perplexity

why big int is required, how much bit number uses

BigInt is required in JavaScript to accurately represent and work with integers larger than what the standard Number type can handle without losing precision.

Why is BigInt required?

- JavaScript's regular Number type is based on the IEEE 754 double-precision floating-point format, which can only safely represent integers in the range from **-9,007,199,254,740,991** to **9,007,199,254,740,991** (that is, from $-(2^{53}-1)$ to $2^{53}-1$)[1] [2] [3].
- Numbers beyond this range cannot be exactly represented—they get rounded, causing *loss of precision* in calculations. This is problematic for fields like scientific computing, cryptography, financial systems, or any case needing very large integer values[1] [4] [3].
- **BigInt** was introduced so JavaScript can work safely and efficiently with integers of *arbitrary size*, well beyond the safe integer limit, without precision loss[1] [4] [3] [5].

How many bits does Number use? What about BigInt?

- **Number**:
  - JavaScript's Number type uses a **64-bit format** (double-precision floating point as per IEEE 754). However, only **53 bits** of that are used for the integer part's precision: 1 for sign, 52 for significand/mantissa[1] [6].
  - Therefore, integers can only be stored exactly within $\pm 2^{53}-1$.
- **BigInt**:
  - *Does not have a fixed bit-width.* BigInts can store integers as large as your available memory allows—they use as much memory as needed for the value[3] [5] [6].
  - This makes BigInt suitable for tasks requiring very large integer values, such as high-resolution timestamps, IDs, cryptographic keys, or mathematical computations that exceed the 53-bit safe integer limit[4] [5].
- For reference, BigInts in WebAssembly are often used to represent **64-bit integers** directly, which Numbers cannot exactly encode[7].

In summary:

- Use BigInt when you need precise integer arithmetic for values larger than ±9,007,199,254,740,991.
- Number is a 64-bit floating-point type (53 bits for integer precision).
- BigInt can grow to hundreds or thousands of bits (practically limited only by available memory)[1] [2] [3] [5] [6].

1. https://www.w3schools.com/js/js_bigint.asp
2. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/BigInt
3. https://blog.nashtechglobal.com/javascript-bigint-handling-large-numbers-with-precision-and-efficiency/
4. https://www.smashingmagazine.com/2019/07/essential-guide-javascript-newest-data-type-bigint/
5. https://blog.logrocket.com/how-to-use-javascript-bigint/
6. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Data_structures
7. https://v8.dev/features/wasm-bigint