**[What is instance of an object in JavaScript?](https://www.quora.com/What-is-instance-of-an-object-in-JavaScript?no_redirect=1" \t "_top)**

Let us discuss the technical aspect first.

An object is a programmatic representation of a real-life entity.

You write code to tell the computer what a particular entity (aka object) is like. Your code tells the computer what the properties of an object are as well as the things (methods) that an object can do:

1. **var** person = {
2. name: 'Juan',
3. age: 40,
4. gender: 'male',
5. greeting: **function**() {
6. alert ('Hi! I\'m ' + **this**.name + '.');
7. }
8. };

You essentially provide a blueprint.

During run time (when your code is executing), you will issue an instruction to create an ***instance*** of this **object**in order to use it:

1. **var** guy = **new** person ();

The **runtime** will go look up the definition of an object of type “**person**” (your blueprint), allocate memory, initialize it and create a pointer named **“guy”** that allows you to refer to this **instance**.

Now let’s make sense of this at a higher level.

A good analogy of this would be a cookbook. It may contain a recipe to make a delicious Potato soup, but you cannot taste the definition written on the pages. There is no soup. Not yet anyways. You have to make one (**instance**) in order to interact with it.

Now, you may make soup in the morning and then realize you want more at night, so you go ahead and create a second **instance.**They are both created using the same recipe, but they are not the same entity (aka object).

**Instanceof**

The Left Hand Side (LHS) operand is the actual object being tested to the Right Hand Side (RHS) operand which is the actual constructor of a class. The basic definition is:

Checks the current object and returns true if the object

is of the specified object type.

Here are some [good examples](http://www.herongyang.com/JavaScript/Prototype-instanceof-Operator-Determine-Object-Type.html) and here is an example taken directly from [Mozilla's developer site](https://developer.mozilla.org/En/Core_JavaScript_1.5_Reference/Operators/Special_Operators/Instanceof_Operator):

var color1 = new String("green");

color1 instanceof String; // returns true

var color2 = "coral"; //no type specified

color2 instanceof String; // returns false (color2 is not a String object)

One thing worth mentioning is instanceof evaluates to true if the object inherits from the class's prototype:

var p = new Person("Jon");

p instanceof Person

That is p instanceof Person is true since p inherits from Person.prototype.

**Per the OP's request**

I've added a small example with some sample code and an explanation.

When you declare a variable you give it a specific type.

For instance:

int i;

float f;

Customer c;

The above show you some variables, namely i, f, and c. The types are integer, float and a user defined Customer data type. Types such as the above could be for any language, not just JavaScript. However, with JavaScript when you declare a variable you don't explicitly define a type, var x, x could be a number / string / a user defined data type. So what instanceof does is it checks the object to see if it is of the type specified so from above taking the Customer object we could do:

var c = new Customer();

c instanceof Customer; //Returns true as c is just a customer

c instanceof String; //Returns false as c is not a string, it's a customer silly!

Above we've seen that c was declared with the type Customer. We've new'd it and checked whether it is of type Customer or not. Sure is, it returns true. Then still using the Customer object we check if it is a String. Nope, definitely not a String we newed a Customer object not a String object. In this case, it returns false.

It really is that simple!