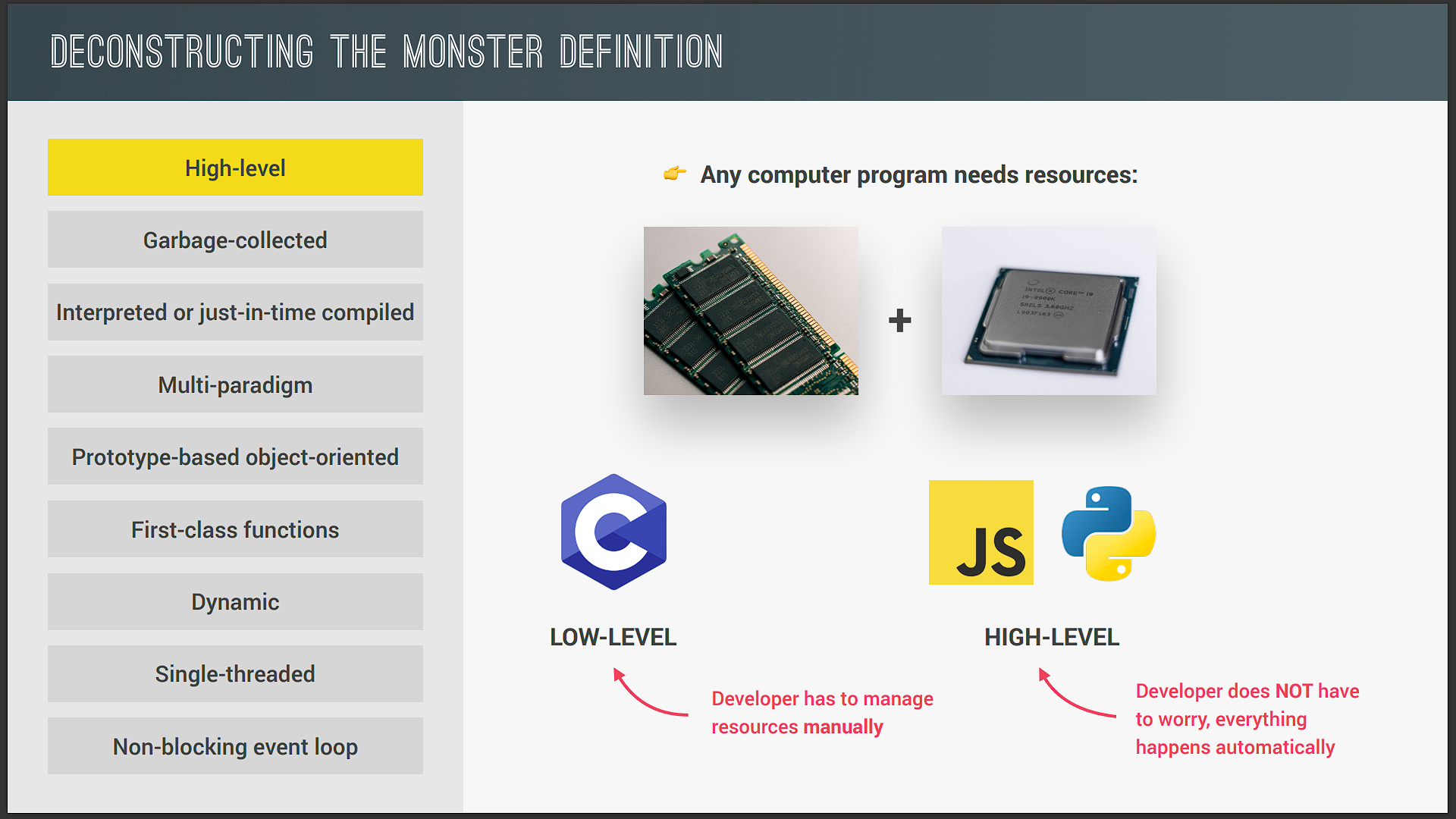
Q1. What is a Javascript Language?

JavaScript is a high-level, object-oriented, multiparadigm programming language.

JavaScript is a high-level, prototype-based, object-oriented, multiparadigm, interpreted, or just-in-time compiled, dynamic, single-threaded, garbage-collected programming language with first-class functions and a non-blocking event loop concurrency model.

High-Level:

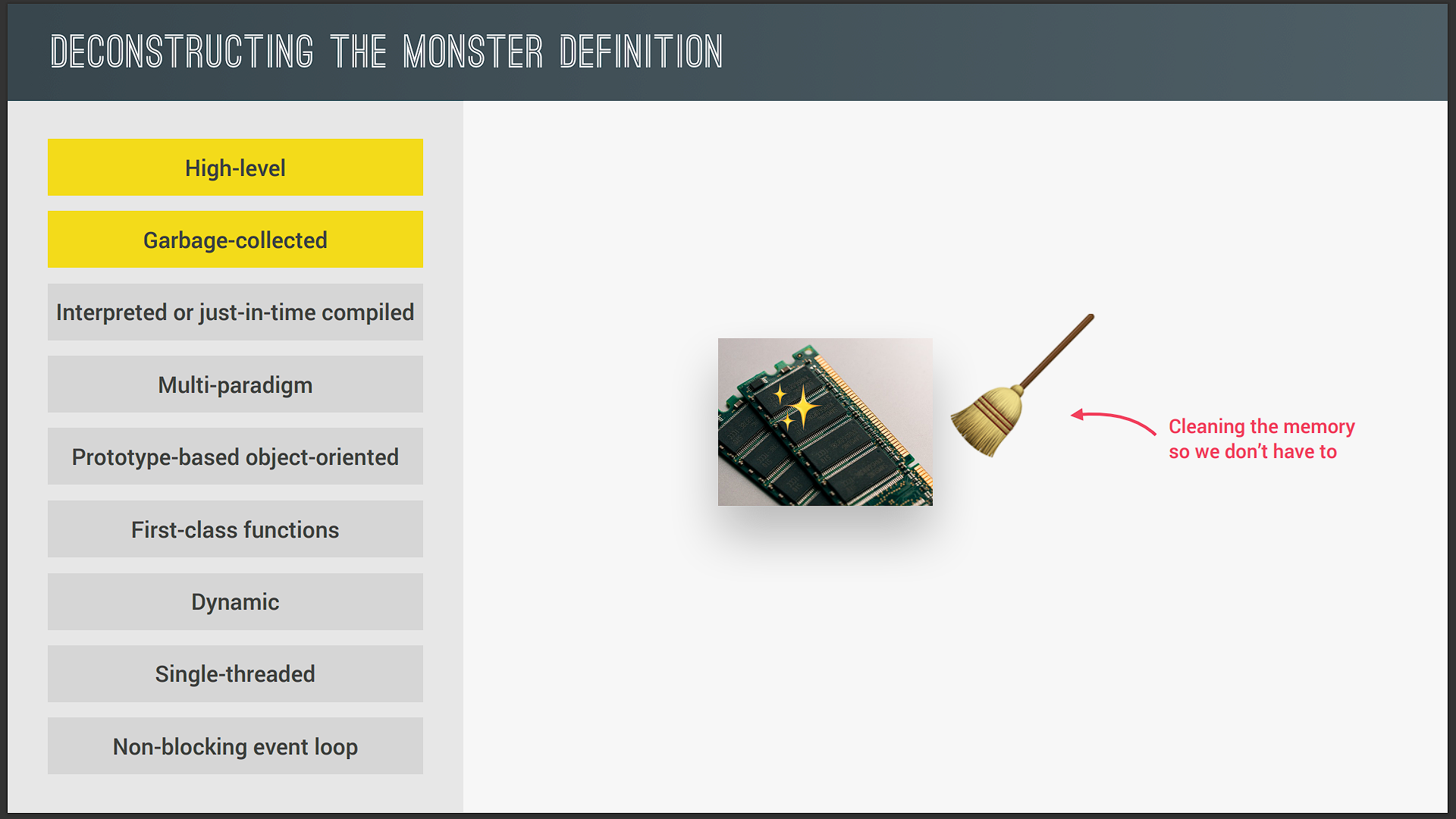


as you might already know, every program that runs on your computer needs some hardware resources, such as memory and the CPU to do its work.

Now, there are low-level languages such as C, where you have to manually manage these resources, for example, asking the computer for memory to create a new variable. On the other side, you have high-level languages such as JavaScript and Python, where we do not have to manage resources at all because these languages have so-called abstractions that take all of that work away from us.

This makes the language easier to learn and to use, but the downside is that programs will never be as fast or as optimized as, for example, C programs.

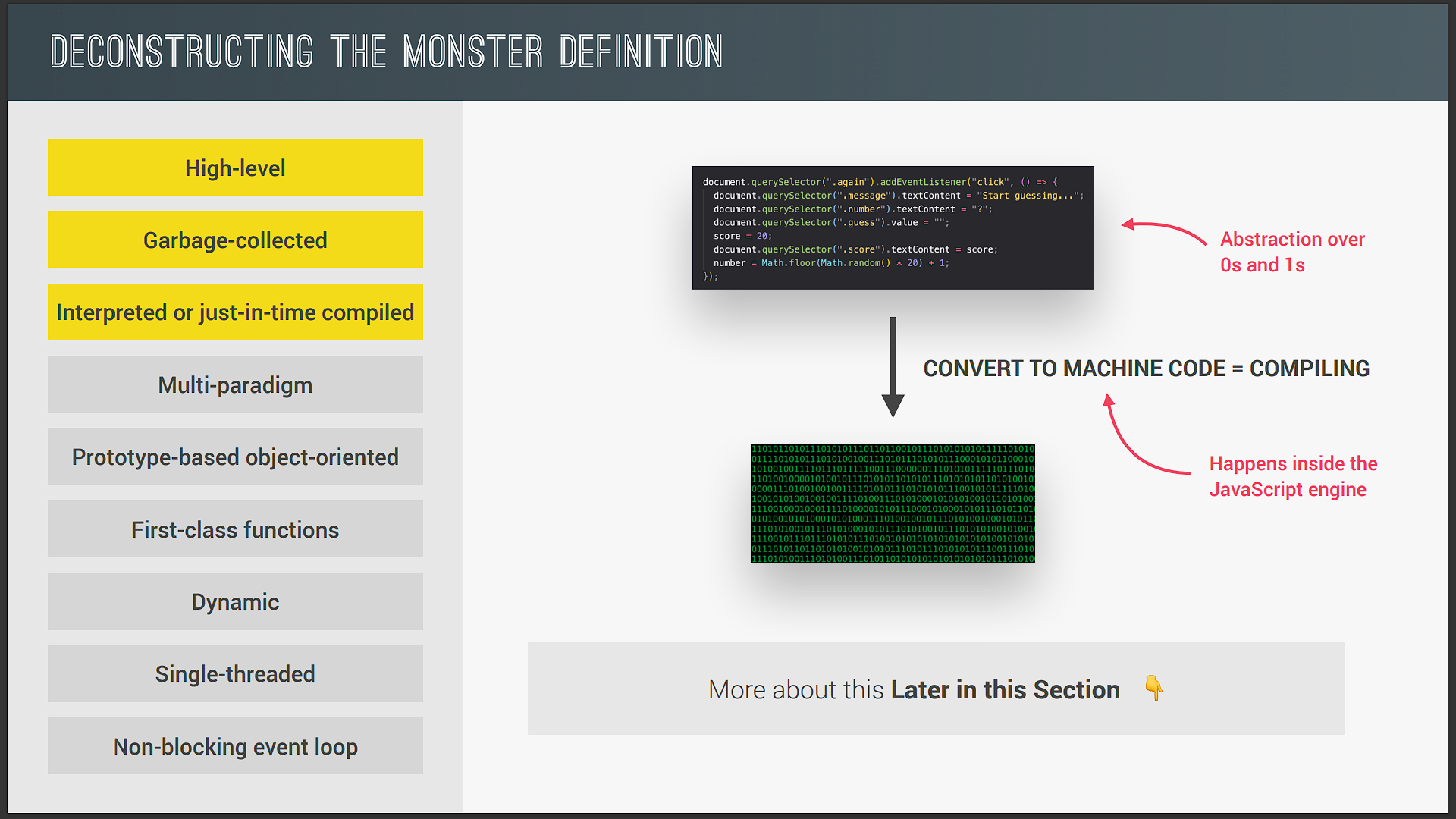
Garbage Collection:-



Now, one of the powerful tools that takes memory management away from us developers is garbage collection, which is basically an algorithm inside the JavaScript engine that automatically removes old unused objects from the computer memory in order not to clog it up with unnecessary stuff.

So it's a little bit like JavaScript has a cleaning guy who cleans our memory from time to time so that we don't have to do it manually in our code.

Interpreted or just-in-time compiled:-

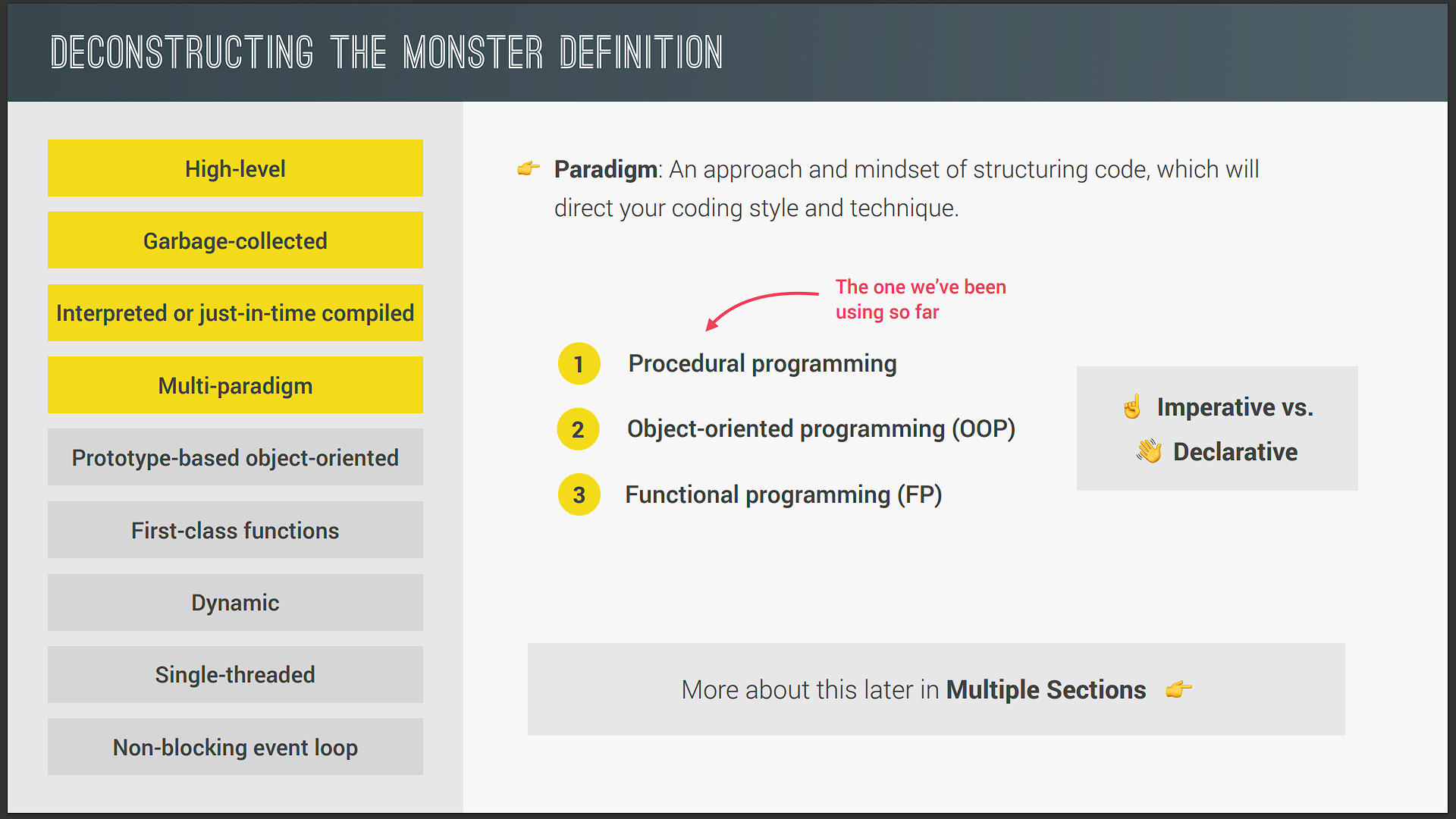


the computer's processor only understands zeros and ones, that's right. Ultimately, every single program needs to be written in zeros and ones, which is also called machine code.

And since that's not really practical to write, is it? We simply write human-readable JavaScript code, which is an abstraction over machine code, but this code eventually needs to be translated to machine code. And that step can be either compiling or interpreting.

This step is necessary in every single programming language because no one writes machine code manually. In the case of JavaScript, this happens inside the JavaScript engine.

Mutli-paradigm:-



Now, one of the things that makes JavaScript so popular is the fact that it's a multiparadigm language. In programming, a paradigm is an approach and an overall mindset of structuring our code, which will ultimately direct the coding style and technique in a project that uses a certain paradigm.

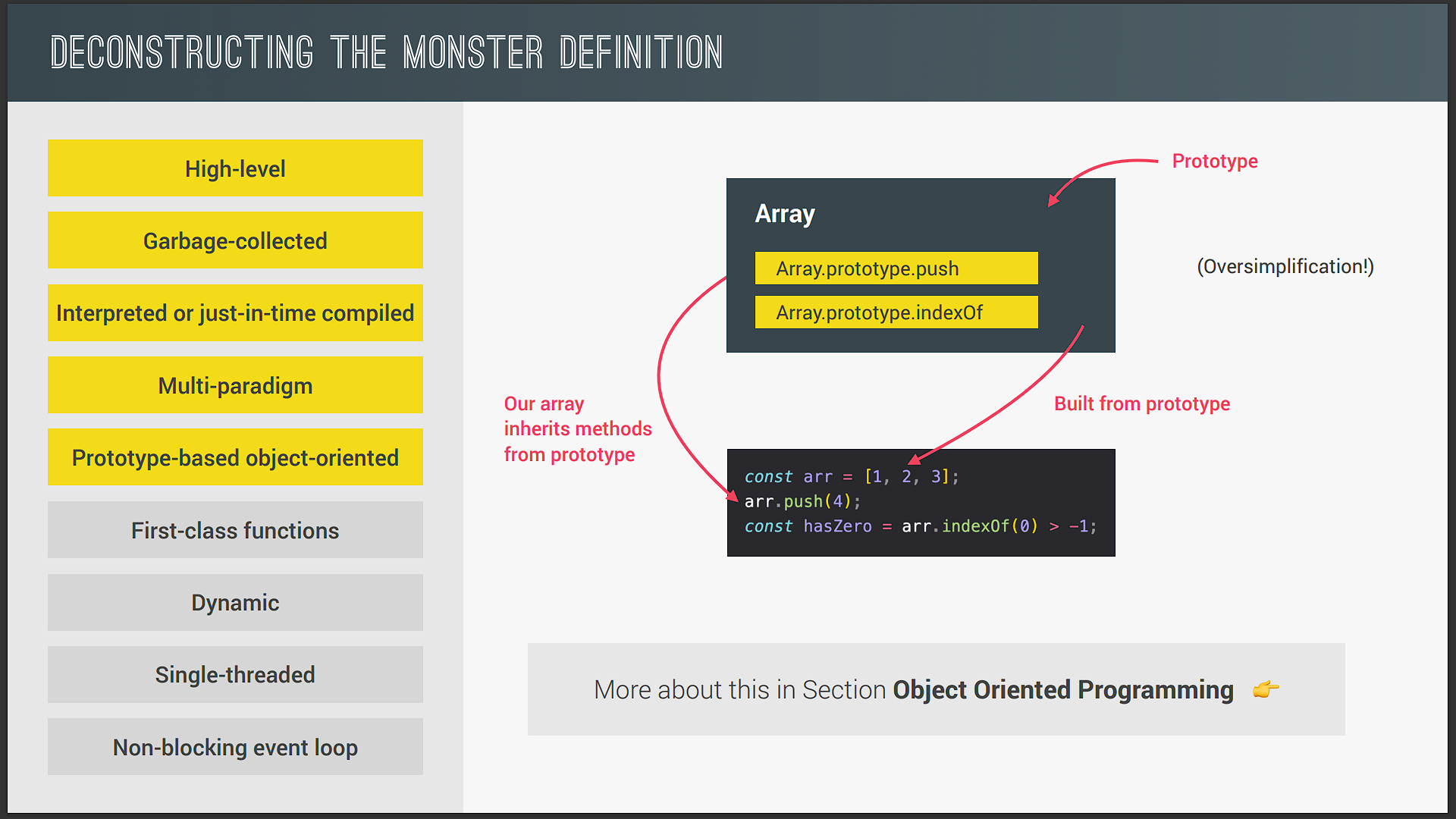
And this definition still sounds kind of abstract, but don't worry, it will become more clear as we move on.

Now, three popular paradigms are procedural, object-oriented, and functional programming. So procedural programming is what we've been doing so far, which is basically just organizing the code in a very linear way and then with some functions in between.

Now, about object-oriented programming and functional programming, I will talk about them in a second. Also, we can classify paradigms as imperative or as declarative, but again, more on that later.

Now, many languages are only procedural or only object-oriented or only functional, but JavaScript does all of it. So it's really flexible and versatile, and so we can do really whatever we want with it. It's our choice. We can use whatever paradigm we want, and in this course, I will show you how to use all of them.

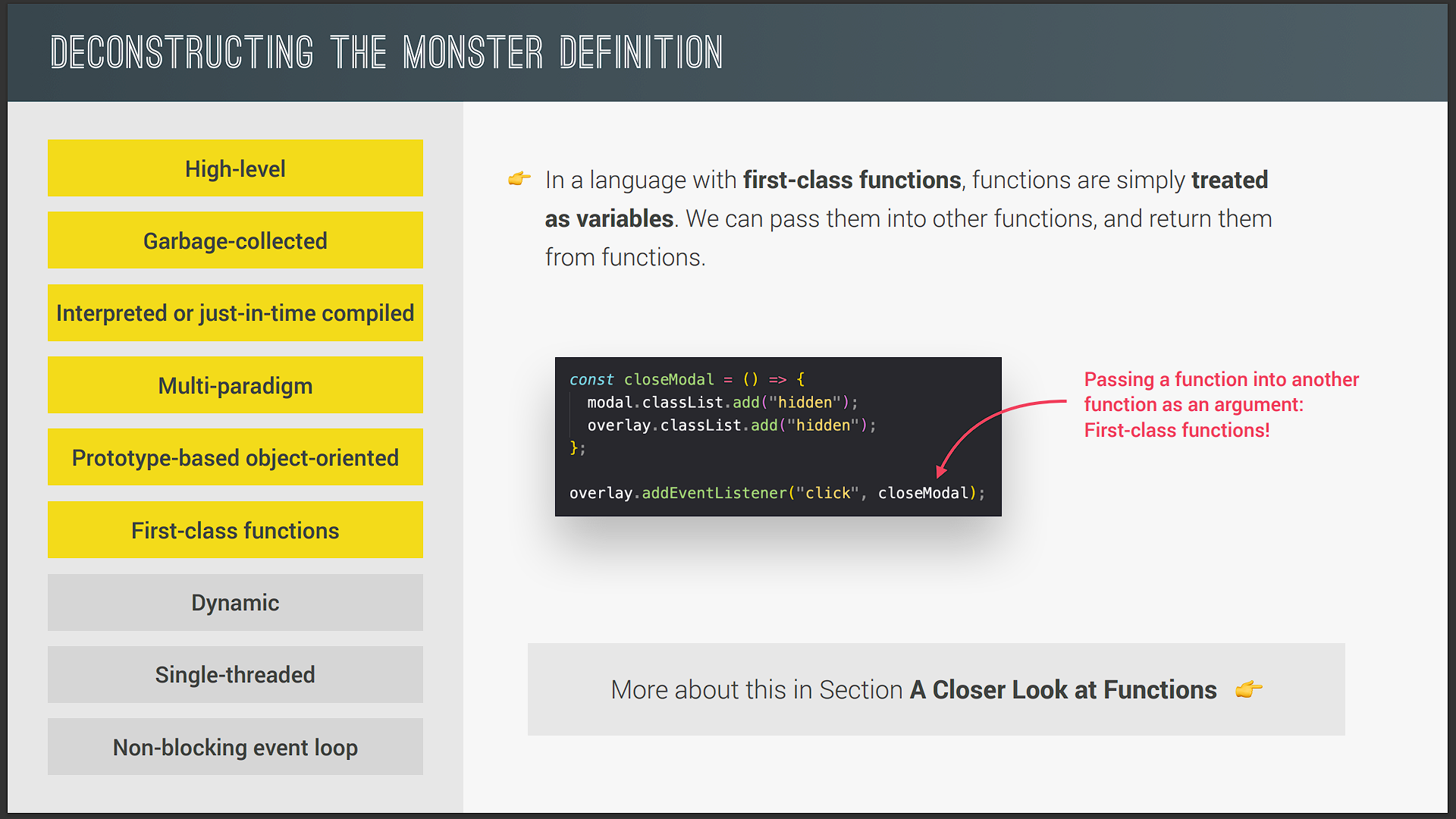
Prototype-based object-oriented:-



So, about the object-oriented nature of JavaScript, it is a prototype-based object-oriented approach. Now, what does that mean? Well, first, almost everything in JavaScript is an object, except for primitive values such as numbers, strings, etc. But arrays, for example, are just objects. And we already saw that in practice, right?

Now, have you ever wondered why we can create an array and then use the push method on it, for example? Well, it's because of prototypal inheritance. Basically, we create arrays from an array blueprint, which is like a template, and this is called the prototype. This prototype contains all the array methods, and the arrays that we create in our code then inherit the methods from the blueprint so that we can use them on the arrays.

First-class function:-

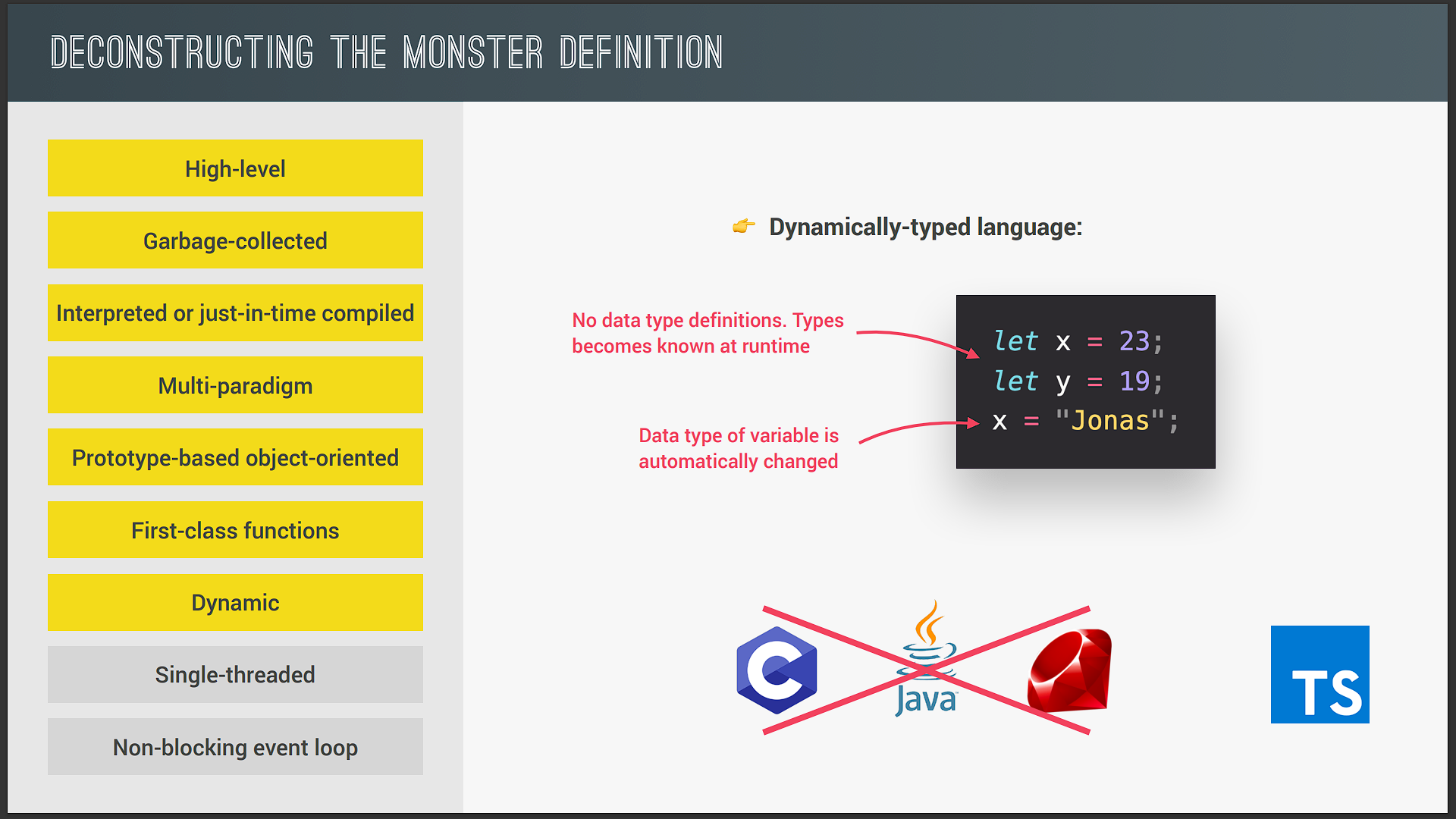


JavaScript is a language with first-class functions, which simply means that functions are treated just as regular variables. So we can pass functions into other functions, and we can even return functions from functions. And this is extremely powerful because it allows us to use a lot of powerful techniques and also allows for functional programming, which is one of the paradigms that we just talked about before.

And in fact, we have already used the power of first-class functions without knowing that they are called first-class functions. So remember this piece of code that we wrote for closing the modal window that we built before? So right here, we pass the closeModal function into the addEventListener function as if it was just a regular variable, right?

And actually, not all languages have first-class functions, but JavaScript has, and it is amazing. Believe me, it's really, really helpful.

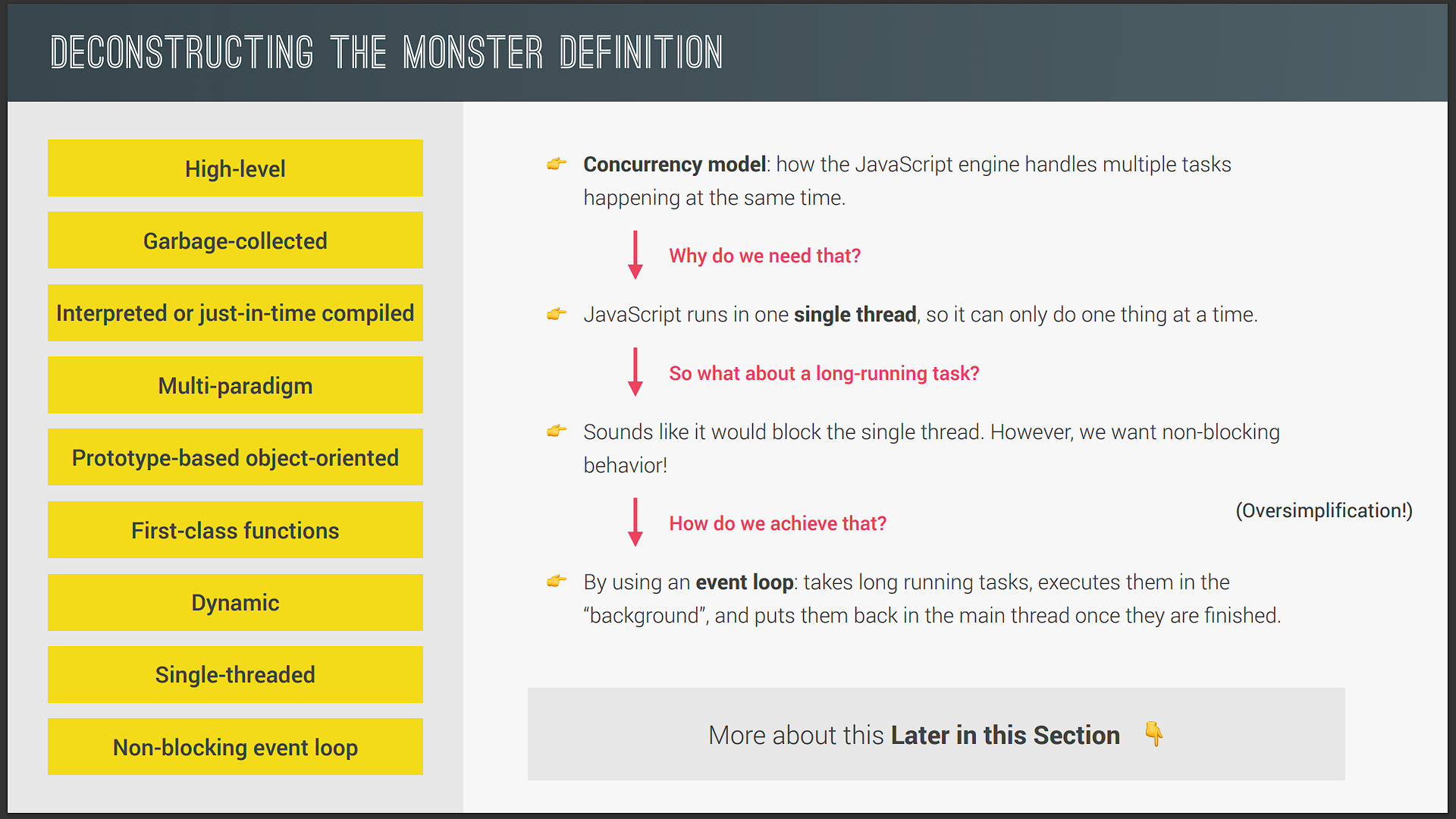
Dynamic:-



JavaScript is a dynamic language, and dynamic actually means dynamically typed. So as we've already seen, in JavaScript, we don't assign data types to variables. Instead, they only become known when the JavaScript engine executes our code. Also, the type of variables can easily be changed as we reassign variables. And this is basically what dynamically typed means.

Now, the same is not true for most other programming languages, where we have to manually assign types to variables, and this then usually prevents bugs from happening, which is the reason why many people say that JavaScript should be a strongly typed language as well. And if you yourself want to use JavaScript with types, then you can always look into TypeScript.

Single-threaded and Non-blocking event loop:-



First, what actually is a concurrency model? Well, it's just a fancy term that means how the JavaScript engine handles multiple tasks happening at the same time.

No, okay, that's cool. But why do we need that? Well, because JavaScript itself runs in one single thread, which means that it can only do one thing at a time, and therefore we need a way of handling multiple things happening at the same time.

And, by the way, in computing, a thread is like a set of instructions that is executed in the computer's CPU. So basically, the thread is where our code is actually executed in a machine's processor.

Alright, but what if there is a long-running task, like fetching data from a remote server? Well, it sounds like that would block the single thread where the code is running, right? But, of course, we don't want that. What we want is so-called non-blocking behavior. And how do we achieve that? Well, by using a so-called event loop.

The event loop takes long-running tasks, executes them in the background, and then puts them back in the main thread once they are finished. And this is, in a nutshell, JavaScript's non-blocking event loop concurrency model with a single thread.