Week 3: Research

1. Select **five methods** that can be used on an Array and describe the following for each:

    1) what the method signature is

    2) what the method does

    3) why would this method be useful (how could you use it)?

1. Array.push()
   -Will allow you to add new elements to the end of an array.
   This method is useful as a simple instance of adding new information to an array either without necessitating where the information is added to the array, or to rearrange the added information with additional methods.

2. Array.splice()
   -Functions as adding or replacing existing elements in the array with new elements, or to just completely remove any existing elements from the array.
   This method gives a bit more functionality than the push method, and can allow you to add elements to the array at specified indexes, or replace existing elements with new ones. It also allows you a method of removing elements from an array based on their index position.

3. Array.concat()
   -Simply merges two or more arrays into a single array.
   The .concat method doesn't change any information in the existing arrays, and can be used to merge multiple arrays. This method also preserves this like empty spaces in arrays as well as data types of elements.

4. Array.sort()
   -Rearranges elements in an array based on parameters set in the method.
   This method allows you to sort information in an array, by default in an ascending order, and doesn't preserve the original order of the elements in the array. Additional functions can be added to further specify how to sort the elements.

5. Array.pop()
   -Deletes the last element in an array and returns that element.
   The .pop method will permanently remove the last element in an array and return the information within that element. This method can be useful if you have a list of items in an array that you want to run through once then have them removed from your existing array.
   [1]

2. What is the difference between == and ===?

The equality operator '==' compares two pieces of data to check if their values are equal to each other and determines a boolean result, but does not check to see if they are of the same data type. The identity operator '===' **does** check two pieces of data for both the same value, as well as data type. For example, 0 == '0' would return the boolean value true, despite one operand being a number, and the other a string. However, using that same comparison with a identity operator, 0 === '0' would return false, despite both values being zero.

[2]