BigInt

Новая возможность

Эта возможность была добавлена в язык недавно. Узнать, где есть поддержка, можно на https://caniuse.com/#feat=bigint.

`BigInt` – это специальный числовой тип, который предоставляет возможность работать с целыми числами произвольной длины.

Чтобы создать значение типа `BigInt`, необходимо добавить `n` в конец числового литерала или вызвать функцию `BigInt`, которая создаст число типа `BigInt` из переданного аргумента. Аргументом может быть число, строка и др.

```
const bigint = 1234567890123456789012345678901234567890

const sameBigint = BigInt("1234567890123456789012345678

const bigintFromNumber = BigInt(10); // то же самое, чт
```

Математические операторы

`BigInt` можно использовать как обычные числа, к примеру:

```
alert(1n + 2n); // 3

alert(5n / 2n); // 2
```

Обратите внимание: операция деления `5/2` возвращает округлённый результат, без дробной части. Все операции с числами типа `bigint` возвращают `bigint`.

В математических операциях мы не можем смешивать `bigint` и обычные числа:

```
alert(1n + 2); // Error: Cannot mix BigInt and other ty
```

Мы должны явно их конвертировать: используя либо `BigInt()`, либо `Number()`, например:

```
let bigint = 1n;
let number = 2;

// конвертируем number в bigint
alert(bigint + BigInt(number)); // 3

// конвертируем `bigint` в number
alert(Number(bigint) + number); // 3
```

Конвертирование `bigint` в число всегда происходит неявно и без генерации ошибок, но если значение `bigint` слишком велико и не подходит под тип `number`, то дополнительные биты будут отброшены, так что следует быть осторожными с такими преобразованиями.

К `BigInt` числам нельзя применить унарный оператор `+`

Унарный оператор `+value` является хорошо известным способом конвертировать произвольное значение `value` в число.

Данный оператор не поддерживается при работе с `BigInt` числами.

```
let bigint = 1n;

alert( +bigint ); // SyntaxError: Unexpected ident
```

Операции сравнения

Операции сравнения, такие как `<`, `>`, работают с `bigint` и обычными числами как обычно:

```
alert( 2n > 1n ); // true

alert( 2n > 1 ); // true
```

Пожалуйста, обратите внимание, что обычные и `bigint` числа принадлежат к разным типам, они могут быть равны только при нестрогом сравнении `==`:

```
alert( 1 == 1n ); // true

alert( 1 === 1n ); // false
```

Логические операции

В `if` или любом другом логическом операторе `bigint` число ведёт себя как обычное число.

К примеру, в `if bigint 0n` преобразуется в `false`, другие значения преобразуются в `true`:

```
if (0n) {
  // никогда не выполнится
}
```

Логические операторы `||`, `&&` и другие также работают с `bigint` числами как с обычными числами:

```
alert( 1n || 2 ); // 1

alert( 0n || 2 ); // 2
```

Полифилы

Создание полифила для `BigInt` – достаточно непростая задача. Причина в том, что многие операторы в JavaScript, такие как `+`, `-` и др., ведут себя по-разному с `bigint` по сравнению с обычными числами.

К примеру, деление `bigint` числа всегда возвращает `bigint` (округлённое при необходимости).

Чтобы эмулировать такое поведение, полифил должен будет проанализировать код и заменить все такие операторы на свои вызовы. Такая реализация будет тяжеловесной, не очень хорошей с точки зрения производительности.

Вот почему на данный момент нет хорошо реализованного полифила.

Существует обратное решение, предложеное разработчиками библиотеки JSBI.

Эта библиотека реализует большие числа, используя собственные методы. Мы можем использовать их вместо встроенных `BigInt`:

Операция	Встроенный `BigInt`	JSBI
Создание из `number`	`a = BigInt(789)`	`a = JSBI.BigInt(789)`
Сложение	`c = a + b`	`c = JSBI.add(a, b)`
Вычитание	`c = a - b`	`c = JSBI.subtract(a, b)`
...	...	...

...А затем использовать полифил (плагин Babel) для замены вызовов JSBI на встроенные `Bigint` для браузеров, которые их поддерживают.

Другими словами, данный подход предлагает использовать JSBI вместо встроенных `BigInt`. JSBI внутри себя работает с числами как с `BigInt`, эмулирует их с соблюдением всех требований спецификации. Таким образом, мы можем выполнять JSBI-код в интерпретаторах, которые не поддерживают `Bigint`, а для тех, которые поддерживают – полифил преобразует вызовы в обычные `Bigint`.

Ссылки

- MDN: BigInt.
- Спецификация: BigInt.