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 `var` statement.
- Function declarations use the same "variable hoisting" system we discussed in the variable section[3]. In practice this means that functions can be used *before* the code that defines them.
- Unlike some other languages, the value of the last statement is not used as the return value of the function. 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 two few values passed, the remaining arguments will default to the value `undefined`.

[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
[3] local_vs_global.html

- 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:

```
function makeRandomArray(n) {
    var newArray = [];
    var i;
    for (i = 0; i < n; i += 1) {
        newArray[i] = Math.random();
    }
    return newArray;
}
makeRandomArray(4);
makeRandomArray(2);
```