Оператор равенства == похож на оператор идентичности, но он использует менее строгие правила. Если значения операндов имеют разные типы, он выполняет преобразование типов и пытается выполнить сравнение:

- Если два значения имеют одинаковый тип, они проверяются на идентичность, как было описано выше. Если значения идентичны, они равны; если они не идентичны, они не равны.
- Если два значения не относятся к одному и тому же типу, оператор == все же может счесть их равными. При этом используются следующие правила и преобразования типов:

- Если одно значение `null`, а другое – `undefined`, то они равны.
- Если одно значение является числом, а другое – строкой, то строка преобразуется в число и выполняется сравнение с преобразованным значением.
- Если какое-либо значение равно `true`, оно преобразуется в 1 и сравнение выполняется снова. Если какое-либо значение равно `false`, оно преобразуется в 0 и сравнение выполняется снова.
- Если одно из значений является объектом, а другое – числом или строкой, объект преобразуется в простой тип (как описывалось в разделе 3.8.3) и сравнение выполняется снова. Объект преобразуется в значение простого типа либо с помощью своего метода `toString()`, либо с помощью своего метода `valueOf()`. Встроенные классы базового языка JavaScript сначала пытаются выполнить преобразование `valueOf()`, а затем `toString()`, кроме класса `Date`, который всегда выполняет преобразование `toString()`. Объекты, не являющиеся частью базового JavaScript, могут преобразовывать себя в значения простых типов способом, определенным их реализацией.
- Любые другие комбинации значений не являются равными.

В качестве примера проверки на равенство рассмотрим сравнение:

```
"1" == true
```

Результат этого выражения равен `true`, т. е. эти по-разному выглядящие значения фактически равны. Логическое значение `true` преобразуется в число 1, и сравнение выполняется снова. Затем строка `"1"` преобразуется в число `1`. Поскольку оба числа теперь совпадают, оператор сравнения возвращает `true`.