- Если значением любого из операндов является объект, он преобразуется в простое значение с использованием алгоритма преобразования объекта в простое значение, описанного в разделе 3.8.3: объекты `Date` преобразуются с помощью их метода `toString()`, а все остальные объекты преобразуются с помощью метода `valueOf()`, если он возвращает простое значение. Однако большинство объектов не имеют метода `valueOf()`, поэтому они также преобразуются с помощью метода `toString()`.
- Если после преобразования объекта в простое значение любой из операндов оказывается строкой, другой операнд также преобразуется в строку и выполняется операция конкатенации.
- В противном случае оба операнда преобразуются в числа (или в `NaN`) и выполняется операция сложения.

Например:

```
1 + 2          // => 3: сложение
"1" + "2"      // => "12": конкатенация
"1" + 2        // => "12": конкатенация после преобразования числа в строку
1 + {}         // => "1[object Object]": конкатенация после
               // преобразования объекта в строку
true + true    // => 2: сложение после преобразования логического значения в число
2 + null       // => 2: сложение после преобразования null в 0
2 + undefined  // => NaN: сложение после преобразования undefined в NaN
```

Наконец, важно отметить, что, когда оператор + применяется к строкам и числам, он может нарушать ассоциативность. То есть результат может зависеть от порядка, в каком выполняются операции. Например:

```
1 + 2 + " blind mice";   // => "3 blind mice"
1 + (2 + " blind mice"); // => "12 blind mice"
```

В первом выражении отсутствуют скобки и оператор + имеет ассоциативность слева направо, благодаря чему сначала выполняется сложение двух чисел, а их сумма объединяется со строкой. Во втором выражении порядок выполнения операций изменен с помощью скобок: число 2 объединяется со строкой, давая в результате новую строку. А затем число 1 объединяется с новой строкой, что дает окончательный результат.