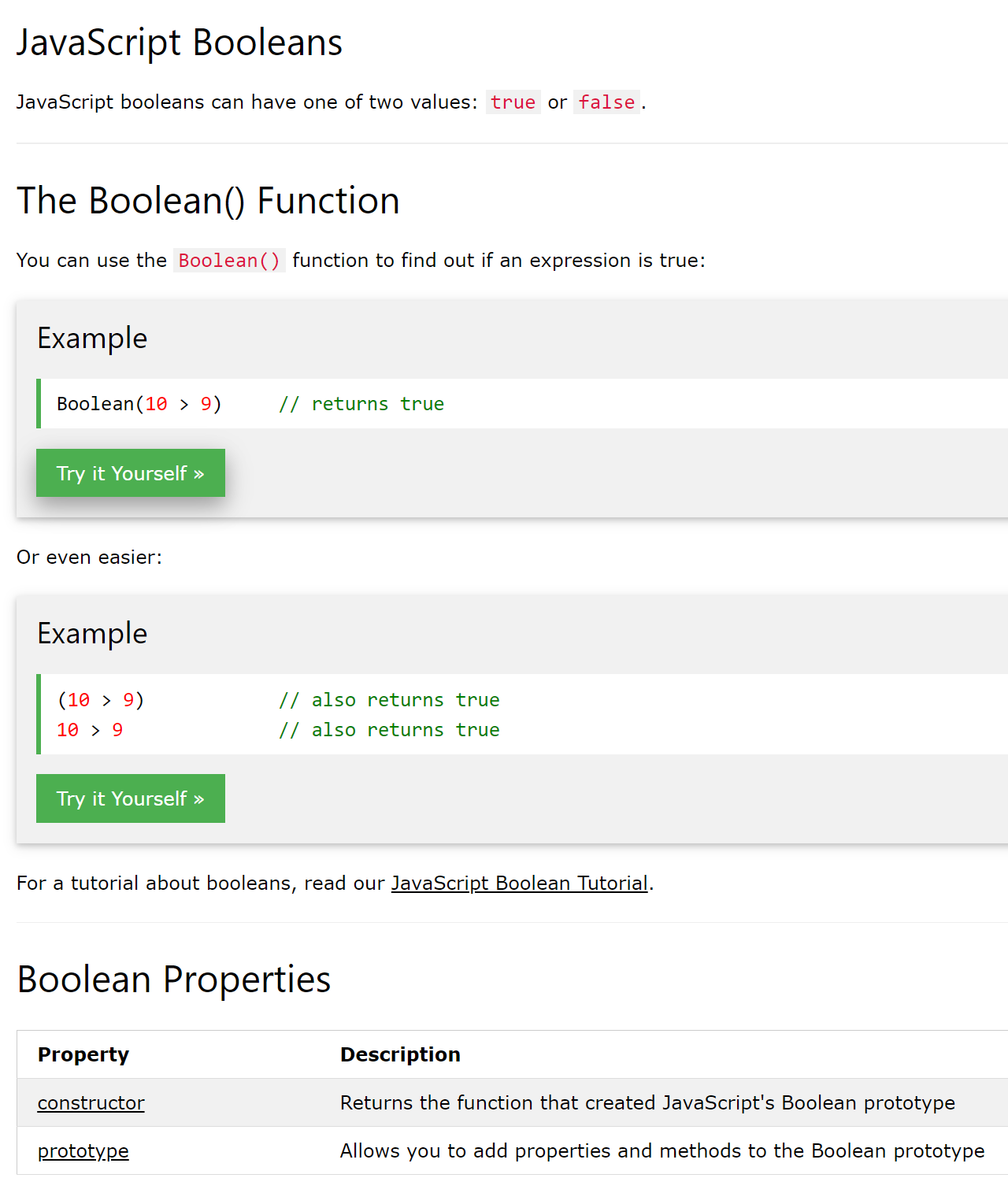
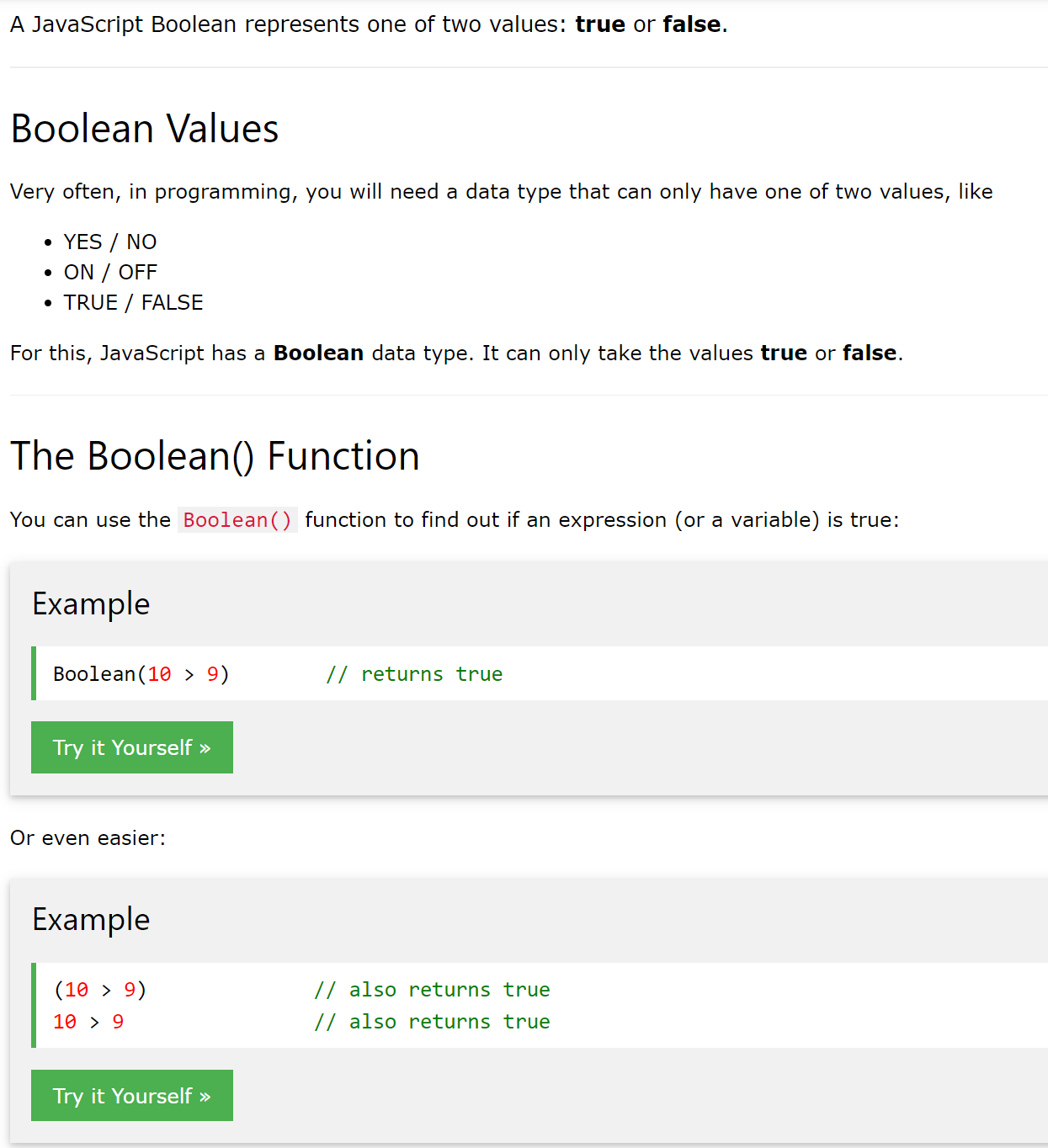
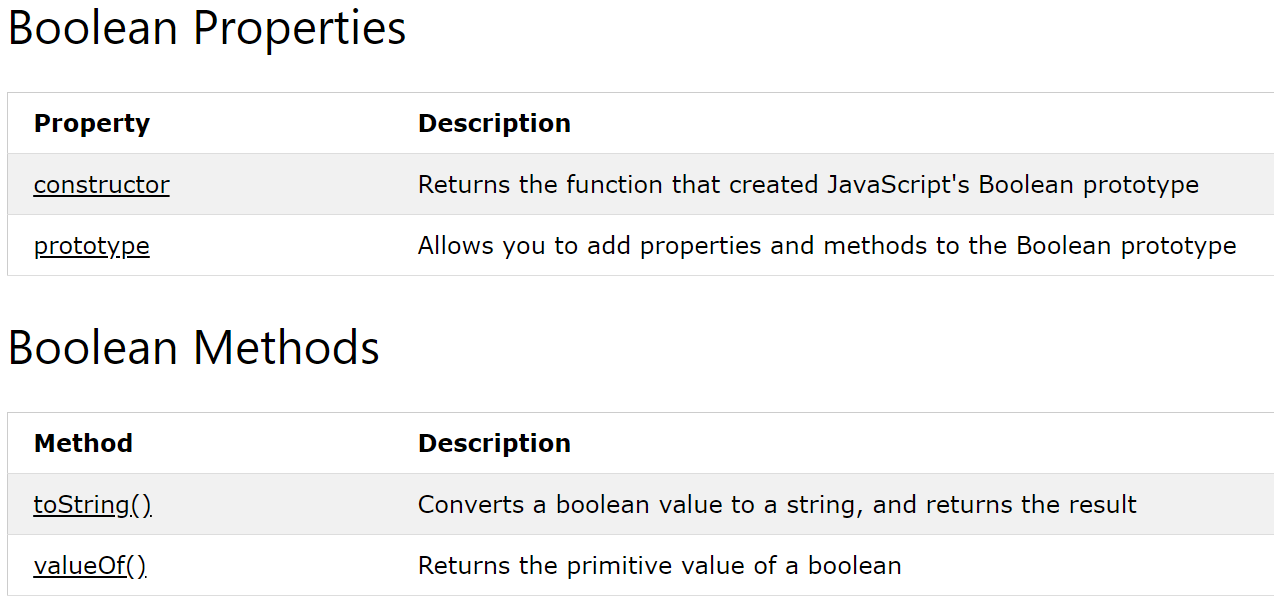
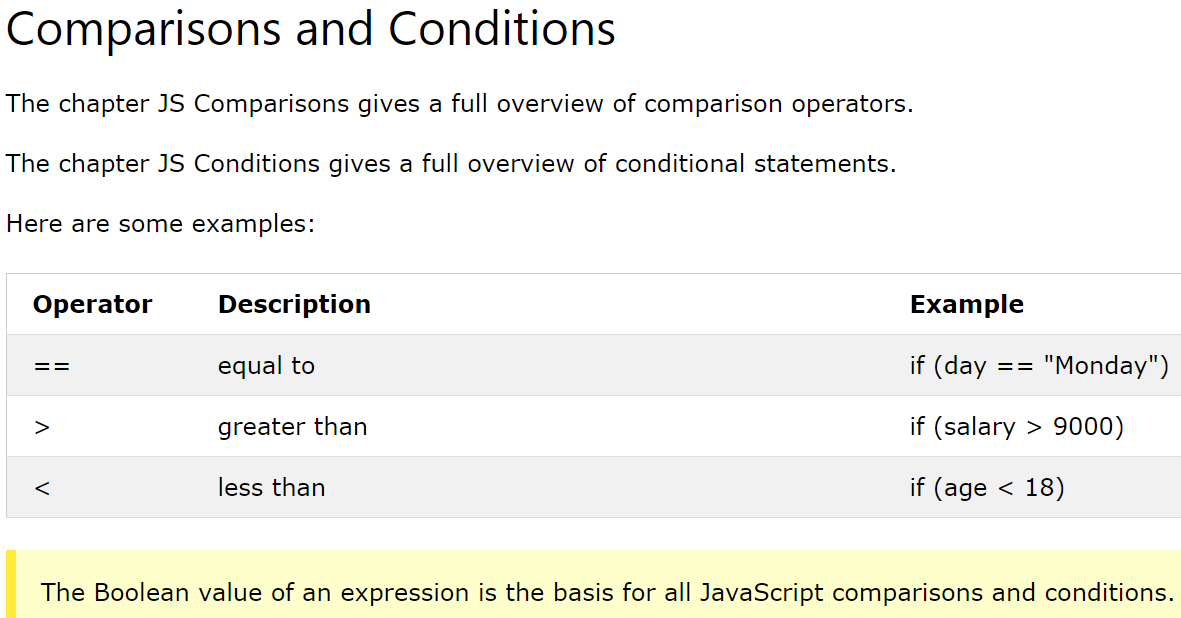
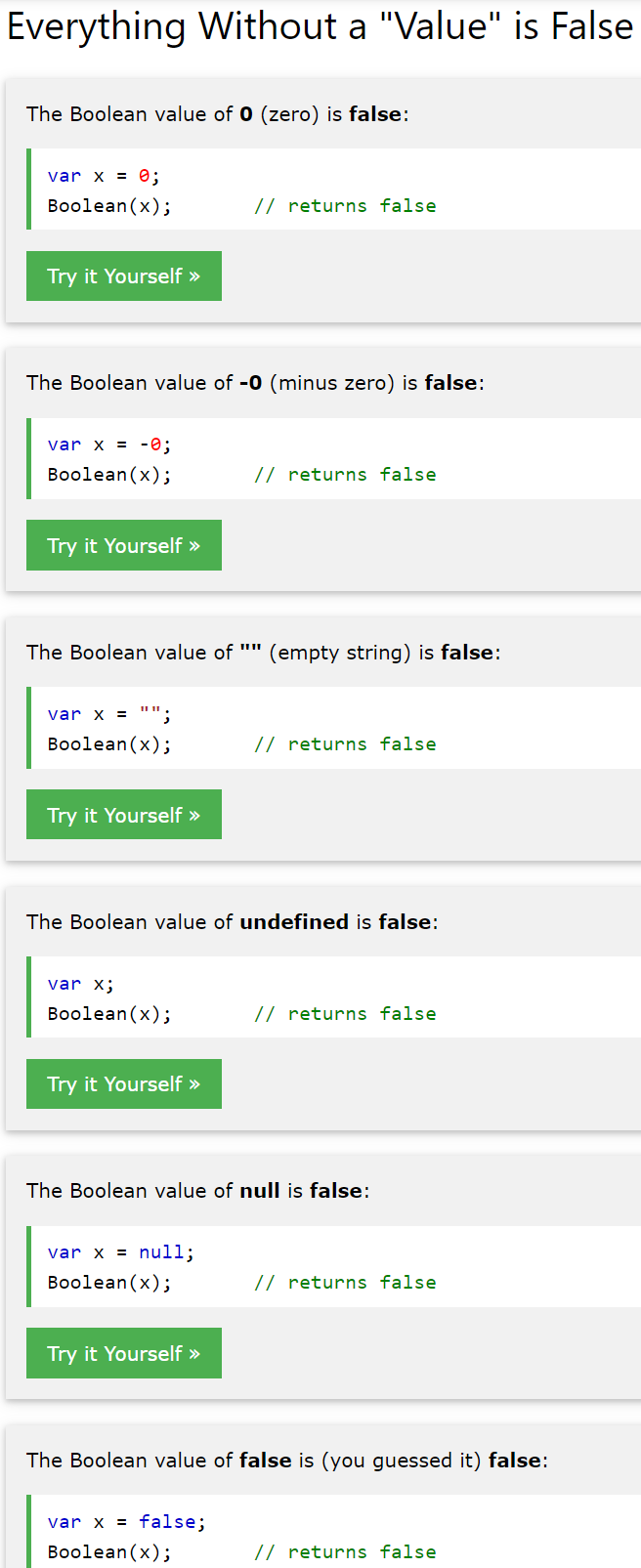
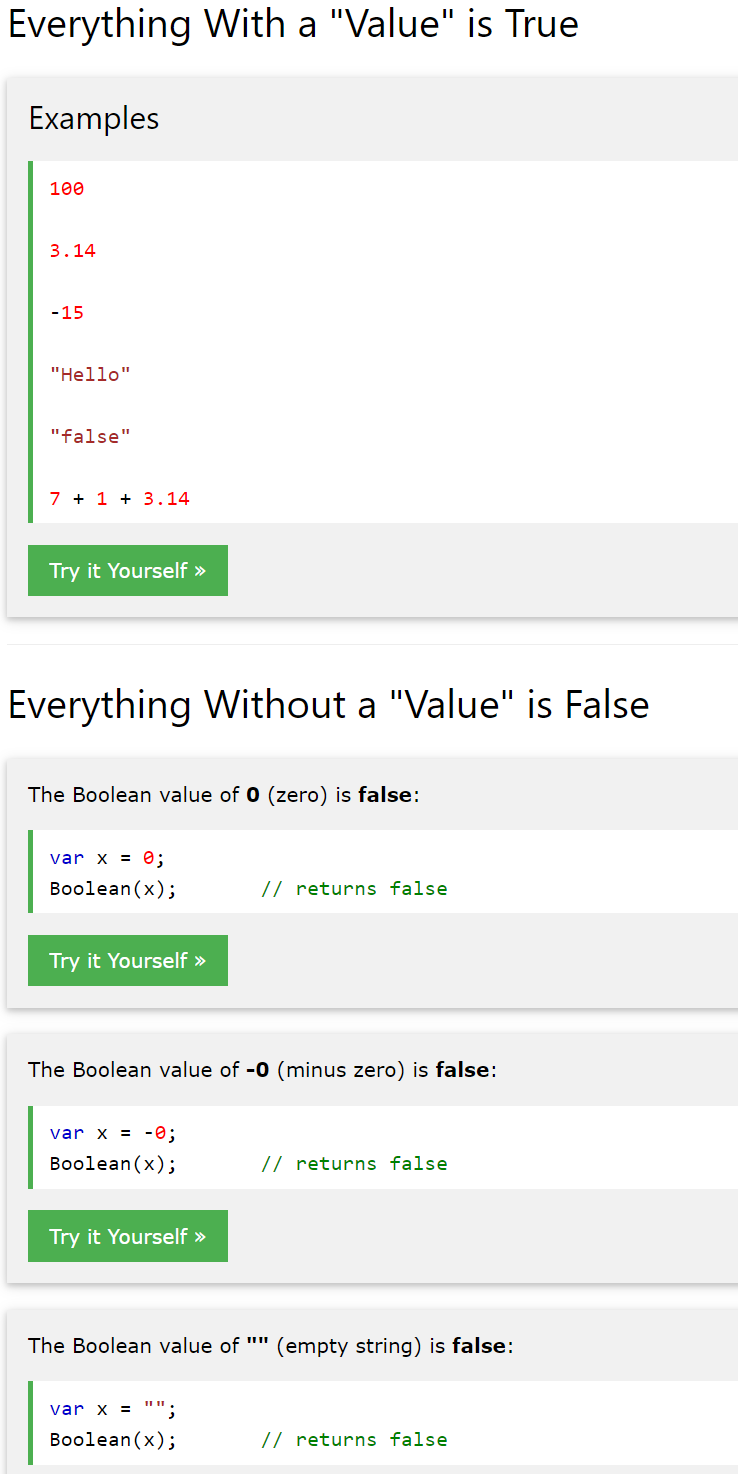
**Boolean Notes:**

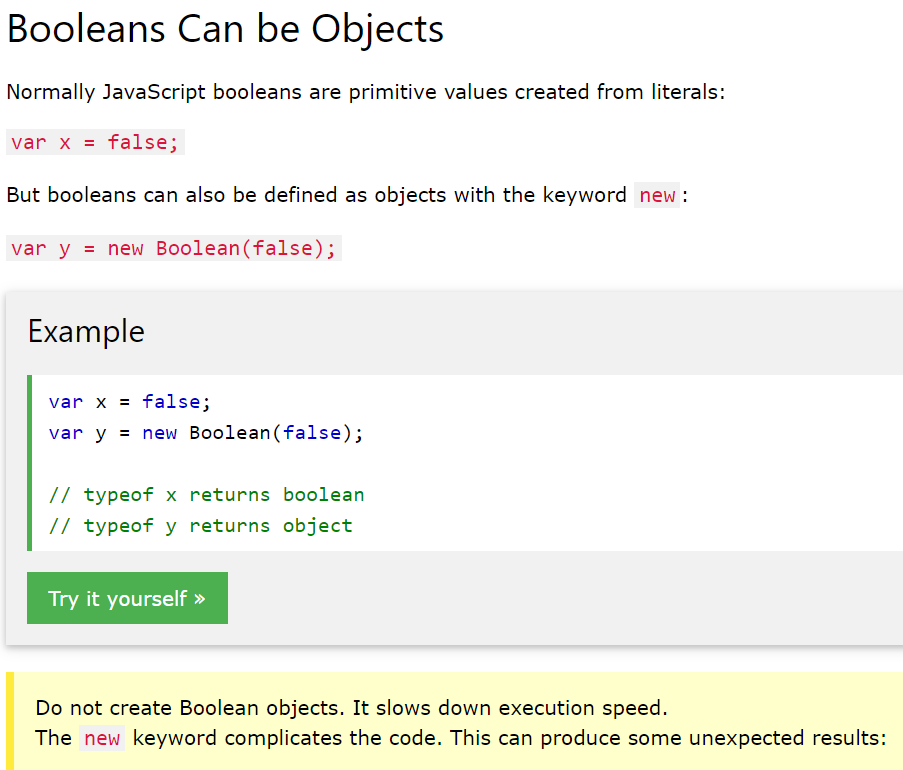


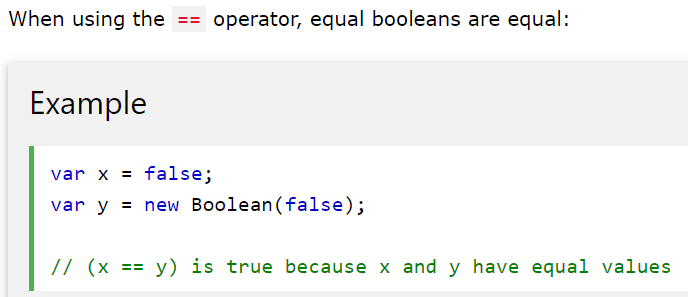


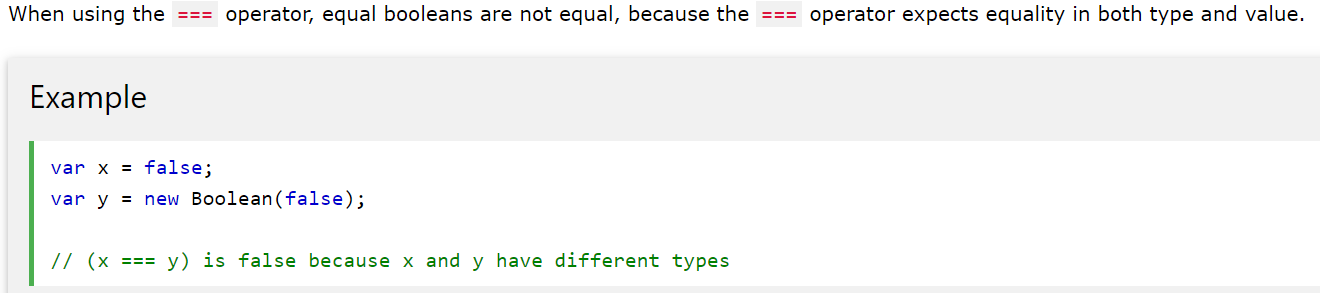


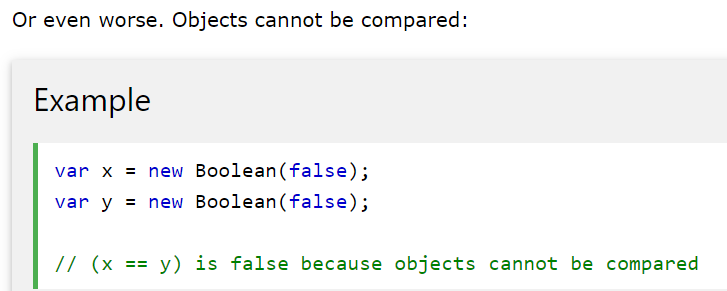












*////////// PROBLEM 7 //////////*

*// Do not edit the code below.*

var **getRandomArbitrary** = function() {

return Math.**floor**(Math.**random**() \* 30);

};

*// Do not edit the code above.*

*/\**

*var numbers = [0,3,4,5,6,7,9,14,17,24,25,26,29,30];*

*Above you're given a function that will return a random number between 0 and 30. There is also a commented out array full of numbers to help you visualize what your function will be receiving.*

*\*\*Write a function named finder that will take in an array as an argument.*

*\*\*It will then get a random number (by invoking getRandomArbitrary).*

*\*\*Loop through the array to see if that random number is in the array.*

*\*\*If it is, return true, if it's not, return false*

*// \*\*Write a function named finder that will take in an array as an argument.*

*// \*\*It will then get a random number (by invoking getRandomArbitrary).*

*//1. Loop through the array*

*//2. see if that random number is in the array.*

*//3. CHECK If the (result = randomArbi....) is (in the array) return true,*

*//4. if it's not,(in the array) return false*

*\*/*

*//NOTES - i.e. code-evaluating questions:*

*//when do we want to exit this function?*

*//what is the purpose of this function?*

*//Do you want to check the entire array for the condition, or only a portion? – Ask this before writing any return or final condition. Write the criteria in your if\else statements based on WHERE or WHEN you want an output or new command to run or exit. EX: If you need to check an entire array, you can use “.length” as part of your final “else” or “conditional block”*

*//Likewise, other built-in browser functions can help you accomplish the “where do I stop and spit the answer I landed on out” in your code.*

*//An iterator can be flexible, and when you run your conditions against it, you can compare the value at the current index of that iterator – or you can also tell the iterator AS AN INDEX to stop AT that index, with another (optionally final) condition.*

*//TIP: Console.log portions of your output that the code is stopping at to compare what the value is and why it might be getting stuck, this will help you debug, simplify, or reconsider code.*

*//Still not 100% clear on "return" - it will exit the function if you are not careful with the way you use it: below the translation for this usage is: "ifStatement#1's condition is met, THEN the function can enter it's block and perform the code (ie returning true)-BUT if it does NOT meet criteria, move to the next block. The next block (elseif-Statement#2) is not accessible to the function until ITS condition is also met. The function must start back at the beginning with the next advancement of the iterator and check the conditions again until the value it is comparing to #1 or #2 meets the required criteria. Run the return statement that is located within the block that becomes accessible first."*

*//Code Here*

const arrs = [0,3,4,5,6,7,9,14,17,24,25,26,29,30];

function **finder**(arrs){

var result = **getRandomArbitrary**();*//result represent the value of a random number invoked*

**console**.**log**('Found '+ result + ' in the array?')

for(i = 0; i < arrs.length; i++){*//Start at index 0, stop repetition once i is no longer less than the last index, run code (check the conditions), and start over at the next index if noether condition returns an output.*

**console**.**log**(result, arrs.length-1===i);

if(result===arrs[i]){*//"RandomNumber" MUST be equal to the VALUE your current array index represents*

return true;*//can only be accessed if above "entry key" condition is met*

}else if(arrs.length-1 === i){*//Will only be tested if first condition block is not made accessible to function. This checks the actual index value of i, rather than the element-value that that index represents.*

return false;*//This code is not accessible to the function until the above condition's criteria is met- i.e. until i is equal to the last index-location in the array.*

}

}*//Boolean(result);? Ugh.... I am returning boolean values... Moving on.*

}

**console**.**log**(**finder**(arrs));