EX1:

Method	Description
Object.create()	static method creates a new object, using an existing object as the prototype of the newly created object.
Object.defineProperties()	static method defines new or modifies existing properties directly on an object, returning the object.
Object.freeze()	static method *freezes* an object. Freezing an object prevents extensions and makes existing properties non-writable and non-configurable. A frozen object can no longer be changed: new properties cannot be added, existing properties cannot be removed, their enumerability, configurability, writability, or value cannot be changed, and the object's prototype cannot be re-assigned. freeze() returns the same object that was passed in.
Object.hasOwn()	static method returns true if the specified object has the indicated property as its *own* property. If the property is inherited, or does not exist, the method returns false.
Object.is()	static method determines whether two values are the same value
Object.seal()	static method *seals* an object. Sealing an object prevents extensions and makes existing properties non-configurable. A sealed object has a fixed set of properties: new properties cannot be added, existing properties cannot be removed, their enumerability and configurability cannot be changed, and its prototype cannot be re-assigned. Values of existing properties can still be changed as long as they are writable. seal() returns the same object that was passed in.
Object.valueOf()	method of Object converts the this value to an object. This method is meant to be overridden by derived objects for custom type conversion logic.
Object.getPrototypeOf()	static method returns the prototype (i.e. the value of the internal [[Prototype]] property) of the specified object.
Object.toLocaleString()	method returns a string representing the object. This method is meant to be overridden by derived objects for locale-specific purposes.
Object.entries()	static method returns an array of a given object's own enumerable string-keyed property key-value pairs.

EX2:

Method	Description
at()	method takes an integer value and returns the item at that index, allowing for positive and negative integers. Negative integers count back from the last item in the array.
copyWithin()	method shallow copies part of an array to another location in the same array and returns it without modifying its length.
entries()	method returns a new **Array Iterator** object that contains the key/value pairs for each index in the array.
every()	method tests whether all elements in the array pass the test implemented by the provided function. It returns a Boolean value.
fill()	method changes all elements in an array to a static value, from a start index (default 0) to an end index (default array.length). It returns the modified array.
findIndex()	method returns the index of the first element in an array that satisfies the provided testing function. If no elements satisfy the testing function, -1 is returned.
flat()	method creates a new array with all sub-array elements concatenated into it recursively up to the specified depth.
flatMap()	method returns a new array formed by applying a given callback function to each element of the array, and then flattening the result by one level. It is identical to a map() followed by a flat() of depth 1 (arr.map(...args).flat()), but slightly more efficient than calling those two methods separately.
forEach()	method executes a provided function once for each array element.
group()	method groups the elements of the calling array according to the string values returned by a provided testing function. The returned object has separate properties for each group, containing arrays with the elements in the group.
map()	method **creates a new array** populated with the results of calling a provided function on every element in the calling array.
reverse()	method reverses an array *in place* and returns the reference to the same array, the first array element now becoming the last, and the last array element becoming the first. In other words, elements order in the array will be turned towards the direction opposite to that previously stated.
values()	method returns a new *array iterator* object that iterates the value of each index in the array.
Array.of()	static method creates a new Array instance from a variable number of arguments, regardless of number or type of the arguments.
reduce()	method executes a user-supplied "reducer" callback function on each element of the array, in order, passing in the return value from the calculation on the preceding element. The final result of running the reducer across all elements of the array is a single value.