# The Importance of Theory and Computer Science

Anand Kumar Keshavan
Chief Architect, Extentia Information Technology

What differentiates a good programmer from the average, and a great one from the good? Many people, especially management and executive-level folks ask me this question very often. My invariable answer is:

- A good programmer has more than working knowledge of the system that he is attempting to program while an average programmer may not have much of a clue about such matters. A bad programmer will not be aware of it at all and will probably show no interest in learning about such stuff.
- A great programmer, on the other hand, also attempts to understand why a program's environment works the way it does. This enables them to avoid errors and find errors faster than their counterparts who do not try to understand these vital aspects of the discipline called programming.

For some strange reason, the usage of theoretical aspects of computer science invites cynicism within software development organizations. Computer science is considered tedious, abstract, and far removed from the world of real programming. It would be interesting if such an attitude were found amongst doctors!

The other day, I found a developer struggling with a progressive web app that would break every time the user clicks on the refresh button of a browser while being offline. (I encourage the reader to figure this out, a very trivial understanding of how browsers work, and some common sense is needed). The real problem is that if you are a front-end developer or a programmer, you are no longer programming a computer; instead you are programming a browser. And hence if anyone wants to be a 'good' web developer, then it is vital that they spend some time understanding the environment that they are trying to

program – in this case the browser. A great programmer will probably know the theory behind how the DOM has been developed and how it has evolved over a period, and hence understand the difference between different browsers and their approaches towards the lifecycle of the DOM. If a front-end developer doesn't know what the DOM is, then that is a developer not worth hiring.


I have seen this pattern over the last three decades or so. In the DOS days, the good programmers had more than a working understanding of the interrupt handlers (21Hex, anyone?) of DOS. In case of Windows, the better developers always knew how DLLs get loaded, how Bit blit works, how events get propagated, and so on. And a similar case can be made for other environments such as Unix and Linux.

Better Java developers (if one is Java developer, one is programming the Java Virtual Machine), usually have a good handle on how the Java Virtual Machines handle memory, how threads are to be programmed, and such stuff. The great Java programmers are extremely comfortable with generic programming i.e., using generic types to build reusable code. This applies to C#, as well as Typescript – the new powerfully typed version of JS.

Typescript is a language that has been built using the foundations of type theory (the dreaded word, 'theory') and unless someone has more than the average understanding of type-level programming, generics and mapping domain entities to types, the person will never be very good at it. For example, Typescript supports algebraic data types (ADTs) which can be used to define very powerful type systems for defining domain entities. Now, Algebraic Data Types (ADTs) come from mathematics, and if one is not prepared to learn about it, then one is missing out on one of the most powerful features of the language. If, on the other hand, one has a working understanding of these topics, one can write fairly error-free code – code that is free of clutter and is readable as well as maintainable. If not, then one is better off using the plain old JavaScript where type mismatches always result in run time errors.

Frameworks like Angular and React have been built on the strong mathematical foundations of functional programming (more specifically Functional Reactive Programming), and if a front-end developer is using Angular without a working knowledge of these ideas, he/she is likely to get into trouble sooner or later. Both Angular and React encourage the use of Typescript as their choice of programming language. In case the developer is ignorant of the foundations of Typescript and these frameworks, then this will more than likely result in comical code. Comical code that is tragic in the areas of readability, extensibility, and maintainability!

Generic classes and functions are powerful features of languages such as Java and C#. The idea of generics comes from the functional world and type theory. Since many developers are not conversant with the theory behind generics, they end up creating poor generic types or avoid them completely. Even in the context of application modules such as domain entity validators, REST API controllers, and API consumers, generics can reduce the size of the code – thereby making the code much more testable and maintainable. But for some odd reason, generics are popular only among framework and library developers.

For engineering managers, and people who have some say in the selection of language/frameworks, tools, etc., it is desirable that they acquire at least some basic ideas about the theoretical foundations of some of these areas. By that, I do not mean that they must be able to solve the theoretical problems of computer science – what I mean is that they should know enough to understand the pros and cons of the technical decisions that are being taken by their team or by any other stakeholder. In my opinion, this plays an important role in the way one hires team members and how technical decisions are taken by the team.

When I am looking for developers, it is no surprise that I look for people who have a good understanding of their environment – the browser, JVM, JS engine, OS, and so on. Occasionally, I am surprised by folks who show more than a passing knowledge of the internals the environment/language, in other words, the underlying theoretical foundations. That really makes my day, but unfortunately those are becoming few and far between.

About Extentia Information Technology

A global technology and services firm that helps clients transform and realize their digital strategies. With a unique Experience Centric Transformation approach, Extentia's ground-breaking solutions are in the space of mobile, cloud, and design. The team is differentiated by an emphasis on excellent design skills that they bring to every project. As an SAP ISV-OEM Partner, Extentia's SAP Practice creates innovative solutions leveraging various features of the SAP Cloud Platform.

Focused on enterprise mobility, cloud computing, and user experiences, Extentia strives to accomplish and surpass their customers' business goals. The company's inclusive work environment and culture inspire team members to be innovative and creative, and to provide clients with an exceptional partnership experience.

www.extentia.com