Для того, чтобы понять как в JavaScript работает строгое сравнение (===), нужно знать несколько базовых для него (языка) концепций:

1) В JavaScript нет переменных, а есть только идентификаторы, которые содержат в себе ссылку на данные. То есть когда Вы пишите например:

*var ident1 = 10;*

*var ident2 = { name: ‘someName’ };*

*var ident3 = ’some string’;*

То *ident1* , *ident2* , *ident3* содержат в себе ссылки на структуры описывающие данные. То есть, вопреки тому, что Вам рассказывают *ident1* и *ident3* не содержит сами данные, но так же как и *ident2* , содержат ссылку на структуру данных.

2) Данные в JS иммутабельные. То есть неизменны. На языке спецификаций данные - это реальные физические байты информации, которые обозначают термином термином Primitive Value\_ То есть объявив строку, средствами языка вы не можете ее изменить. Изменения исходной строки (или любых других Primitive Value), приводят к созданию новой. Следствием этого является тот факт, что когда Вы пишите например:

*var str1=”my string”;*

*var str2=”my string”;*

то идентификатор str1 и str2 содержит в себе одну и туже ссылку на одни и те же данные. То есть RunTime не создает две структуры данных с одной и той же строкой. То есть RunTime строго следит за тем, чтобы те Primitive Value которые уже описаны, и заявляются снова в нашем коде, получали ту же самую ссылку.

3) Строгое сравнение в JS (===) сравнивает только ссылки из идентификаторов. Никакого сравнения типов и прочей чепухи о которой Вам рассказывают - нет. Потому, что это не нужно учитывая все 3 пункта. Например:

*var myNum=3;*

*var myEnotherNum=3;*

*myNum === myEnotherNum;* // true

Потому что оба идентификатора содержат одну и туже ссылку на структуру описывающую Primitive Value: 3. или

*var myObj = { name: ‘someName’ };*

*var myObj2 = myObj;*

*var myEnotherObj2 = { name: ‘someName’ };*

*myObj === myObj2;* // true

*myObj === myEnotherObj2;* // false

*myObj['name'] === myEnotherObj2['name'];* // true