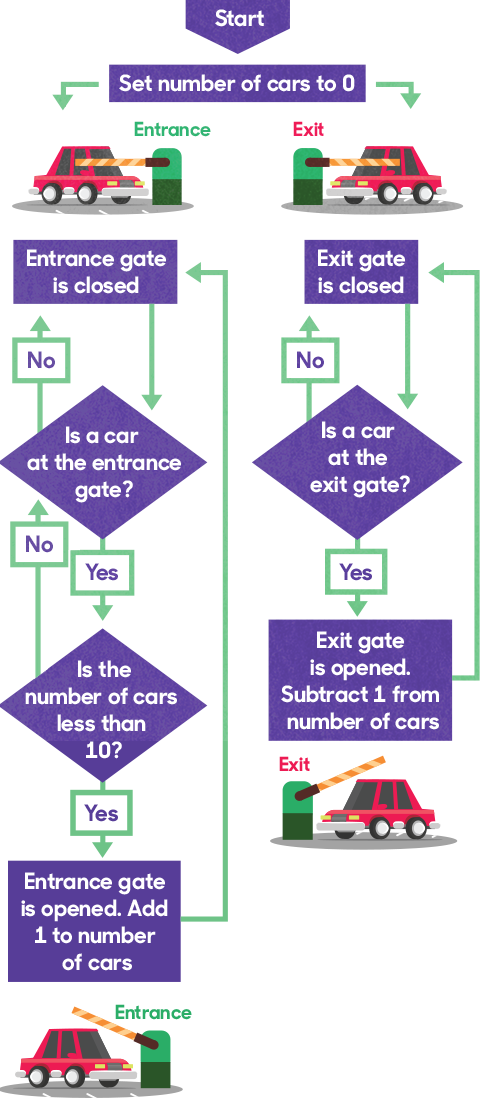
Algorithms

* Nicklaus Wirth wrote a computer programming textbook titled Algorithms + Data Structures = Programs (Prentice-Hall).
* An algorithm (pronounced AL-go-rith-um) is a procedure or formula for solving a problem.
* An algorithm usually means a small procedure that solves a recurrent problem.
* An algorithm is a sequence of instructions or a set of rules that are followed to complete a task. This task can be anything, so long as you can give clear instructions for it.
* Computers won’t understand your algorithm as they use a different language.
* It will need to be translated into code which the computer will then follow to complete a task.
* This code is written in a programming language. There are many different types of programming languages.
* Some that you may come across are Logo, Scratch, Blockly, Python and Kodu. Each of these languages are suited to different things.
* Example of Car park algorithm



* Any computer program that goes beyond the trivial “Hello, world!” will usually require some type of structure to manage the data the program is written to manipulate, along with one or more algorithms for translating the data from its input form to its output form.
* Arrays are great for some problems, but for many complex problems, they are simply not sophisticated enough. Most experienced programmers will admit that for many programming problems, once they come up with the proper data structure, the algorithms needed to solve the problem are easier to design and implement.
* Studying algorithms is important because there is always more than one algorithm that can be used to solve a problem, and knowing which ones are the most efficient is important for the productive programmer. For example, there are at least six or seven ways to sort a list of data, but knowing that the Quicksort algorithm is more efficient than the selection sort algorithm will lead to a much more efficient sorting process. Or that it’s fairly easy to implement a sequential or linear search algorithm for a list of data, but knowing that the binary sort algorithm can sometimes be twice as efficient as the sequential search will lead to a better program.
* The comprehensive study of data structures and algorithms teaches you not only which data structures and which algorithms are the most efficient, but you also learn how to decide which data structures and which algorithms are the most appropriate for the problem at hand.
* The this keyword is used to tie each function and property to an object instance.

Decision Construct

* Decision constructs allow our programs to make decisions on what programming statements to execute based on a Boolean expression.

Loop:

While Loop

* When we want to execute a set of statements while a condition is true.

For Loop

* When we want to execute a set of statements a specified number of times, we use a for loop

Functions

* JavaScript provides the means to define both value-returning functions and functions that don’t return values (sometimes called sub procedures or void functions).
* All function parameters in JavaScript are passed by value, and there are no reference parameters.

Variable Scope

The scope of a variable refers to where in a program a variable’s value can be accessed. The scope of a variable in JavaScript is defined as function scope. This means that a variable’s value is visible within the function definition where the variable is declared and defined and within any functions that are nested within that function.

When a variable is defined outside of a function, in the main program, the variable is said to have global scope, which means its value can be accessed by any part of a program, including functions.

Recursive Function

* When a function is called recursively, the results of the function’s computation are temporarily suspended while the recursion is in progress.

This

* The this keyword is used to tie each function and property to an object instance.

Object

* An object is a container of properties, where a property has a name and a value.
* A property name can be any string, including the empty string.
* A property value can be any JavaScript value except for undefined.
* Objects are useful for collecting and organizing data.
* Objects can contain other objects, so they can easily represent tree or graph structures.
* JavaScript includes a prototype linkage feature that allows one object to inherit the properties of another. When used well, this can reduce object initialization time and memory consumption.

Object Types

Object Literals

* Object literals provide a very convenient notation for creating new object values.
* An object literal is a pair of curly braces surrounding zero or more name/value pairs.

var empty\_object = {};

var stooge = {

"first-name": "Jerome",

"last-name": "Howard"

};

* Objects can nest:

var flight = {

airline: "Oceanic",

number: 815,

departure: {

IATA: "SYD",

time: "2004-09-22 14:55",

city: "Sydney"

},

arrival: {

IATA: "LAX",

time: "2004-09-23 10:42",

city: "Los Angeles"

}

};

Retrieval

* Values can be retrieved from an object by wrapping a string expression in a [ ] suffix. If the string expression is a constant, and if it is a legal JavaScript name and not a reserved word, then the dot (.) notation can be used instead. The dot (.) notation is preferred because it is more compact and it reads better

stooge["first-name"] // "Joe"

flight.departure.IATA // "SYD"

* The || operator can be used to fill in default values:

var middle = stooge["middle-name"] || "(none)";

var status = flight.status || "unknown";

Object in JS

* An object is a collection of properties, and a property is an association between a name (or key) and a value.
* A property's value can be a function, in which case the property is known as a method.
* A property of an object can be explained as a variable that is attached to the object.
* Object properties are basically the same as ordinary JavaScript variables, except for the attachment to objects.
* You access the properties of an object with a simple dot-notation

objectName.propertyName

Object Initializer

* An object initializer is a list of zero or more pairs of property names and associated values of an object, enclosed in curly braces ({}).
* The advantage of the literal or initializer notation is, that you are able to quickly create objects with properties inside the curly braces.

Constructor Function

* Alternatively, you can first create a constructor function and then instantiate an object using that function and the new operator.

Using a constructor function

* Alternatively, you can create an object with these two steps:
* Define the object type by writing a constructor function. There is a strong convention, with good reason, to use a capital initial letter.
* Create an instance of the object with new.

Object Property

* An object can have a property that is itself another object.

Duplicate property names

* When using the same name for your properties, the second property will overwrite the first.

Object.create()