Basics on functions

Relevant links

- Flanagan's book, section 4.3, 8.1, 8.2.1, 8.3.1
- MDN's reference for function statements[1]
- MDN's reference for function expressions[2]

Functions in Javascript

- Functions are first class objects in Javascript. While we will explore later what that means in more detail, for now it means that functions can be defined anywhere in the code, even within other functions, or in the value field of an object property, and so on.
- The basic syntax for a function definition is: `function f(arg1, arg2) { ... }`.
- You can often have anonymous functions: `function(args) { ... }`. These can be stored as local variables via a `let` or `const` statement.
- Function declarations employ a "hoisting" system: They behave as if they appeared at the very top of the current block. In practice this means that functions can be used *before* the code that defines them, as long as it is done within the same block.
- Your function must use explicit `return` statements if it wants to return a result.
- The function stops execution the moment a `return` statement is encountered. Any lines following that `return` statement will be ignored.
- There are various ways of calling functions with some subtle semantics that we will discuss later. But for now, the name of the function followed by values for the arguments will suffice: `f(val1, val2)`
- The number of values provided when a function is called may differ from the number of arguments in its definition, and that is OK.
- Any extra values passed can be accessed via the `arguments` object, which we may touch on at a later time.
- If there are too few values passed, the remaining arguments will default to the value `undefined`.
- Here is an example of a function that each time it is called with an argument `n` returns a new array of `n` random values:

[1] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/function
[2] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/function

```
function makeRandomArray(n) {
    let newArray = [];

    for (let i = 0; i < n; i += 1) {
        newArray[i] = Math.random();
    }

    return newArray;
}
makeRandomArray(4);
makeRandomArray(2);
```

Practice Functions

1. Write a function indexOf that is given an array and an element, and it searches through the array for that element (using == to check equality). If it finds the element then it returns the index where it found it. Otherwise it returns -1. Do NOT use the built-in array function that does this, just do it via a for-loop.
2. Write a function isOrdered that takes as argument an array of numbers and returns true if the array elements are in ascending order, false otherwise. Do NOT change the elements in the array, and do NOT use any sort method.
3. Write a function isOrdered2 that behaves like isOrdered except that it takes a second optional argument, called asc. If that argument is false, then it should check to see that the array is ordered in *descending* order, in all other cases it should check for *ascending* order.
4. Write a function takeNames that takes as input an array of objects. It then goes through the array, and for any object that has a property called "name" it takes that value and puts it in a new array. It then returns that array.
5. Write a function takeNames2 which takes a second optional argument. If that argument is specified and is a string, then it uses that string as the key/property it looks for, instead of using "name".