JAVAWORLD

OPINION

# 12 predictions for the future of programming

**By Peter Wayner**

InfoWorld | Feb 3, 2014 6:52 AM PT

If hitting a target is hard and hitting a moving target is even harder, then creating a new hit technology is next to impossible because the shape and nature of the target morphs as it moves. Think of building a swish new laptop just as laptops are heading out of favor, or a must-have mobile app just as smartphones plateau, or a dynamite tablet experience just as the wearable future takes hold.

It's no secret that technology trends move fast -- and the tools and means for building those technologies constantly evolve. But if you don't lift your head up every once in a while to look past the next year's projects, you could end up coding yourself down an inescapable rabbit hole.

To help you prepare for -- or at least start contemplating -- a future that's screaming across the sky faster than we can see, we've compiled a dozen predictions about how the next five years of programming will shake out. Our crystal ball is very subjective, and some of the following conjectures might not prove universal. Some won't be fully realized in five years. Others are already true, but the extent of their truth is not as well-established or widely known as it will be fairly soon. Some may surface as half-truths because some faction of coders may take a different path. Some might even be flat-out wrong.

Despite all of those caveats, there's truth here in the main. Read them quickly because the future is changing faster than we know.

## Future of programming prediction No. 1: GPUs will be the next CPUs

Remember the days when people bragged about the CPU in their box? Now even the best CPUs rarely cost more than $200, while fancy graphics cards routinely cost $500, $600, or much more. Gamers love to brag about the power of their graphics cards, not their CPUs, and that's driving the market.

The rest of the world is slowly catching on. More and more software uses the GPUs. True, some of these early forays are inherently graphical processes, like the work of Web browsers, but increasingly we're seeing applications that have nothing to do with drawing fancy pictures being rewritten to use the parallel architecture of GPUs. Physicists use them to study matter; chemists use them to study reactions; astronomers simulate the galaxy with graphics cards; biologists crunch statistics via GPUs for population studies. And for a while this year, I heated my office by using my GPU to mine bitcoins.

Practices like these will only become more widespread. Soon better compilers will make it possible to write code and let the compiler detect when it can run effectively on the GPU. The academic tools can already do this at a limited level, but it will become more popular when it is easier for everyone to do so with any project.

Bank your bitcoins on this one coming true.

## Future of programming prediction No. 2: Databases will perform increasingly sophisticated analysis

Once programmers ceded control of data to something called a "database," they stopped being "programmers" and started being "database users." This isn't a bad thing. After all, databases are engineered to be more efficient at handling data than anything a normal human programmer can create, especially under tight deadline. They're also (usually) very efficient at using RAM and juggling the memory hierarchy. Dumping numbers into a database and letting the database do its thing is more often than not the most intelligent solution to dealing with data, and it's going to get even more intelligent as the databases themselves get more intelligent with time.

That's because databases of the future are certain to do more than just store numbers. Many database systems already have sophisticated report engines (aka "business intelligence"), and these extras will only become more powerful, enabling databases to run more sophisticated algorithms on tables, search more efficiently for patterns in the data, and do much of the work currently touted by the buzzword "big data."

This power and sophistication will be driven by the cost of moving data around. Simply extracting the information from the database and handing it to a separate "big data" package will become increasingly time consuming and require much more programming. Leaving the data in the database and letting its engine perform the analysis will be much faster because it will limit the overhead of communication, as well as decrease the amount of programming necessary to extract value from the data store.

## Future of programming prediction No. 3: JavaScript for everything

No, JavaScript won't be the only language in the programming world, especially given the surprising number of ads for Cobol programmers still being filed, but it will certainly begin to seem that way. JavaScript is inescapable on the browser, which now dominates almost everything a client computer does. Now the server side is embracing it with tools like Node.js.

JavaScript will assuredly become more dominant in other areas as well. Once the only way into a smartphone was to write code in the native language demanded by the manufacturer: Objective-C for the iPhone; Java for Android; C# for Microsoft. Now most mobile developers can get the performance they want out of HTML5 applications running in browserlike views. The result may not be as zippy as native code, but the JavaScript is good enough and portable to the Web, too.

The browser isn't just colonizing phones; it's eating entire platforms. Chrome OS and Chromebooks are making regular operating systems obsolete. Why worry about that layer when JavaScript and the browser can do everything?

The mainframe will have Cobol. Biologists will probably stick with Python. Linux will be written in C. But almost everything else is fair game as JavaScript gobbles the world.

## Future of programming prediction No. 4: Android on every device

When the browser doesn't win, Android is close behind. Camera designers, threatened by cellphones with good lenses, started putting Android on cameras, so now you can run Instagram on a Nikon. Does that make Nikon a computer company now? Does it even matter?

There are Android refrigerators, car stereos, watches, televisions, even headphones. Some complain that the UI is too complicated because it can do too much, but that's missing the point. The UI layer can always be simplified. If Android is running underneath, the platform will dominate.

It's going to get even more complicated. PC manufacturers are looking at the burgeoning tablet world and feeling left out. Their solution is to run Android on Windows and let people use their Android apps on their desktops, too. Some just run the stock Android emulators used by programmers, but others are looking beyond that to create brands like "PC Plus." Once Android takes over the PC, it may combine with the browser to push Windows native apps into a distant third place for mindshare on the box.

## Future of programming prediction No. 5: The Internet of things -- more platforms than ever

One side effect of Android (and Linux) colonizing the world is that more and more objects will be joining the Internet of things. Android on your refrigerator will mean the opportunity to write code for the refrigerator itself, whether it's an app for displaying kid art, some calorie-counting scold, or a recipe suggestion engine for what's left inside the ice box. Who knows?

Of all the many new platforms to come, the most important will be the car. Navigation and shopping are just the beginning. When the autonomous cars roll out, there will be even more opportunities for developers to cash in on car location and information.

The key is to figure out which new features are essential to the new domains. While operating systems like Android offer a unifying core, each platform will require customized features. Robotic toys, for instance, may need content filters to provide age-appropriate content. Thermostats will want to interact with the weather forecast. Building these APIs with the right features will be key to colonizing the Internet of things.

## Future of programming prediction No. 6: Open source will find new ways to squeeze us

For all of the success of open source stacks like Android, Ubuntu, or MySQL, there remains a sticky problem with finding the revenue to support development. There are plenty of good stories about how open source code has helped hackers, but there aren't very many examples of how companies built a relatively stable ecology that let the programmers buy health care or food.

The vast majority of open source companies distribute what might better be called a demonstration version under an open source license. Then some kind of secret sauce is kept locked away to give the programmers something to bargain with. It's the way of the world. Why buy the cow if you can get the milk for free? The best open source projects will find a way to tighten the screws in a comfortable way without scaring away customers.



## Future of programming prediction No. 7: WordPress Web apps will

**abound**

The biggest mistake that the Obama administration made was trying to build its insurance exchange websites from scratch. No one does that any more. Why bother when you can add a plug-in to WordPress? If you're really picky, you could work with Joomla or Drupal. The point isn't really which platform, just the fact that there are fewer and fewer reasons to create your own Web apps because so much functionality is built into the dominant frameworks.

The game gets even more interesting when you start hacking the code. WordPress has its own editor built into it, so you can do your development inside WordPress, too. There's no debugger, but you can get around that. If WordPress adds a nice database browser like PHPMyAdmin and provides a bit of basic debugging tools, development will really accelerate.

## Future of programming prediction No. 8: Plug-ins will replace full-fledged programs

Basic Web apps aren't the only ones riding the power of code snippets that can be plugged into a bigger framework. Photoshop used to be the dominant engine for reworking images, in part because of the fertile world of plug-ins. Now the newer apps like MagicHour have made plug-ins even simpler. MagicHour users, for instance, can share filters just like they share photos. Most major platforms offer a good plug-in API, and the ones with the best have fertile ecologies filled with thousands of modules, libraries, and plug-ins.

This burgeoning ecology for code means that programmers will write more snippets and fewer applications. The right bit of glue code can be a million times more powerful than a great, hand-built application with megabytes of binary file. A small snippet can leverage everything in the entire ecology. A big app must do everything on its own.

The savvy programmers will learn to leverage this by creating plug-ins, not programs. They'll learn the APIs for the host systems and string together parts. Very few will ever build anything from scratch. We'll all be part of the emerging Borg.

## Future of programming prediction No. 9: Long live the command line

While it has become easier to click your way to a working app, a surprising amount of work is still done in text editors and terminal windows. So, contrary to what you might think, the command line will not be going away. In fact, more and more modern tools will work only with the command line.

The fact is, command lines are too flexible and too universal to be supplanted. Pretty GUIs with clicky interfaces and drag-and-drop widgets may get attention, but the programmers keep reverting to text.

Ease of scripting is at the center of the command lines' continual resurgence. While some companies such as Apple have decent tools for scripting GUIs, they've never been as flexible or as stackable as the command line. How many people write shell scripts for their Macs, and how many write AppleScripts?

The command line will live on because it's simple and extensible. If you have a script, you can easily extend it and glue it into other scripts.

## Future of programming prediction No. 10: Dumbing it down will fail

For the past 50 years, programmers have tried to make it easy for people to learn programming, and for 50 years they've succeeded -- but only at teaching the most basic tasks. Ninety-five percent of the world may be able to figure out if-then-else structures, but that's not the same thing as being a programmer.

That won't prevent well-meaning folks from trying to dumb down programming even further so that everyone will be able to do it. Evangelists will sell big dreams of a world where everyone programs, and they'll wheedle big grants with claims that the only way country X can stay ahead of the game is if every citizen in X learns to program Ruby or their VCR.

Alas, true programming means understanding the unseen numbers dancing around in the little box. It means understanding the conventions for creating software and for partitioning responsibility, so the software can run cleanly. Only a few brains seem to be able to handle this work, and it seems unlikely that the proportion of people with this ability will change markedly after 50-plus years of trying. Teaching everyone to grok if-then-else clauses is a nice idea, but it's not the same as creating more programmers.

And no, developing languages whose syntaxes are more "English-like" won't help either.

1 | 2 | NEXT >