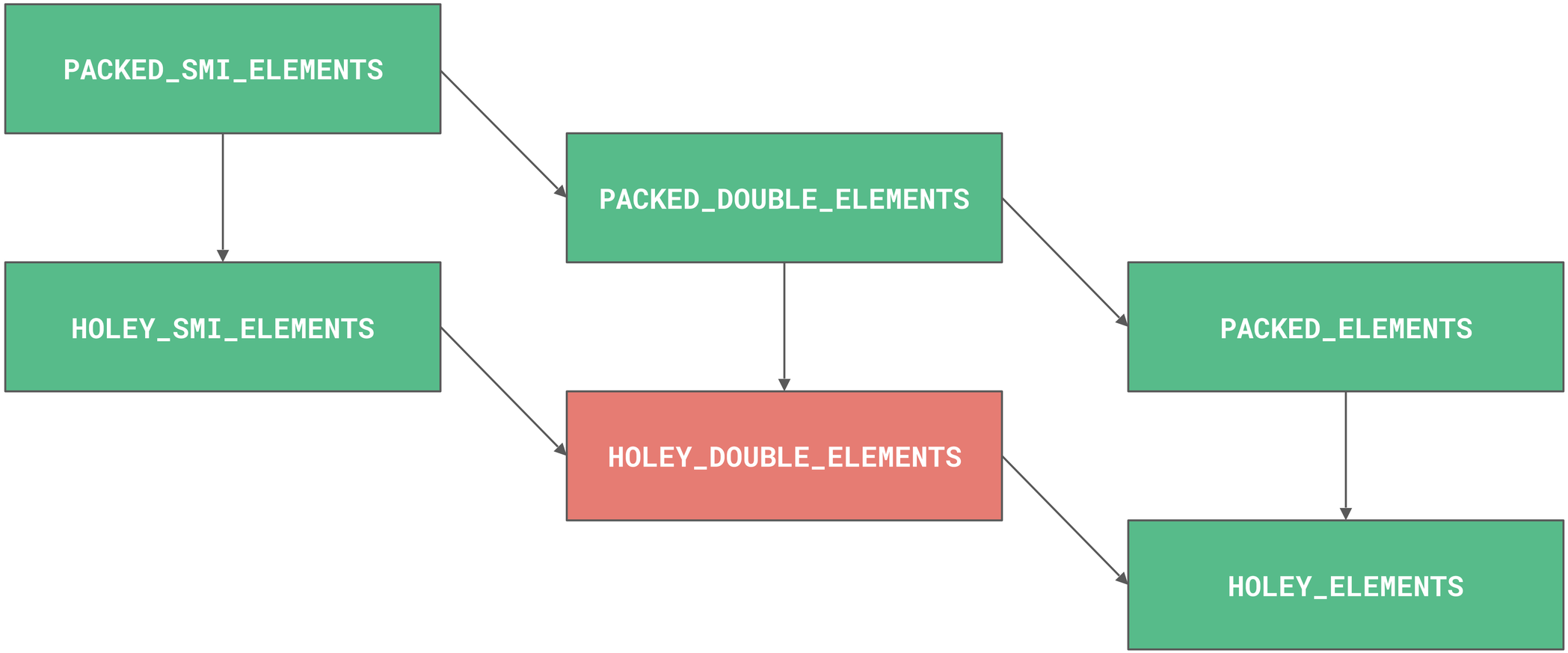
Array.prototype.reduceRight



@mathias

Chrome 64

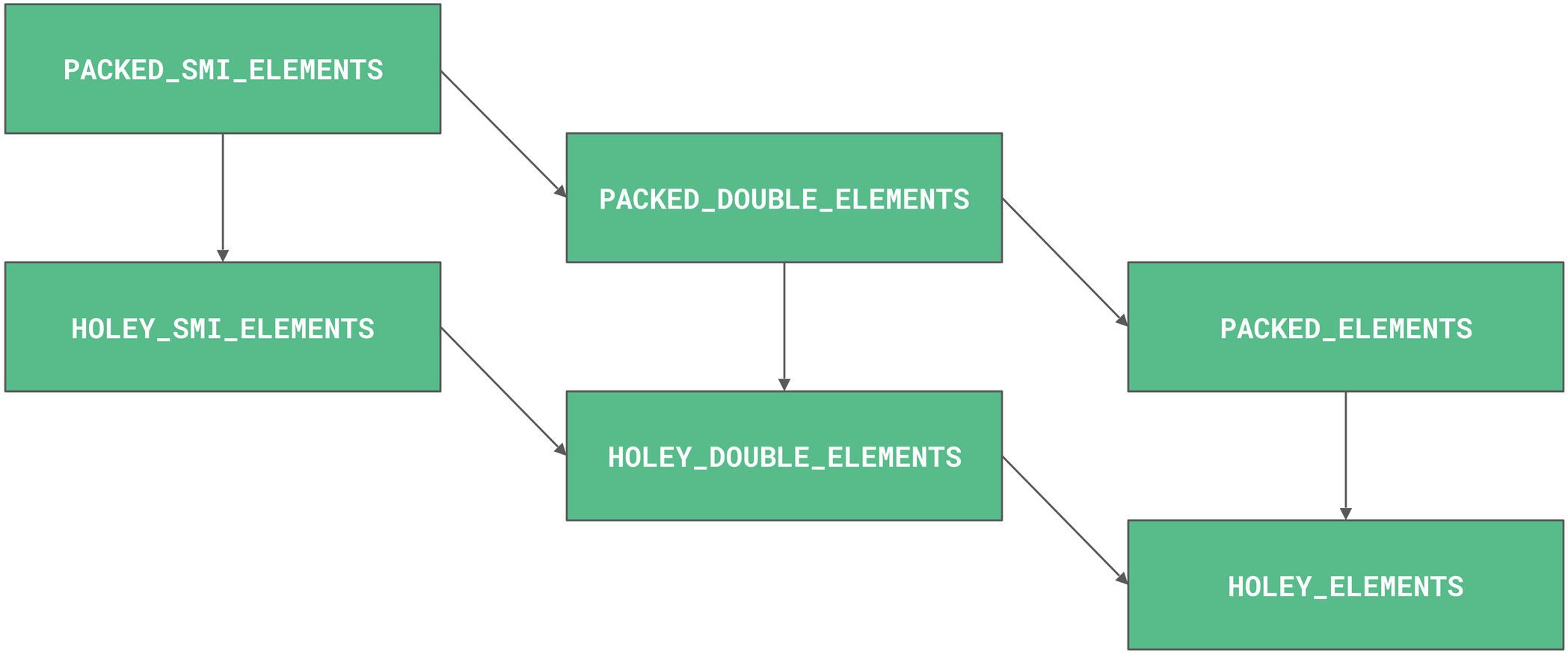
Array#{find, findIndex} 🤦



@mathias

Soon!

Array#{find,findIndex}



@mathias

```
const array = new Array(3);
```

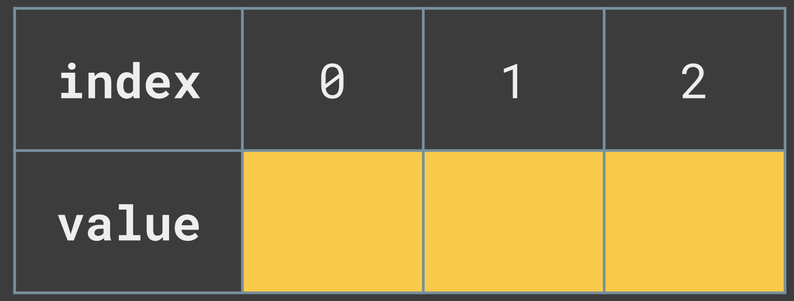
@mathias

```
const array = new Array(3);
```

index	0	1	2
value			

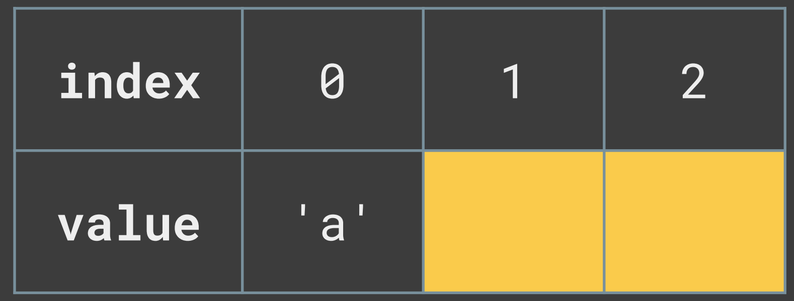
@mathias

```
const array = new Array(3);

// HOLEY_SMI_ELEMENTS
```



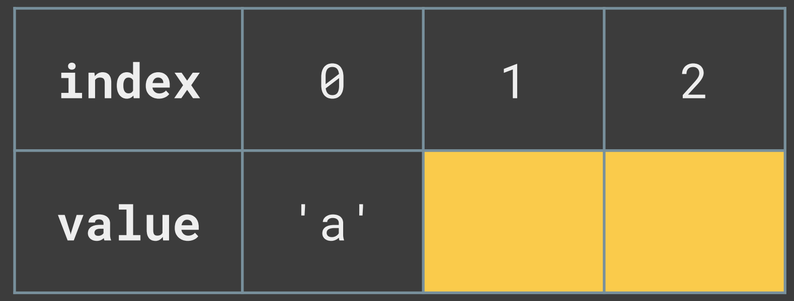
@mathias

```
const array = new Array(3);
// HOLEY_SMI_ELEMENTS
array[0] = 'a';
```



@mathias

```
const array = new Array(3);
// HOLEY_SMI_ELEMENTS
array[0] = 'a';
// HOLEY_ELEMENTS
```



@mathias

```
const array = new Array(3);
// HOLEY_SMI_ELEMENTS
array[0] = 'a';
// HOLEY_ELEMENTS
array[1] = 'b';
```

index	0	1	2
value	'a'	'b'	

@mathias

```
const array = new Array(3);
// HOLEY_SMI_ELEMENTS
array[0] = 'a';
// HOLEY_ELEMENTS
array[1] = 'b';
array[2] = 'c';
```

index	0	1	2
value	'a'	'b'	'c'

🎉 now packed! 🎉

@mathias

```
const array = new Array(3);
// HOLEY_SMI_ELEMENTS
array[0] = 'a';
// HOLEY_ELEMENTS
array[1] = 'b';
array[2] = 'c';
// HOLEY_ELEMENTS (still!)
```

index	0	1	2
value	'a'	'b'	'c'

🎉 now packed! 🎉

but it's too late 😥

@mathias

```
const array = ['a', 'b', 'c'];
// elements kind: PACKED_ELEMENTS
```

@mathias

```
const array = ['a', 'b', 'c'];
// elements kind: PACKED_ELEMENTS


// …
array.push(someValue);
array.push(someOtherValue);
```

@mathias


Avoid holes

```
for (let i = 0, item; (item = items[i]) != null; i++) {
  doSomething(item);
}
```

@mathias

```
for (; item; (item = items[i]) != null; i++) {
  doSo     (item);
}
```

@mathias

```
for (; item; (item = items[i]) != null; i++) {
  doSo     (item);
}
```

```
for (let index = 0; index < items.length; index++) {
  const item = items[index];
  doSomething(item);
}
```

@mathias

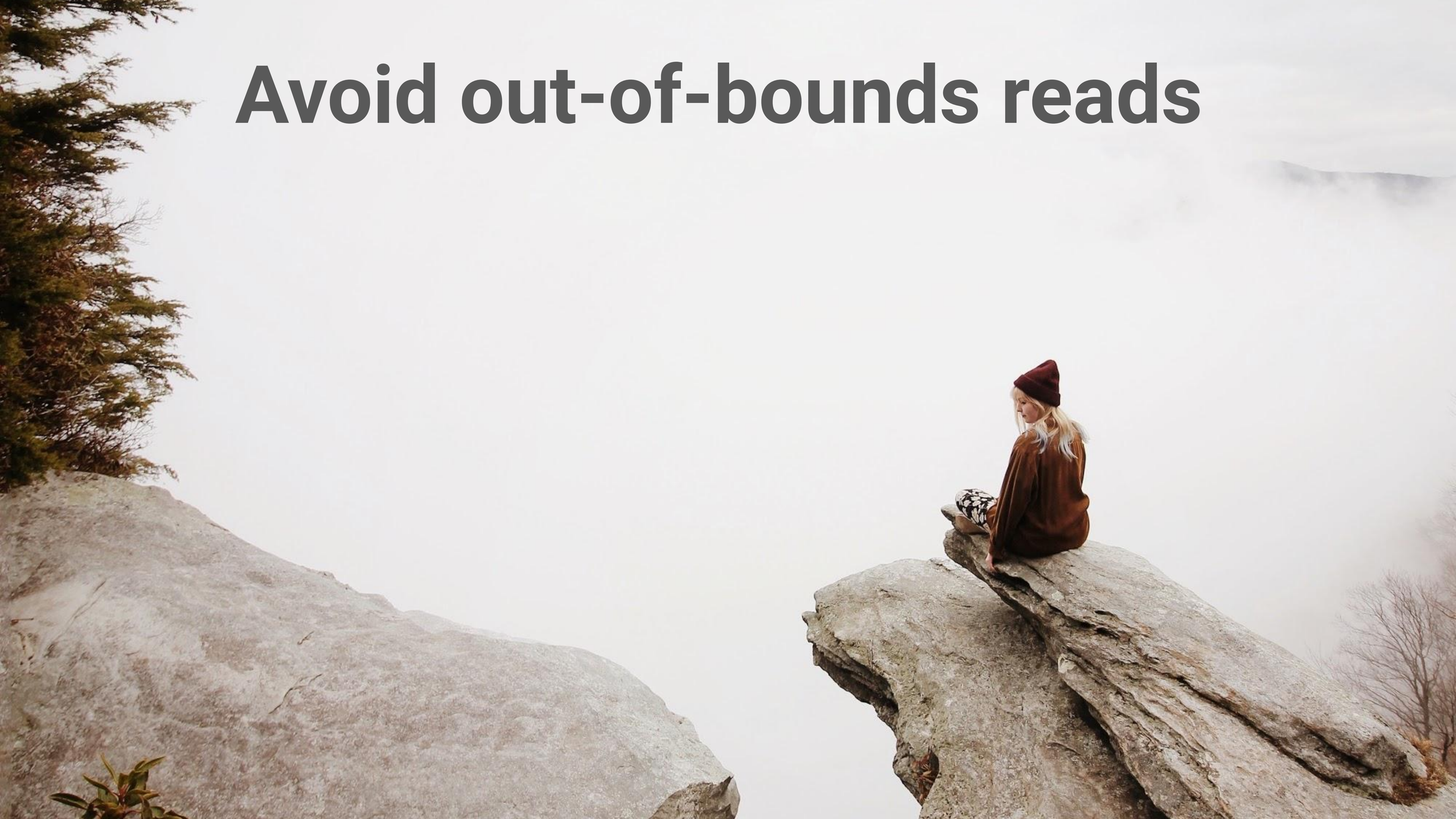
```
for (const item of items) {
  doSomething(item);
}
```

@mathias

```
items.forEach((item) => {
  doSomething(item);
});
```

@mathias

Avoid out-of-bounds reads



```
+0 === -0;

// → true
```

@mathias

```
+0 === -0;

// → true

Object.is(+0, -0);

// → false
```

@mathias

```
[3, 2, 1, +0];
// PACKED_SMI_ELEMENTS
```

@mathias

```
[3, 2, 1, +0];
// PACKED_SMI_ELEMENTS
[3, 2, 1, -0];
// PACKED_DOUBLE_ELEMENTS
```

@mathias

```
[3, 2, 1, +0];
// PACKED_SMI_ELEMENTS
[3, 2, 1, -0];
// PACKED_DOUBLE_ELEMENTS
[3, 2, 1, NaN, Infinity];
// PACKED_DOUBLE_ELEMENTS
```

@mathias

Avoid elements kind transitions

```
const arrayLike = {};
arrayLike[0] = 'a';
arrayLike[1] = 'b';
arrayLike[2] = 'c';
arrayLike.length = 3;
```

@mathias

```
Array.prototype.forEach.call(arrayLike, (value, index) => {
  console.log(`${ index }: ${ value }`);
});
// This logs '0: a', then '1: b', and finally '2: c'.
```

@mathias

```
const actualArray = Array.prototype.slice.call(arrayLike, 0);
actualArray.forEach((value, index) => {
  console.log(`${ index }: ${ value }`);
});
// This logs '0: a', then '1: b', and finally '2: c'.
```

@mathias

```
const logArgs = function() {
  Array.prototype.forEach.call(arguments, (value, index) => {
    console.log(`${ index }: ${ value }`);
  });
};
logArgs('a', 'b', 'c');
// This logs '0: a', then '1: b', and finally '2: c'.
```

@mathias

```
const logArgs = (...args) => {
  args.forEach((value, index) => {
    console.log(`${ index }: ${ value }`);
  });
};
logArgs('a', 'b', 'c');
// This logs '0: a', then '1: b', and finally '2: c'.
```

@mathias

Prefer arrays over array-like objects

$

@mathias

```
$ rlwrap ~/projects/v8/out/x64.debug/d8
```

@mathias

```
$ rlwrap ~/projects/v8/out/x64.debug/d8 --allow-natives-syntax
```

@mathias

```
$ rlwrap ~/projects/v8/out/x64.debug/d8 --allow-natives-syntax
V8 version 6.7.96 (candidate)
d8>
```

@mathias

```
$ rlwrap ~/projects/v8/out/x64.debug/d8 --allow-natives-syntax
V8 version 6.7.96 (candidate)
d8> const array = [1, 2,, 3];
```

@mathias

```
$ rlwrap ~/projects/v8/out/x64.debug/d8 --allow-natives-syntax
V8 version 6.7.96 (candidate)
d8> const array = [1, 2,, 3]; %DebugPrint(array);
```

@mathias

```
$ rlwrap ~/projects/v8/out/x64.debug/d8 --allow-natives-syntax
V8 version 6.7.96 (candidate)
d8> const array = [1, 2,, 3]; %DebugPrint(array);
DebugPrint: 0x313389e0e551: [JSArray]
 - map = 0x3133e0582889 [FastProperties]
 - prototype = 0x313360387f81
 - elements = 0x313389e0e4c9 <FixedArray[4]> [HOLEY_SMI_ELEMENTS (COW)]
 - length = 4
 - properties = 0x3133dae02241 <FixedArray[0]> {
    #length: 0x31336c242839 <AccessorInfo> (const accessor descriptor)
 }
…
```

@mathias

```
$ rlwrap ~/projects/v8/out/x64.debug/d8 --allow-natives-syntax
V8 version 6.7.96 (candidate)
d8> const array = [1, 2,, 3]; %DebugPrint(array);
DebugPrint: 0x313389e0e551: [JSArray]
 - map = 0x3133e0582889 [FastProperties]
 - prototype = 0x313360387f81
 - elements = 0x313389e0e4c9 <FixedArray[4]> [HOLEY_SMI_ELEMENTS (COW)]
 - length = 4
 - properties = 0x3133dae02241 <FixedArray[0]> {
    #length: 0x31336c242839 <AccessorInfo> (const accessor descriptor)
 }
…
```

@mathias

```
$ rlwrap ~/projects/v8/out/x64.debug/d8 --allow-natives-syntax
V8 version 6.7.96 (candidate)
d8> const array = [1, 2,, 3]; %DebugPrint(array);
DebugPrint: 0x313389e0e551: [JSArray]
 - map = 0x3133e0582889 [FastProperties]
 - prototype = 0x313360387f81
 - elements = 0x313389e0e4c9 <FixedArray[4]> [HOLEY_SMI_ELEMENTS (COW)]
 - length = 4
 - properties = 0x3133dae02241 <FixedArray[0]> {
    #length: 0x31336c242839 <AccessorInfo> (const accessor descriptor)
 }
…
```

@mathias

Avoid holes.

— J.K. Rowling

Avoid holes. Avoid out-of-bounds reads.

— ancient Chinese proverb

Avoid holes. Avoid out-of-bounds reads.
Avoid elements kind transitions.

— Justin Bieber

Avoid holes. Avoid out-of-bounds reads. Avoid elements kind transitions. Prefer arrays over array-like objects.

— Albert Einstein

Avoid holes. Avoid out-of-bounds reads. Avoid elements kind transitions. Prefer arrays over array-like objects. Eat your vegetables.

— this slide, just now

One more thing…

```
const array = [
  someValue,
  someOtherValue,
  theLastValue
];
```

@mathias

```
const array = [
  someValue,
  someOtherValue,
  /* more values */,
  theLastValue
];
```

@mathias

```
const array = new Array(9001);
// …
array[0] = someValue;
array[1] = someOtherValue;
// …
array[9000] = theLastValue;
```

@mathias

```
const array = new Array(9001);

// → an array with 9001 holes :'(
```

@mathias

`new Array(n)`

+ allows engines to preallocate space for *n* elements
+ optimizes array **creation**

- creates a *holey* array
- slower array operations (compared to packed arrays)

```
const array = [];
// …
array.push(someValue);
array.push(someOtherValue);
// …
array.push(theLastValue);
```

@mathias

```
array = []; array.push(x);
```

+ creates a *packed* array (never has any holes in it)
+ optimizes array **operations**

- engines need to reallocate space as the array grows
- slower array creation

@mathias

Use `new Array(n)` to optimize the **creation** of the array by pre-allocating the correct number of elements.

Avoid `new Array(n)` to optimize **operations** on the array by avoiding holeyness.

Write modern, idiomatic JavaScript, and let the JavaScript engine worry about making it fast.

Thank you!

@mathias // @v8js

mths.be/v8ek