# English teachers,
# if you really want to help your students,
# teach them how to code!

- *"But coding is for math whizzes!"*

**Wrong.** In coding, we talk about language, syntax, even grammar, but never about derivatives or integrals.

- *"Coding is complicated and takes a long time to learn."*

In general, this is also **incorrect**. It all depends on what method is used. We'll come back to this later.

- *"Anyway, what can coding do for my students?"*

For excellent students, not all that much. But for all the rest, a great deal.

Coding is everywhere – as an example, let's take the rules of the road. If drivers were to understand street signs to mean different things, it would lead to misunderstanding and accidents.

It's the same thing with communication. Some words serve to structure a message. If you and your students assign different meanings to words, this will also lead to misunderstanding and accidents. The concept of coding is illustrated in the diagram below.


In this example, the parts of the sentence have three different roles;

1. "the baker slices the bread" is a simple action;
2. "the customer requests it" is a condition;
3. "If" is a *control structure* that subordinates the action to the condition. It is a variation on an "If–then–else" statement, as are words such as "when," "in case," etc.

This example may seem simple, but this is no guarantee that all your students will fully understand it. Every "word map" calls up some type of mental diagram, a mental framework that connects the various parts of the perceived message. Whenever a student misinterprets a word map, this represents a new challenge – and these types of words, both oral and written, are indeed omnipresent in our messages.

Students facing difficulties are usually sent to remedial classes where teachers tend to double down on the examples, exercises, etc., all the while remaining at the level of abstract communication. But the student's problem is in the decoding of abstraction, i.e., understanding the meaning assigned to the words. Hammering away at an abstract message is like continuing to pedal when a bicycle slips out of gear.

But what about drawing a connection: Abstract ⇔ Concrete? I.e., how about creating situations in which students' thoughts are put into specific terms, clearly showing them whether or not their reasoning is correct and why? **YES**, particularly using robotics. In robotics, students design programs and launch robots that execute code exactly as it is received, either successfully carrying out a task or stubbornly reproducing the same error until it is corrected. Students are thus steered toward reshaping their **mental framework** until they manage to make the robot correctly execute the task.

You may say that this is all well and good for drawing squares, circles, even rosettes, but that it has no bearing on the task of mastering natural language. Indeed! This is where "**virtual robotics**" comes in. In virtual robotics, there are varied "universes" suited to the mastery of three types of word maps, namely:

1. **Conditional** structures, such as that found in the above example;
2. The **logical operators** "and," and "or" and their opposites;
3. **Repetitive** structures and their conditions for exiting.

These basic structures enable students to sort through their mental frameworks to find a sizeable number of word maps, or at least enough so that any remedial classes, if still necessary, are in fact effective.

Clearly, prevention is preferable to remediation, and the practice of coding is desirable for all students beginning at age 10 or 11. Everyone seems convinced that coding represents "The Future of the Nation," even if there are many reasons for believing so. In any case, if it's "good for a lot of things" then there's no reason not to just forge ahead with it.

So, to be specific, how should we do this? Let's review the multitude of new methods and their strengths and weaknesses just waiting to be discovered upon opening their shiny packages. The method being proposed here is the culmination of over 50 years of combined experience of two educators who taught hundreds of pupils, students and adults. The method is interactive, can be used alone or in groups, and is accessible online anonymously, free of charge.

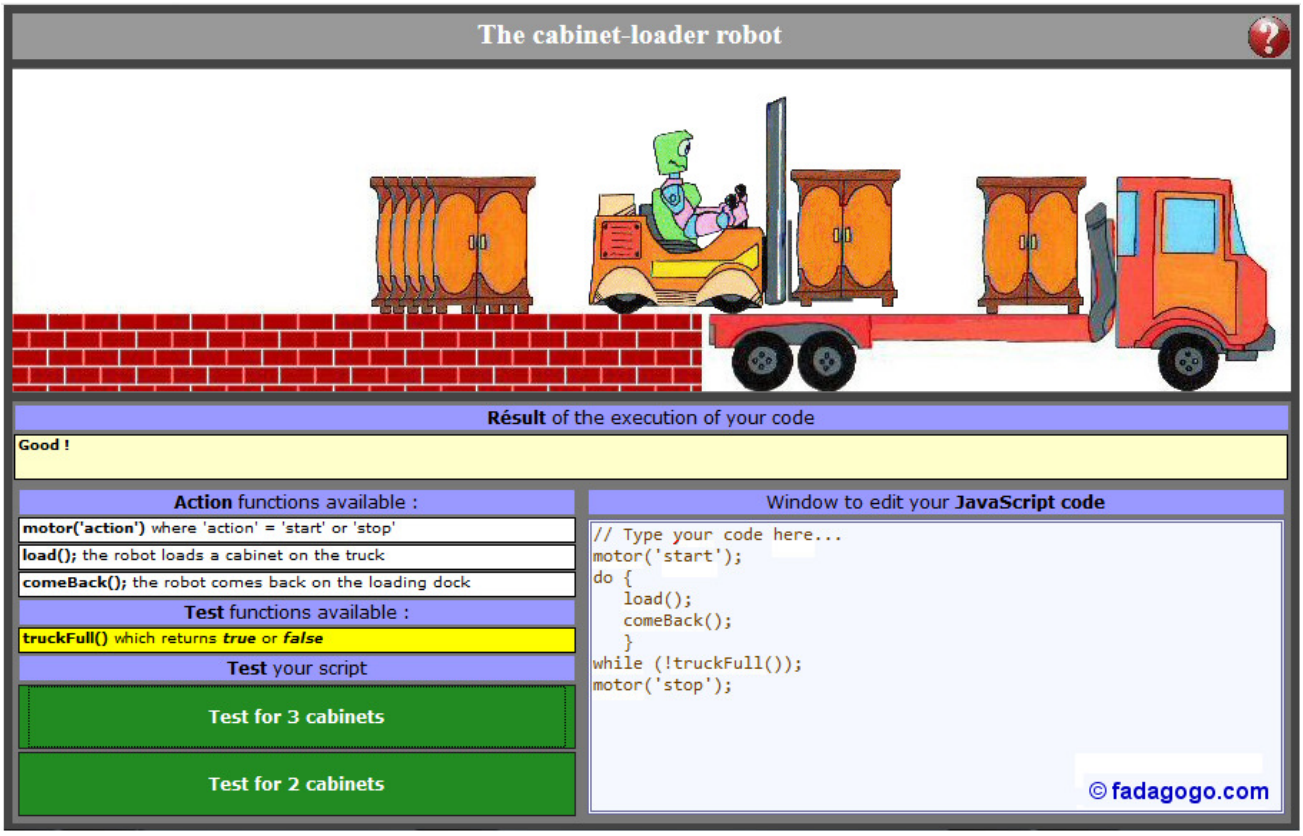
The first step, which is essential and most interesting, is mastering algorithms, i.e., learning the art of organizing "things" in order to achieve a desired result. This step is of great importance in reorganizing students' message structures. This is particularly, although not exclusively, useful for students having difficulties. Algorithms will help develop rigorous thinking, logic, abstraction, analysis and prediction skills for all students. Anyone who works with algorithms will see significant improvements in their thinking, self-expression, reading comprehension and learning in all subjects in which rules must be applied (grammar, math, science, etc.), since these rules are themselves a form of **algorithm**. Significant progress has been observed in these areas, particularly with hearing-impaired children.

Students must first understand which parts of a message to either work around or deal with directly (conditional structures), which parts must be repeated (repetitive structures), and what "and," "or" and their opposites (logical operators) mean. When the method used is effective, coding makes it possible for students with a weak grasp of abstraction to achieve these steps.

Using the method recommended here, the three above structures are approached through 25 progressive exercises presented as challenges. These 25 challenges are grouped into three chapters, each organized as follows:

1. A brief introduction to theory;
2. Questions on these notions of theory, with immediate correction;
3. Challenges corresponding to each chapter.

Each challenge involves a virtual robot which students must steer around the screen.




A detailed description of the robot's abilities and the tasks to be carried out are provided, thus defining the "operating universe." This description makes it possible to take advantage of successes as well as errors, which then serve as counterexamples. The error messages are carefully worded so as to best enable students to work independently. The teacher is thus freed from lecturing and is able to play the role of consultant by focusing attention on each student's progress and the difficulties they may be having. Further information on the benefits of this method are available in "**The Teaching Corner**" of the site. A helpful **teacher's guide** provides all the advice needed to monitor your students, along with detailed solutions to each challenge.

So now, please just give it a whirl and decide for yourself the value of this approach. And we would love to hear about **your adventure as a coder.** Go to our website at:

http://fadagogo.com

and click on "**Introductions au codage**" → Introduction to coding with JavaScript

Plan on spending 5 to 6 hours mastering algorithms, a key topic for your students.

This learning period can optionally be extended for students motivated to pursue variable management, pseudocode programming and introduction to a language. This advanced material is designed for independent study, so there is no need for you to delve into it unless you have a particular interest.

Please feel free to contact us at info@fadagogo.com with any questions, or to provide feedback about your personal experience or that of your students – we are very interested in hearing from you.

We hope you enjoy trying out this coding method, and discover that a field you once thought was too technical is very accessible indeed.

Rupert Meurice de Dormale
info@fadagogo.com