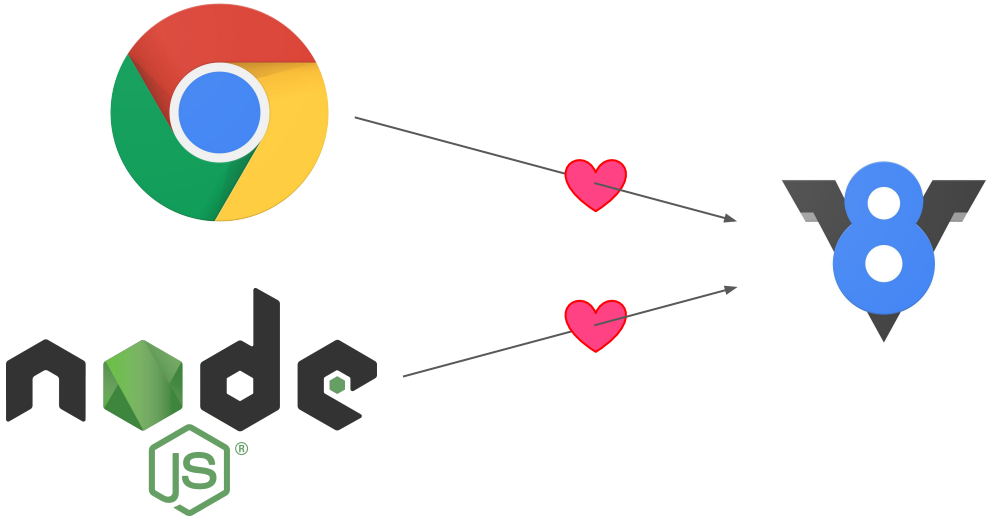
High Performance JS in V8

Peter Marshall, Software Engineer, *Google*

Q: What does fast, modern JavaScript look like?

What is an Engine, Anyway?



- Language Dialects
- Architecture of V8
- Code examples and Benchmarks

Why do we care?

- Servers cost money
- Users like speed
- Transformative use-cases, e.g. Node.js

node JS Interactive
October 4-6, 2017 | Vancouver, BC

Why isn’t JavaScript performance a solved problem?

Why I Still Have a Job

1. The language is growing

Why I Still Have a Job

1. The language is growing
2. High level → Low level mapping requires tradeoffs

Why I Still Have a Job

1. The language is growing
2. High level → Low level mapping requires tradeoffs
3. Performance on low-end mobile is definitely not solved

Why I Still Have a Job

1. The language is growing
2. High level → Low level mapping requires tradeoffs
3. Performance on low-end mobile is definitely not solved
4. **People change the way they use JavaScript**

There are so many ways to do the same thing in JavaScript (and most languages)

- This leads to idioms and styles that form

dialects

Enter:

CrankshaftScript™*

*not really trademarked

CrankshaftScript

"*A JavaScript dialect whose only purpose is to run as fast as possible in V8's Crankshaft compiler*" - me, now.

Q: Can we just write CrankshaftScript and get fast, modern JS?

A: No, we deleted it. Sorry.

```
+2 -130380
```

v8 / v8
mirrored from https://chromium.googlesource.com/v8/v8.git

Unwatch 669 | Star 6,732 | Fork 1,502

Code | Pull requests 0 | Projects 0 | Wiki | Insights

[crankshaft] Remove Crankshaft.

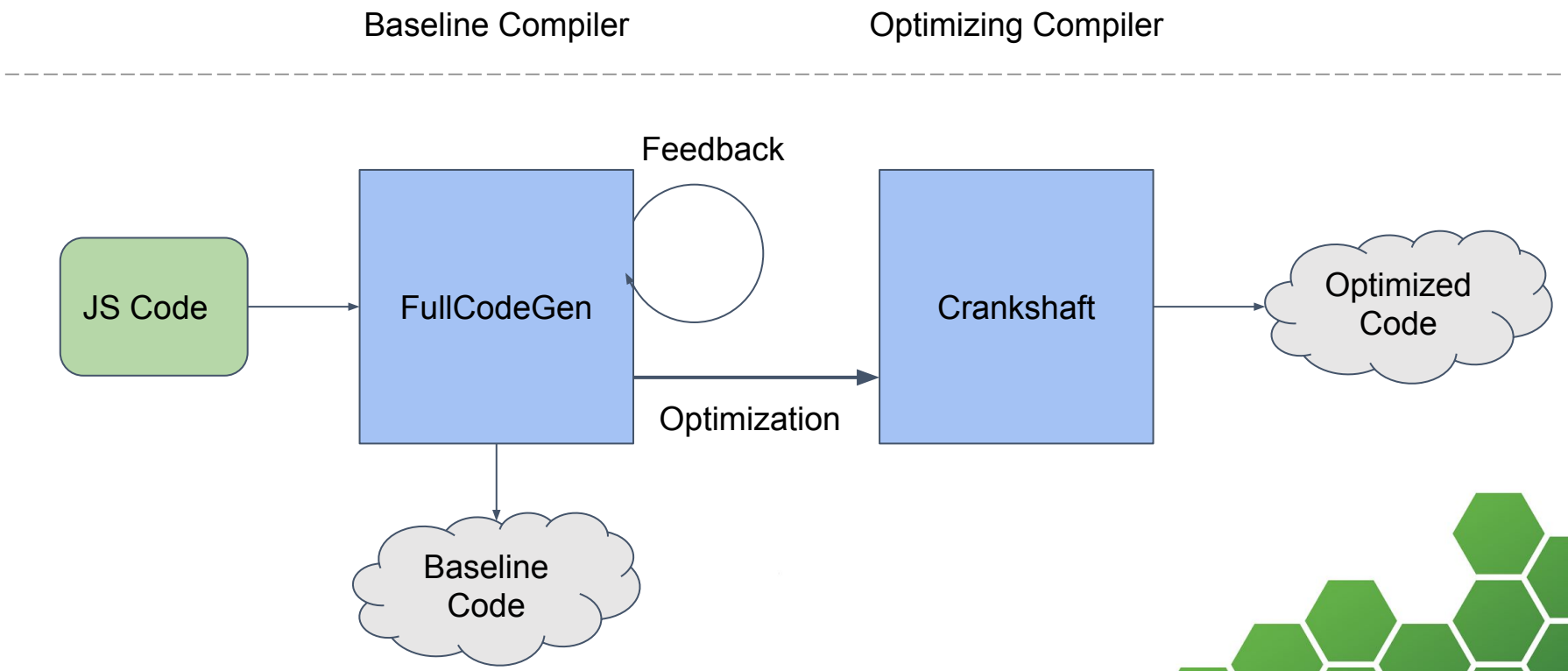
Browse files

```
R=danno@chromium.org
BUG=v8:6408
```

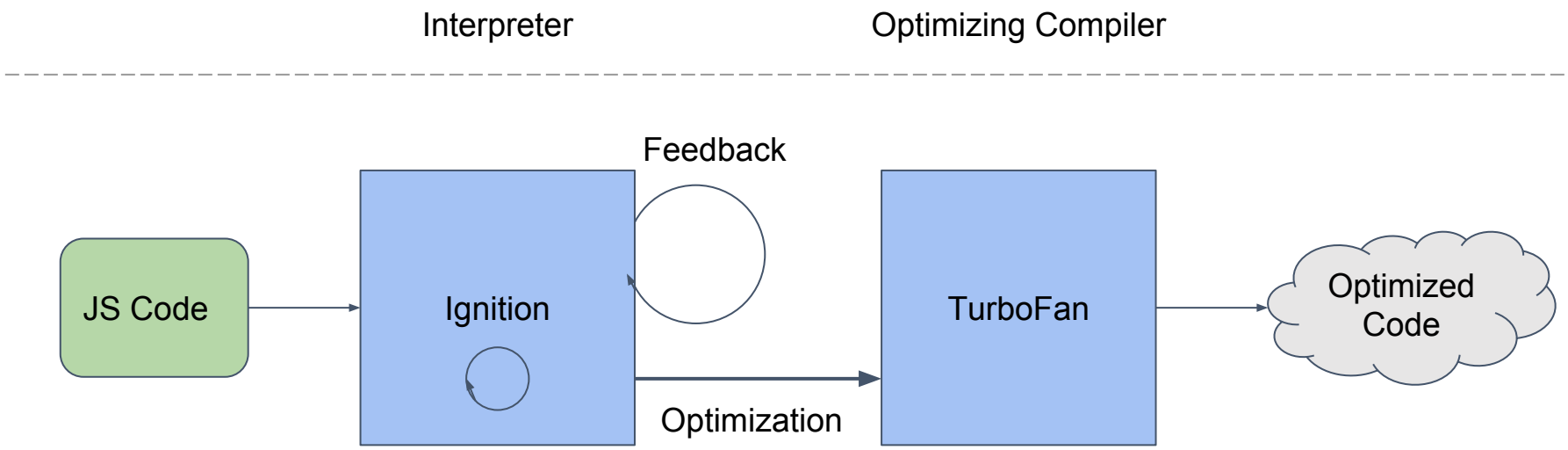
What was Crankshaft?

- Crankshaft was the optimizing compiler in V8.

Old Pipeline in V8



New Pipeline in V8



Problems with Crankshaft

- Crankshaft was starting to show its age:
- Could not support new language features (ES6, etc.)
  - V8 supported them, but not directly in optimized code
- Potential for a lot of deoptimization
- Performance cliffs cause huge, unexpected slowdown
- No clear separation of ‘phases’ of compilation
- A lot of handwritten code required for 9 architectures
- FullcodeGen produced a lot of code, costing memory

TurboFan Goals

- Provide dependable baseline performance
- Wider fast-paths
- Reduce performance cliffs

node JS Interactive

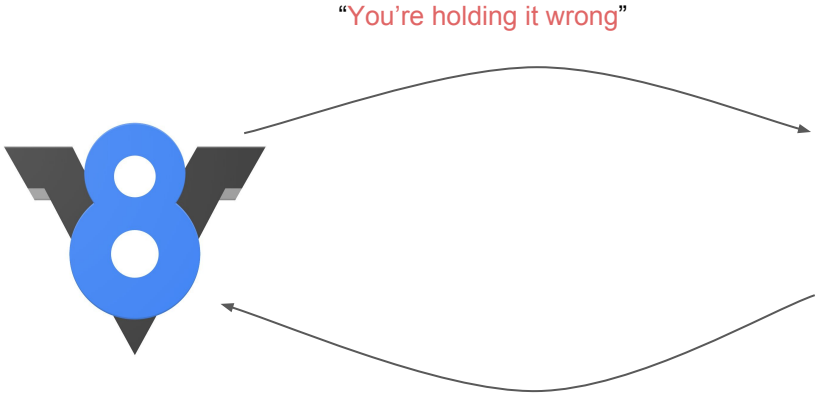
October 4-6, 2017 | Vancouver, BC

Crankshaft Performance Advice

- “Don’t use let/const”
- “Don’t use try/catch or try/finally”
- “Don’t use for-in”
- “Don’t use generators or async functions”

- CrankshaftScript™
- Crankshaft guessed way too much, and had terrible performance on bailout

“Communication” Process



CrankshaftScript 1

```
var validTokens = [0, 1, 0, …  /*256 length */ , 0];

function checkIsHttpToken(val) {
  if (!validTokens[val.charCodeAt(0)])
    return false;
  if (val.length < 2)
    return true;
  if (!validTokens[val.charCodeAt(1)])
    return false;
  if (val.length < 3)
    return true;
  if (!validTokens[val.charCodeAt(2)])
    return false;
  for (var i = 3; i < val.length; ++i) {
    if (!validTokens[val.charCodeAt(i)])
      return false;
  }
  return true;
}
```

"/ This implementation of checkIsHttpToken() loops over the string instead of using a regular expression since the former is up to 180% faster with v8 4.9 */"*

CrankshaftScript 1

```
const token = /^[a-zA-Z0-9_!#$%&'*+.^`|~-]+$/;

function checkIsHttpToken(val) {
  return typeof val === 'string' && token.test(val);
}
```

- Brittle binding between code and engine
- Easily outdated
- Much harder to read or maintain
- We aren’t supporting real-world use-cases

CrankshaftScript 2

V8 5.1

1.27x

V8 6.0

1.00x

(19x)

```
function foo(x) {
  try {
    return bar(x);
  } catch (e) {
    return 0;
  }
}

function bar(x) {
  return x.thingThatMightThrow();
}
```

CrankshaftScript 3

“/ This exists because `Object.create(null)` is absurdly slow compared to `new EmptyObject()`. In either case, you want a null prototype when you're treating the object instances as arbitrary dictionaries and don't want your keys colliding with build-in methods on the default object prototype.*/”*

CrankshaftScript 3

V8 5.1

10x

V8 6.0

1.00x

(5x)

```
const proto = Object.create(null, {
  constructor: {
    value: undefined,
    enumerable: false,
    writable: true
  }
});

function EmptyObject() {}
EmptyObject.prototype = proto;
```

new EmptyObject()
vs.
Object.create(null)

node JS® Interactive

October 4-6, 2017 | Vancouver, BC

What does TurboFanScript look like?

Fast JS for Turbofan

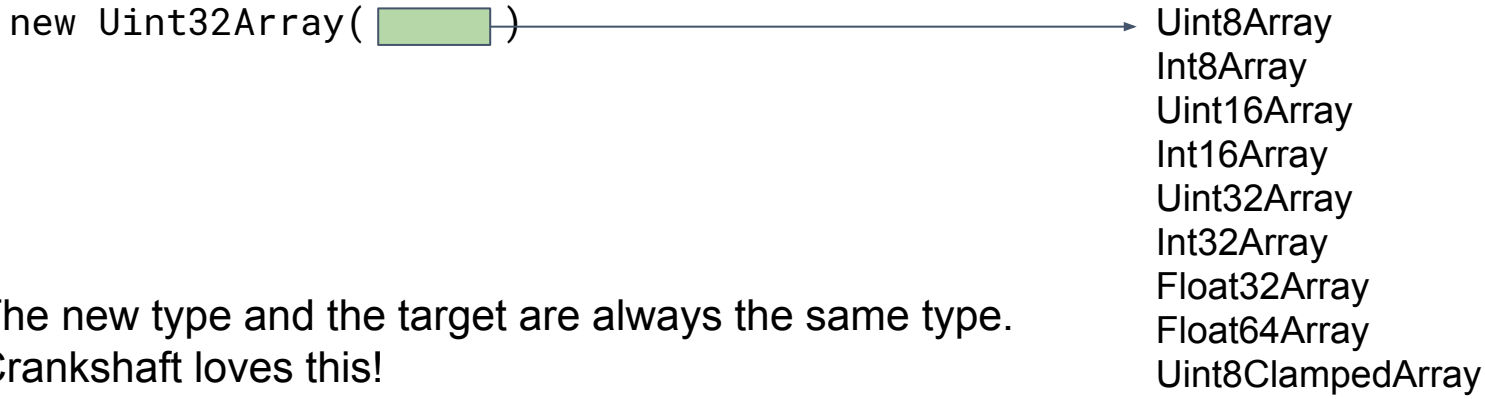
- Regular, readable JavaScript
- Uses ES6 features
- Should work well on any engine

- Ignition + TurboFan are stable in Chrome
- Also released in Node!

A Word About Microbenchmarks

- Microbenchmarks are really great at measuring one thing
- It’s hard to figure out what that one thing is
- They rarely stress optimizing compilers in the right way

Typed Array Example



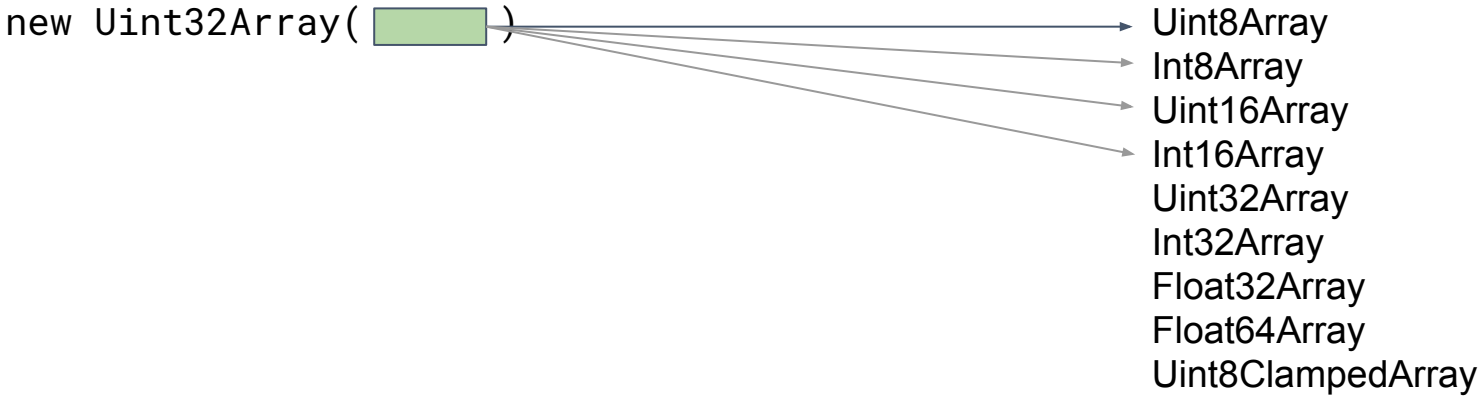
The new type and the target are always the same type.
Crankshaft loves this!
We are measuring *peak performance*.

Typed Array Example

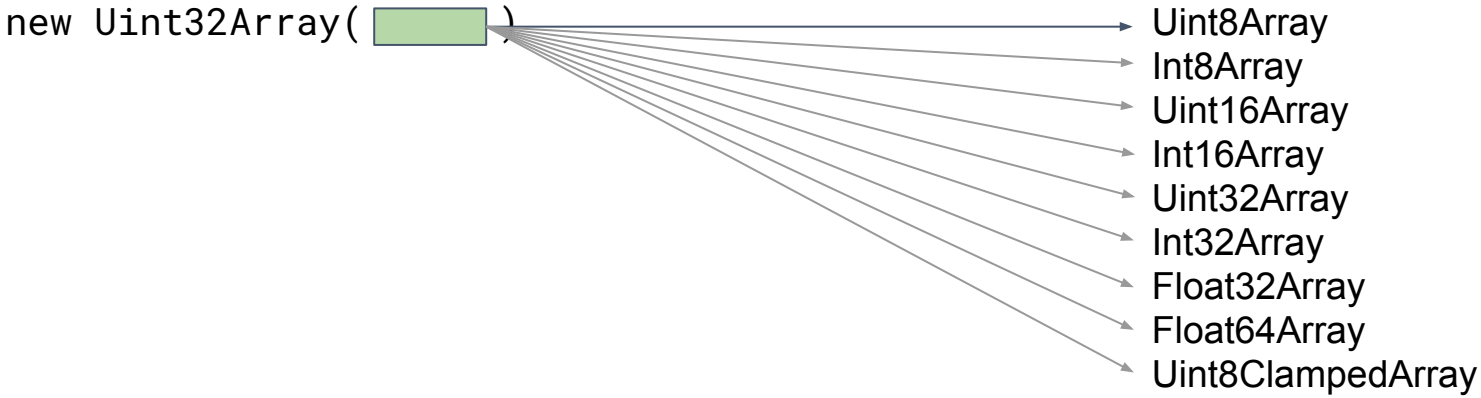
```
new Uint32Array( )
```

- Uint8Array
- Int8Array
- Uint16Array
- Int16Array
- Uint32Array
- Int32Array
- Float32Array
- Float64Array
- Uint8ClampedArray

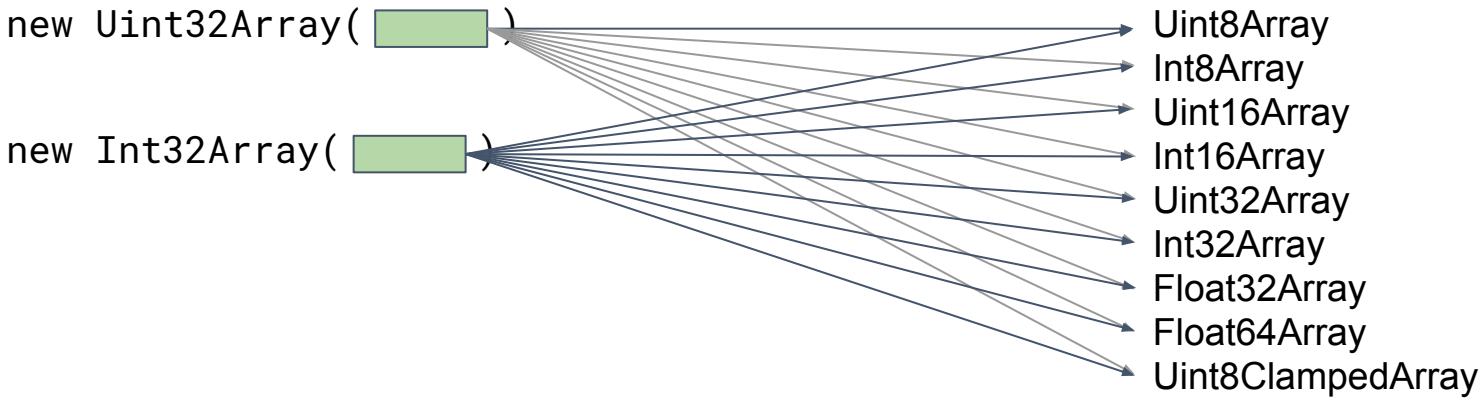
Typed Array Example



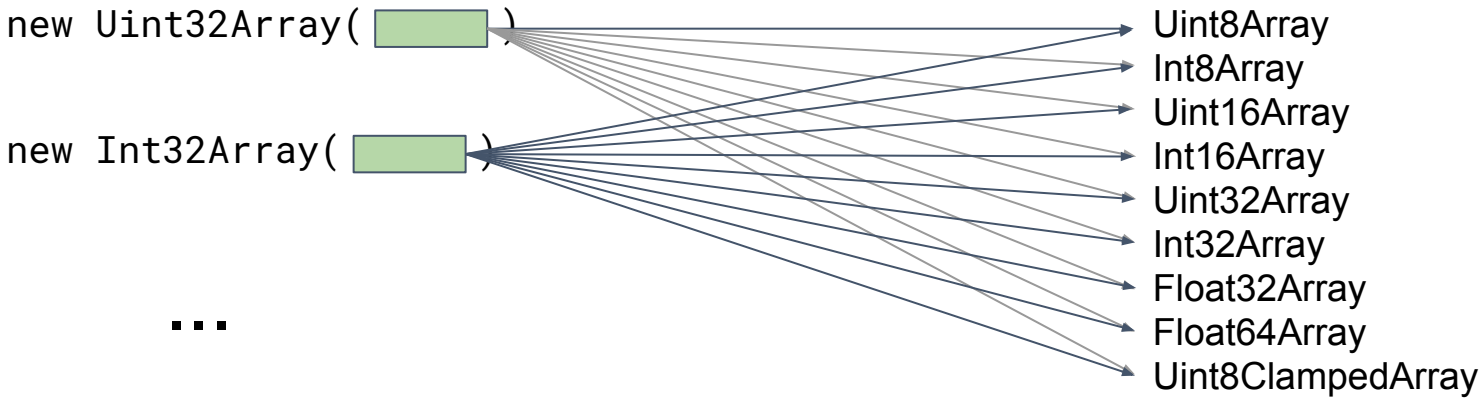
Typed Array Example



Typed Array Example



Typed Array Example

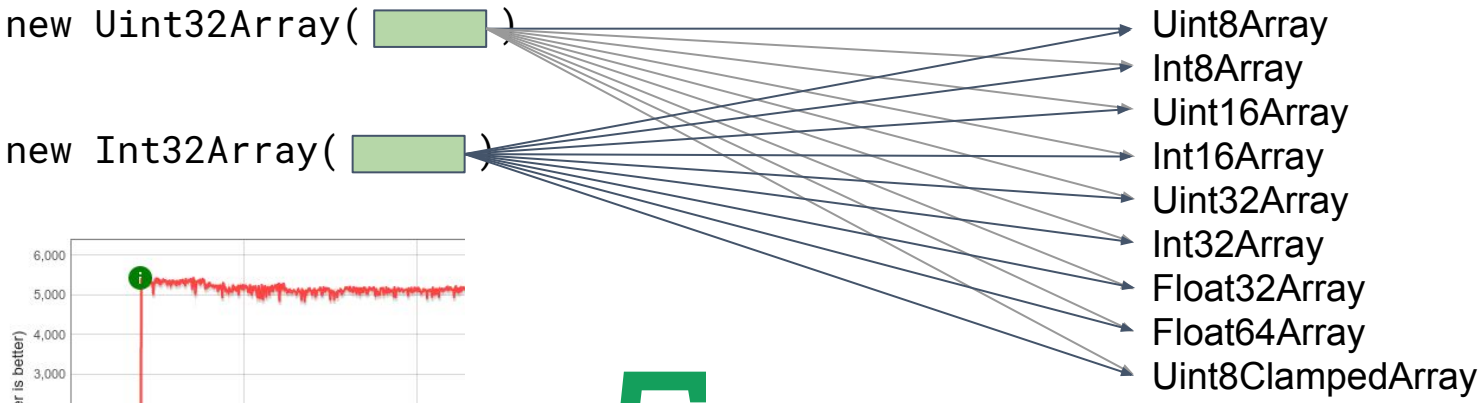


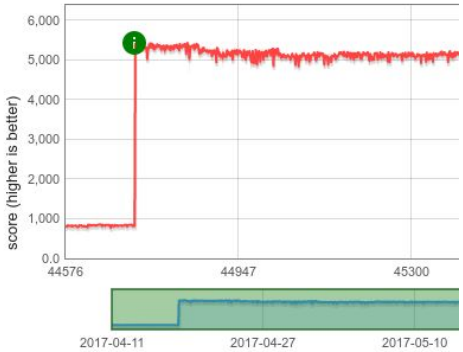
Typed Array Example

Uint8Array
Int8Array
Uint16Array
Int16Array
Uint32Array
Int32Array
Float32Array
Float64Array
Uint8ClampedArray

This benchmark performs equally well in Crankshaft and TurboFan.

Typed Array Example





5x

with TurboFan

Benchmarks

- Peak performance gains are impressive, but don't tell the whole story
- Think about predictability of performance, too
- Get some real-world benchmarks

Takeaways

- No more CrankshaftScript
- Aim for average cases, not peak performance
- We want your use cases

Thanks!

petermarshall@google.com

http://crbug.com/v8

@bmeurer, @mathias, @fhinkel

Check out Yang's talk - Thursday, 11:20am "New DevTools Features for JavaScript"