- Blog Home

Don't Let Tech Lingo Hold You Back: 7 Common Developer Terms Explained

Brian Cooksey / August 26, 2014
Tweet

Spend a little time with developers and it won't be long until you hear acronyms like "DB" and "AWS" or see the word "get" repeatedly misspelled as "git". Instead of letting this lingo leave you befuddled, it can be extremely beneficial to learn the basics of common tech terms. Gaining this knowledge can yield increased efficiency in a marketing role, higher effectiveness in a sales position, or smarter decision making in a small business.

The following is a plain-English guide to some of the common terms and tools we developers use, explaining them in a way that will hopefully make sense to folks on the other side of the code.

Tech Terms Explained

What is the Cloud?

Let's start with the term you probably hear the most, the mystical cloud. If your data lives on the internet, whether that's your list of favorite bands on Facebook or documents in your Google Drive account, then it is in somebody's cloud. So, what exactly is the cloud? Well, this:

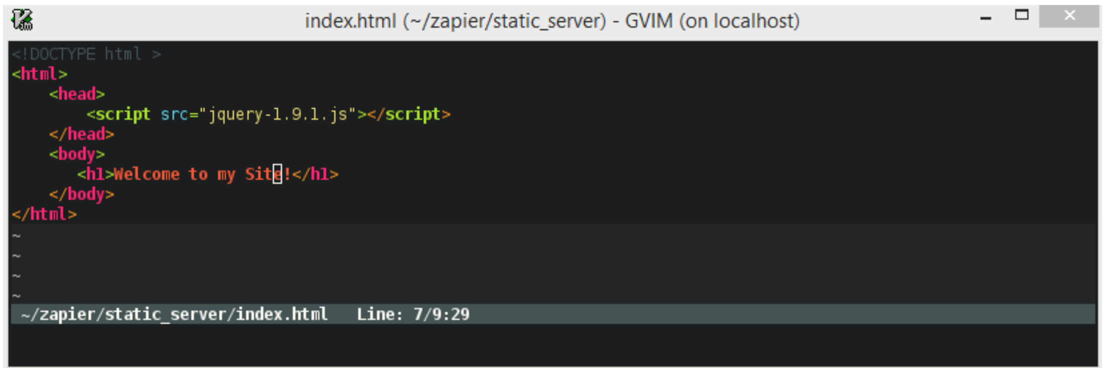
A bunch of computers!? Yup. No gnomes, no elves, just row after row of computers. We typically call them *servers* instead of computers, but at their heart, the machines on those shelves have the same pieces as your laptop, minus the screen and keyboard. I know, not nearly as cool as the movies make it.

Nonetheless, cloud computing is very important. To understand why, imagine for a second what the old process was like. Say a backroom at the Zapier office contains five servers to run our website. As traffic grows, Zapier eventually decides it's time to add another server, which means placing an order, waiting for it to ship, then having to physically set it up in the office and install everything needed to run the site. That process is on the order of days.

In today's viral culture, where a website's usage can skyrocket overnight, days won't cut it. If, for example, Zapier ends up on the frontpage of a popular news site, it needs another server *now*. Enter cloud computing, where a developer can rent a server from a company's datacenter (cloud) and have it up and running in minutes. That speed lets Zapier remain stable with increased traffic, so it never wastes money on servers it doesn't need, nor get caught with too few.

There are a bunch of companies that offer cloud resources. The most basic are web hosting companies like **Bluehost** or **DreamHost**, where you get one-click installs of WordPress and a simple interface to tweak your site, while they take care of the rest. At the other end are products like **Amazon Web Services (AWS)** or **Azure** where you get an empty server and can set it up however you want. Somewhere in the middle are the Platform as a Service (PaaS) companies like **Heroku**, who try to make running servers as easy for developers as the web hosting companies make running a WordPress site for non-devs.

What is a Text Editor?



```
<!DOCTYPE html >
<html>
    <head>
        <script src="jquery-1.9.1.js"></script>
    </head>
    <body>
       <h1>Welcome to my Site!</h1>
    </body>
</html>
~
~
~
~
 ~/zapier/static_server/index.html   Line: 7/9:29
```

You've likely seen one when you've walked by a developer's desk. A monitor full of colors and line after line of what looks like very broken English.

This is a text editor, a developer's version of a hammer. It's the tool we use write code. You might be surprised to find out that code is nothing more than text in a file. A developer edits code just like you might edit a document in Microsoft Office.

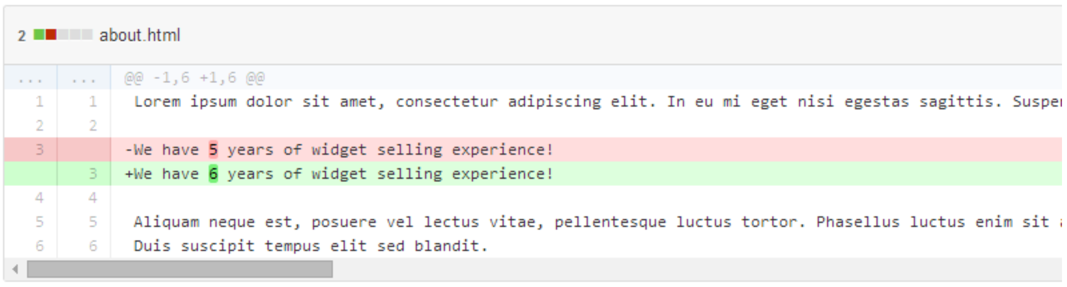
The difference between word processors and text editors like **Emacs**, **Vim**, or **Sublime Text**, is that editors are designed for code. To work on software, a developer often moves between files—the focus is on a set of documents rather than a single one. Editors make navigating the codebase quick and easy.

Another difference is in semantics. In a normal document, you are conveying a thought. If a paragraph or sentence doesn't fit, you can rewrite it to get the same meaning. For code, you are writing instructions. Each line is important and the specific words on each line are significant. That is why editors highlight the code in different colors. Imagine if your word processor diagrammed each sentence for you, highlighting the nouns and verbs in different colors. That's like what the editor does for a developer, letting them quickly tell what the lines of code are doing.

There is another class of editing software called Integrated Development Environments (IDE). These are text editors that bundle in other tools devs use regularly, allowing us to live inside one program instead of switching between several. A helpful comparison is to think of a text editor like a pocket knife and an IDE like a Swiss Army Knife.

Note: Typically, an IDE is best suited to a particular programming language. If you hear a dev mention **Eclipse**, they likely program in Java. The same is true for **PyCharm** and Python, or **Zend Studio** with PHP. Of course, devs like to tinker, so it's not uncommon for us to load up our IDE with plugins that make it work for any language.

What is Version Control?



```
2 about.html
@@ -1,6 +1,6 @@
 Lorem ipsum dolor sit amet, consectetur adipiscing elit. In eu mi eget nisi egestas sagittis. Suspe

-We have 5 years of widget selling experience!
+We have 6 years of widget selling experience!

 Aliquam neque est, posuere vel lectus vitae, pellentesque luctus tortor. Phasellus luctus enim sit
 Duis suscipit tempus elit sed blandit.
```

Nobody likes to lose their work. Perhaps you're the type of person who insistently pounds the "Save" button on your documents, or maybe uses cloud services like Google Docs where everything is saved automatically all the time. Whatever the case may be, developers are just as paranoid. That's why we use version control.

A Version Control System (VCS) is software that helps developers track changes to code. You may have heard the developers on your team talk about Git, Subversion or Mercurial. These are all different tools that do version control.

There are two main benefits to version control. The first is history. As a developer works on code, making changes and testing, they eventually reach a good stopping point. To save their current status, they "commit" their changes to the VCS. From now on, the VCS will remember exactly how the code looked at that moment. As the dev makes more changes, they can commit again, creating a second snapshot. As this history grows, devs can look back and see what changes were made and when. If some bad changes are made, they can restore the code to a previous commit, effectively removing the bad edits (kind of like you might restore your hard drive from a backup to get rid of a virus).

The second benefit of version control is collaboration. If you have ever tried to edit a document with multiple people, you've likely run into issues where somebody edits an older version and gets stuck trying to incorporate their changes into the current one. Or worse, maybe they upload that older version and wipe out everyone else's work. These types of situations are where version control really shines.

When a developer is ready to share the commits they created, they do what's called a "merge". In a merge, the dev tells the VCS to download the latest version of the code and intelligently combine all their changes with it. This saves devs lots of work because they don't have to worry about having old copies of the code. The VCS magically brings them up-to-date and figures out how their commits fit with the newest version. Once all the commits are merged in, the dev can upload the changes so the rest of the team can see them.

So if that is what version control offers, what are all these web apps like **GitHub**, **Bitbucket**, and **SourceForge** adding in? Well, remember how that merge downloaded the latest code? It has to download it from somewhere, and that somewhere is the space these apps fill.

Sites like the ones above provide hubs where devs can go look for the latest code. When a team uses one of those apps, they are agreeing to make that app the place where the final version of their work always ends up. These sites also add in helpful features like bug trackers, wikis, and reporting tools so all the knowledge relating to the project is contained in one app.

What does "Fork Me on GitHub" mean? Perhaps the most amazing thing about these hubs, is their promotion of open source software. Open source refers to software whose code is published publicly for others to modify and distribute as they wish. Sites like those listed above are popular places for developers to share projects they are working on. On GitHub, for example, users can allow their code to be "forked"—fully copied elsewhere—by others. Making the code available means other developers in the community can contribute to the project by reviewing and editing the code, then sending those edits off to the original author for approval.

What is a Database?

Database ("DB" for short) is another one of those mysterious words that regularly gets thrown around. You are probably aware that it holds your company's data, but what is it exactly?

It turns out storing information is difficult when you have lots of it. Think for a moment about a small bookcase. With 20 books on the shelves, it's not too hard to skim the cover of each to find the one you want. Now picture a library with thousands of books. You certainly don't want to waste your time going through each book one-by-one. You need them organized in a way that makes it easy to find the book you are looking for.

Databases are the equivalent of a library for a program's data. A database is a program designed to let other *programs* store data in them. They handle the organization aspect, making it easy for developers to store and retrieve info that is important to their app. Removing the concern about how to store stuff lets devs focus on new features for the app they are building.

Databases come in a wide variety, each designed for particular strengths. Some are tailored to ensuring they never crash. Others make performing complex searches easy, while another group trades searching for the ability to record a whole bunch of data really quickly. Depending on the app, one database may be better suited to the job than another.

Among all these different types, though, there is one set that tends to be developers' go-to: relational databases. **MySQL**, **PostgreSQL**, and **Oracle** fall into this category. These databases do several things, one of the most relevant to biz folk being search and data aggregation.

In database lingo, searching is called "querying", and these DBs provide so much flexibility that there is a special language to do it in. You may have heard a dev mention writing a "see-quill" statement. They are referring to Structured Query Language (SQL). This is the special language built into MySQL and other relational databases that lets apps ask the database questions like "What are all the orders for this user?" or "Give me user signups for the year, grouped by month." There is a lot developers can do with SQL statements and chances are some of the reports your app generates are relying on SQL to gather the data out of the database.

What are HTML, CSS, and JavaScript?

You're looking at it! HTML, CSS, and JavaScript are the trifecta of the web. They are what web pages are made out of, with each piece serving a specific role.

Let's first consider HyperText Markup Language (HTML). In a webpage, the HTML is the content. It's a mixture of markup symbols and text that specifies what will be on the page and what pieces belong together. Here is a simple example to show how it works:

```
<p>This is a paragraph. It has a few sentences in it. Like this one.</p>
<p>Here is a second paragraph. It has two sentences.</p>
```

The above HTML uses paragraph "tags" (the `<p>`s) to say the page will have two paragraphs. The words between the opening and closing tags (`<p>` and `</p>` respectively) are the actual content to be shown on the page.

Getting used to HTML's syntax can take time, but it actually has some similarity to how books works. In a book, the author indicates a new paragraph by starting a new line and indenting the first word. A new chapter starts on a new page and has a heading. Using visual cues, the author indicates the structure of the book. With HTML, developers use tags rather than visual cues to communicate the structure.

Structure alone, though, doesn't make for a very compelling website. To engage users, a site needs to have *style*. This is where Cascading Style Sheets (CSS) come in. CSS is the language used to specify how a site looks. Through a set of rules (a style sheet), a developer says where the tags on a page should be displayed, and what they will look like (font, color, etc.). Here is a quick example:

```
p {
    color: grey;
}
```

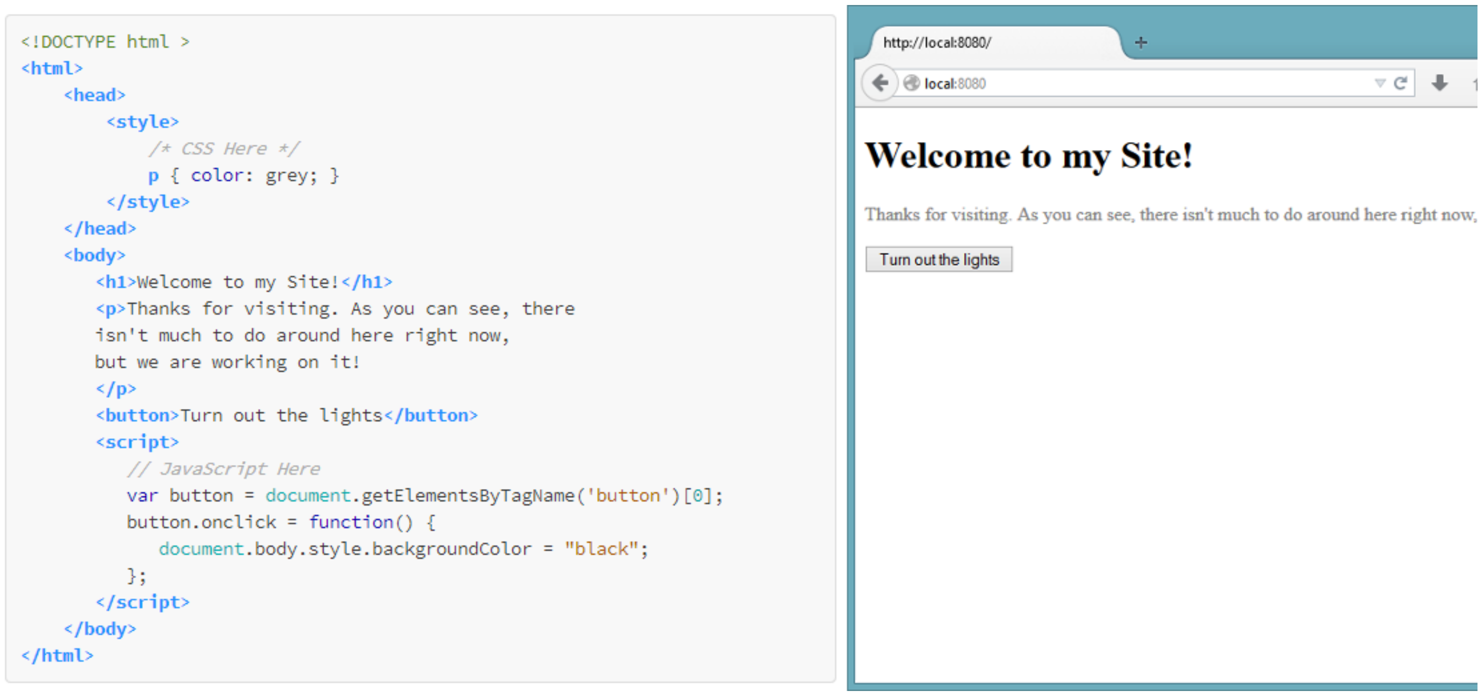
This CSS rule says that all paragraphs will have grey text. The rules in a style sheet can be generic, like this one, or very specific, perhaps to only make one paragraph on one particular page of a site be a different color.

Equipped with HTML and CSS, a developer can make beautiful web sites, but the experience won't be interactive. To pull off effects like drag-and-drop or auto-suggestion while typing in text boxes, a developer needs JavaScript. JavaScript (JS) is a programming language built-in to web browsers. It allows developers to write little programs that are sent along with their webpages. These programs can change anything about a webpage based on events such as button clicks, mouse movement, or typing.

It's beyond the scope of this post to teach you how to write JavaScript, but here is a small JavaScript program so you at least know what it looks like. The code below changes the background color of a webpage to black when you click a button.

```
var button = document.getElementsByTagName('button')[0];
button.onclick = function() {
    document.body.style.backgroundColor = "black";
};
```

Combining our little examples together, the illustration below shows the complete HTML, CSS, and JavaScript on the left and the resulting web page on the right.



```
<!DOCTYPE html >
<html>
    <head>
        <style>
            /* CSS Here */
            p { color: grey; }
        </style>
    </head>
    <body>
      <h1>Welcome to my Site!</h1>
      <p>Thanks for visiting. As you can see, there
      isn't much to do around here right now,
      but we are working on it!
      </p>
      <button>Turn out the lights</button>
      <script>
         // JavaScript Here
         var button = document.getElementsByTagName('button')[0];
         button.onclick = function() {
            document.body.style.backgroundColor = "black";
         };
      </script>
    </body>
</html>
```

Welcome to my Site!

Thanks for visiting. As you can see, there isn't much to do around here right now,

Click image to expand

What Does a Typical Day Look Like Using These Tools?

To bring this all together, we're going to walkthrough how a developer (me) uses these tools on an average day. The first thing I do in the morning is go grab the latest code from GitHub. With a single click, my **version control system** downloads the changes my teammates made the day before and adds them into whatever I'm working on, so now I'm up-to-date.

Ready to start work, I open a file in my **text editor**. I've been thinking it would be helpful to find out what new apps are joining Zapier, so I'm going add a field to our registration form that asks for the app's URL. Finding the right spot in the file, I write some **HTML** that creates another text box and asks for the domain. To share this change with my teammates, I need to add it to the **version control system**, so I create a new commit. The commit includes my name, the date I made the change, and a before-and-after so everyone can see exactly what I updated. After the commit is made, I upload it to GitHub for the rest of the team.

New feature in hand, it's time to get this on to the site! Our servers run in Amazon's EC2 **cloud**, so I do a bit of work to get the new code installed on each of them. Once the feature is live, I can come back in the afternoon and do a query in our **database** to get a list of apps added today and what the URL is for each.

More Tech Terms

Hopefully you read those last three paragraphs and felt more enlightened as to what was going on. Of course, there are many terms that we skipped over. We could keep talking about code repositories, the terminal or server logs. That's why I need your help.

What are the other tech terms that make you scratch your head? We're planning to continue this tech lingo series, but need to know what terms to tackle next—leave a comment below or email me with suggestions.

If you enjoyed this post, you might also check out our free guide: "An Introduction to APIs".

Zapier is an app integration and automation tool made for tech and non-tech professionals alike. With a few clicks, you're able to automate adding contacts to a CRM, lines to a spreadsheet, data to a database and more.

Credits: Header photo courtesy Ivan Dervisevic. CERN Server photo courtesy Florian Hirzinger.

Tweet

Get Productivity Tips In Your Inbox

Learn about workflow, company building, and how to get things done.

Your email...

Subscribe

What is Zapier?

Zapier gives you internet superpowers by making it easy to get your web apps to work together. Use Zapier to integrate GitHub with over 400 other apps.

About the Author

Brian Cooksey is a Platform Engineer at Zapier and Missouri native.

Follow @brian_cooksey
comments powered by Disqus

- **Get Started**

- How It Works
- Pricing
- Sign Up for Zapier

- **Explore**

- App Directory
- Search for Zaps
- Case Studies
- Use Cases

- Customers
- **Helpful**
- Help & Support
- Learning Center
- App + Feature Updates
- Contact Support
- **Developers**
- Developer Platform
- Documentation
- API Status Board
- Engineering Blog
- Bug Bounty
- **Company**
- Blog
- About
- Brand + Logos
- Jobs

makes you happier :)