

BigIntの良いとこ悪いとこ

~ JavaScriptで大きい整数を扱う ~

Kota Yatagai (@kota_yata)

- 19歳
- P2Pネットワークとか暗号技術とか勉強しています
- 仕事はフロントエンドエンジニア&コンテスト運営
- 麻雀とチーズナンカレーにハマった



すごくなりたい
がくせい
ぐるーぶ



(アジアン料理サハラ 狛江店)

2022/07/30 - @kota_yata

JavaScriptにおける大きい整数

通常のNumber型に収まらない値: $2^{53} - 1$ 以上 (9007兆1992億5474万991)

```
Number.MAX_SAFE_INTEGER // 9007199254740991  
Number.MIN_SAFE_INTEGER // -9007199254740991
```

```
9007199254740992 === 9007199254740993 // true
```

```
9007199254740992 < 9007199254740993 // false
```

2^{53} は大きい数ではない

- 暗号処理においては1024ビットの素数が必要になったりする
- UUIDやハッシュ値を数値で保存できない



BigInt

- ES2020で追加されたプリミティブな符号整数型
- ECMAScriptでは最大値が定められていない
 - 実装に任されている（多くのブラウザでは1Mビット）
 - アプリケーション単位でユースケースに合わせて最大値を設定することが推奨されている
- BigInt同士であれば通常の演算子はほとんど使用可能

記法:

```
const bigint: bigint = BigInt(1);
// もしくは
const bigint: bigint = 1n;
```

BigIntの良くないところ

弱点1：既存オブジェクト/型との相性の悪さ

BigIntとNumberを混合した四則演算はTypeErrorになる

```
1 + 1n
// TypeError: Cannot mix BigInt and other types, use explicit conversions
[0, 1n, 2, 3n].sort((a, b) => a - b);
// TypeError: Cannot convert BigInt value to Number value
```

なのに比較演算子は使える

```
1n < 2 // true
2n > 1 // true
2 > 2 // false
2n > 2 // false
```

- Mathオブジェクトにも非対応
- JSONのシリアルライズにも未対応



弱点2：激遅

Testing in Chrome 83.0.4103 / Windows 10 0.0.0		
	Test	Ops/sec
Number multiplication	<code>104723 * 104729</code>	875,220,857 ±0.48% fastest
BigInt multiplication	<code>104723n * 104729n</code>	12,606,366 ±0.37% 99% slower

Number型の演算は**浮動小数点演算**...ハードウェアで実装されている(FPU)

BigInt型に対しては**任意精度演算**を行うためソフトウェアで演算を実装する必要がある



すぐなりたい
がくせい
ぐるーぶ

弱点3：暗号処理には不適切

"Constant-time Operation"がサポートされていない



Constant-time Operation

- ある演算において処理結果を返す時間を定数にすること
- 処理結果が返ってくるまでの時間を計測し、ヒントとして利用する**タイミング攻撃**への対策

例えば...



例：文字列の線形なマッチング

- 正しい文字列を `abcde` とする
- 文字列 `kdisw` とマッチングすると、1文字目で違う文字列であることが分かる。
 - 1文字比較した時点で `false` を返せる
- 文字列 `abcdef` とマッチングすると、6文字目まで違う文字列であることは分からない
 - `kdisw` に比べて処理の終了が遅くなる
- 攻撃者はこれより `abcdef` の方が遙に正しい文字列と近いことが分かってしまう

本来の処理時間に関わらず一定の時間が経ってから結果を返せば攻撃者はヒントを得られなくなる => これがConstant-time Operation



じゃあなんでBigIntの演算でできないの？

- BigIntは可変長であるため、最長でかかる処理時間を予測できない。
- できたとしても全ての値に対してその時間待つのは非効率すぎる
 - 1ビットのマッチングで1Mビットのマッチング分待つのはヤバい

BigInt で対応している演算は、実行時間が一定ではないので、タイミング攻撃を受ける可能性があります。したがって、JavaScript の BigInt は暗号処理での使用には向きません。

(BigInt - developer.mozilla.org)

セキュリティとトレードオフで巨大な整数を扱っているのがBigInt



すごくなりたい
がくせい
ぐるーぶ

BigIntの良いとこ

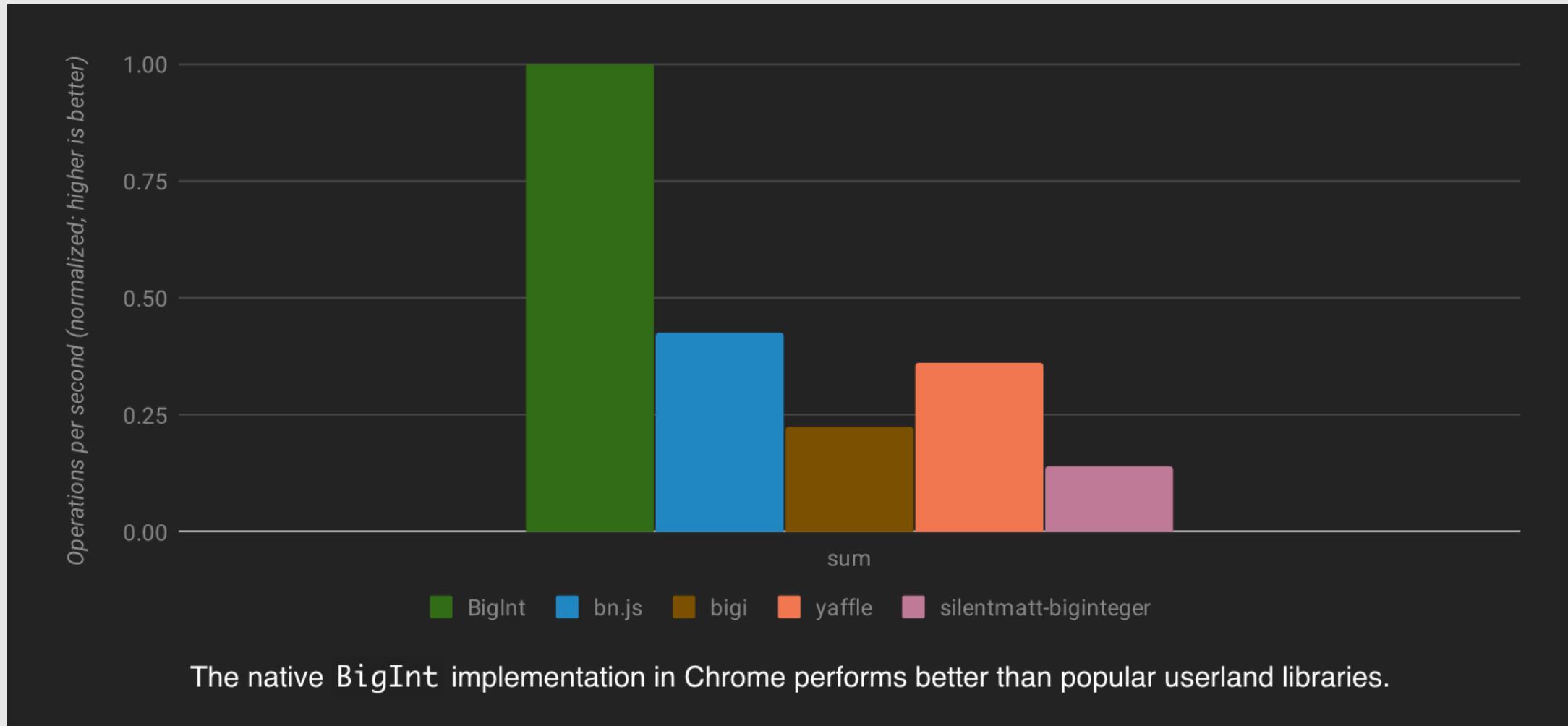
他言語からの値の受け取りとか

- 64ビット固定長など、JavaScriptのNumber型よりも大きい値を保存できる言語は多数(Cのlong longとか)
- 他言語で書かれたサーバーからAPIで値をもらってくる。これまでだったら文字列として扱っていたものをBigInt型に代入でき、数値として演算ができるようになる
- UUIDやハッシュ値も数値として扱えるようになる

実質Uint128Array（かそれ以上）的な使い方ができる



既存サードパーティーライブラリに比べれば格段に速い



(BigInt: arbitrary-precision integers in JavaScript - v8.dev)

2022/07/30 - @kota_yata



さくないたい
がくせい
ぐるーぶ

15

なんか金融系のライブラリにも嬉しいらしい

- 金融系の演算は巨大な整数をたくさん扱う（らしい）

A screenshot of a GitHub issue page. The repository is 'tc39/proposal-bigint'. The issue is titled 'for BigInts in Javascript #53'. It is marked as 'Closed' by 'emschwartz' on Jul 25, 2017, with 12 comments. A comment from 'emschwartz' is highlighted, stating: 'This proposal would also be very useful for financial and crypto applications that currently depend on big number libraries for number manipulation and/or need to represent numbers as strings to avoid losing precision.' The page also shows statistics: Watch 57, Fork 58, Star 561.



おまけ：各主要ブラウザにおけるBigInt実装経緯

SpiderMonkey (FireFox) が先頭を切る

- 2017年5月19日

BigIntがまだステージ2だった段階でMozillaがBigIntスレを立てる

Closed Bug 1366287 (js-bigint) Opened 5 years ago Closed 3 years ago

Implementation of BigInt value

2017-05-19 08:11 PDT onkey

▼ Categories

Product: Core ▾ Component: JavaScript Engine ▾ Type: + enhancement Priority: P2 Severity: normal

▼ Tracking

Status: RESOLVED FIXED

► People (Reporter: terpri, Assigned: terpri)

► References (Depends on 3 open bugs, Blocks 1 open bug)

► Details (Keywords: dev-doc-complete)

▼ Attachments

Show Obsolete

3ヶ月後にJSC (Safari) が実装開始

- 2017年8月8日

プロポーザルがステージ3に昇格した段階でスレが立つ

Caio Lima 2017-08-08 17:05:40 PDT

The BigInt proposal advanced to stage 3 and we should consider implement it.

Reference:

<https://github.com/tc39/proposal-bigint>



すぐなりたい
がくせい
ぐるーぶ

9月にV8 (Chrome) が実装開始

- 2017年9月12日

Comment 3 by bugdroid1@chromium.org on Tue, Sep 12, 2017, 3:56 AM GMT+9 Project Member

The following revision refers to this bug:
<https://chromium.googlesource.com/v8/v8.git/+/0c246c33a36f613078a086fa4f9918b05e665963>

commit [0c246c33a36f613078a086fa4f9918b05e665963](https://chromium.googlesource.com/v8/v8.git/+/0c246c33a36f613078a086fa4f9918b05e665963)
Author: Georg Neis <neis@chromium.org>
Date: Mon Sep 11 18:55:48 2017

[bigint] Introduce BigInt type.

BigInt is a new primitive type of arbitrary precision integers, proposed in <https://tc39.github.io/proposal-bigint>.

This CL introduces a corresponding instance type, map, and C++ class to V8 and adds BigInt support to a few operations (see the test file). Much more is to come. Also, the concrete representation of BigInts is not yet fixed, currently a BigInt is simply a wrapped Smi.



すごくなりたい
がくせい
ぐるーぶ

20

フォーク案とか再実装とか色々案が出る

JSCではV8からフォークして実装を開始する案、SpiderMonkeyでは割と実装が進んだ頃になってやっぱ違うライブラリ使った方が良かった説のスレッドが立つが結局独自実装でShippingまで進むことになる

As Dan pointed out, I started to implement a library based in libtommath/CPython BigInts but it is in an early stage. Mozilla is using GMP because they do not have any license problem but the other browsers do. My advice would be to start with the V8's implementation as you said.

Closed Bug 1502797 Opened 4 years ago Closed 4 years ago

Reimplement BigInt in terms of JSC/V8 code instead of GMP



すごくなりたい
がくせい
ぐるーぶ

まとめ



BigIntの良いとこ

- 大きい整数が扱える
- これまでのサードパーティライブラリに比べると格段に速い

BigIntの悪いとこ

- Number型に比べて制約が多く、実装時に混乱する
- Number型に比べると遅い（それはそう）
- 暗号処理には向かない



参考文献



SpiderMonkey: Threads and Implementations

https://bugzilla.mozilla.org/show_bug.cgi?id=1502797

https://bugzilla.mozilla.org/show_bug.cgi?id=1494346

https://bugzilla.mozilla.org/show_bug.cgi?id=1366287

<https://github.com/mozilla/standards-positions/issues/65>

https://developer.mozilla.org/docs/Web/JavaScript/Reference/Global_Objects/BigInt

<https://gmplib.org/>

V8: Threads, Blog Posts and Implementations

<https://v8.dev/blog/bigint>

<https://v8.dev/features/bigint>

<https://github.com/v8/v8/blob/master/src/objects/bigint.cc>

<https://bugs.chromium.org/p/v8/issues/detail?id=6791>



すごくなりたい
がくせい
ぐるーぶ

JSC: Threads and Implementations

https://bugs.webkit.org/show_bug.cgi?id=175359

<https://lists.webkit.org/pipermail/webkit-dev/2017-October/029676.html>

<https://trac.webkit.org/browser/webkit/trunk/Source/JavaScriptCore/runtime/JST BigInt.cpp>

<https://github.com/WebKit/WebKit/commit/954a77a437aee376dcc4c3bd8ec5e112fa9e03a1>

TC39: Proposals and Specs

<https://github.com/tc39/proposal-bigint/issues/53>

<https://tc39.es/proposal-bigint/>

<https://tc39.es/process-document/>



すごくなりたい
がくせい
ぐるーぶ

26

Others

<https://golb.hplar.ch/2018/09/javascript-bigint.html>

<https://betterprogramming.pub/the-downsides-of-bignums-in-javascript-6350fd807d>

https://inst.eecs.berkeley.edu/~cs10/labs/cur/programming/algorithms/timing/constant-time.html?topic=berkeley_bjc%2Fareas%2Falgorithm-complexity.topic

<https://www.tektutorialshub.com/javascript/bigint-vs-number-in-javascript/#bigint-is-an-integer-number-is-a-decimal>

https://en.wikipedia.org/wiki/Double-precision_floating-point_format#JavaScript



すごくなりたい
がくせい
ぐるーぶ