TestRail

The Skills for Agile Testers E-book

The Skills for Agile Testers E-Book

In a modern software engineering team, keeping pace with the latest tools, skills and practices can be a significant challenge. Make sure you stay ahead of the Agile Testing curve by implementing the Agile Practices below. Combined with some Technical Skills that Aren't Automation, and by selecting one or more of the Top Coding Skills to work on – you can take both your team, and your career to the next level.

Like the Content? Want More?
Sign up for our weekly Testers Email here on our blog!

SIGN UP HERE!

Contents

Everyday Agile Practices for Every Tester

It's a big, Agile world out there, filled with lots of advice about how Agile practices can help your company deliver more stuff, better and faster. As a tester, you might see those Agile practices as being great ideas, but be unable to get them implemented in your organization. Perhaps instead, you can incorporate some of them into your everyday testing practice?

Here are three ways that any tester can implement some fundamental principles of the Agile Manifesto into their day-to-day work.

Working Software: Minimize Necessary Documentation

In some organizations, highly regulated industries for example, it's difficult to reduce the amount of documentation required to prove that appropriate testing has been carried out. For these companies and testers, documentation is a serious part of their work, providing external auditors with an understanding of everything that has been tested. Since most companies aren't regulated though, testers have a lot of freedom in how testing is performed in order to reduce unnecessary documentation.

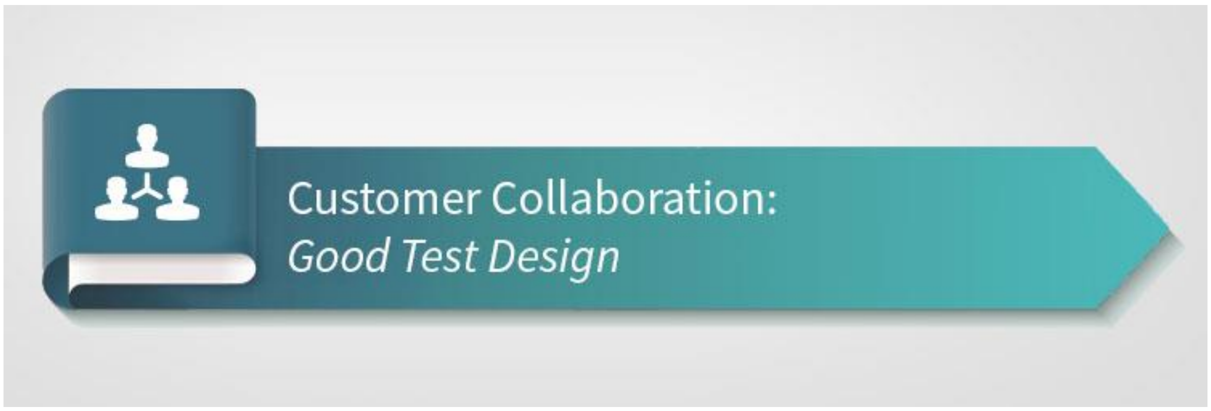
Session-based testing (allotting a specific amount of time, or a session, in which to complete an exploratory test), or test charters (a way to document the details of an exploratory test session), are ways to reduce documentation while still maintaining the desired level of testing evidence. Good test charters contain rich descriptions of the ways in which testers interacted with the product. They also contain bugs that might be logged, notes about potential issues, and questions that can be used to enhance regression suites, create further test charters, or inform acceptance criteria in new features.

A sample test charter for a 90-minute session, testing e.g. a checkout page on a website, would include technical information -- like the browser, operating system, and any other information pertinent to the environment in which you are testing. Then you might make a claim: "Today, I will test the checkout page of XYZ Widgets Website", and make a shortlist of areas that you want to cover: shipping name and address, billing name and address, and billing information, might be part of a checkout page for example.

In the charter, you keep detailed notes about your activities. There is no script. As you are testing the shipping name and address, you might find that no zip code is required to save shipping information. In your notes, you describe what happened, and what tests you decided to perform because you found out that the zip code is not required. You can write about issues and bugs too.

Furthermore, the test charter can be a collaborative artifact in development teams, shared with product managers and developers using whatever communications tools are available. The charter can be read by developers and project managers, who in turn can leave their comments, questions, and concerns, leading to meaningful dialogue about the software under test.

Customer Collaboration: Good Test Design



Testers serve the end user, but in a collaborative development environment, testers work with developers, product managers, and customer support in order to do their work. Robust test cases and acceptance criteria are generated by getting feedback from developers, product managers, and even consumers (or their representatives in the company) to ensure tests are designed that address a wide variety of use cases.

Collaboration with customer support can help testers get into the mindset of the customer during exploratory testing. Customer support hears customer issues daily, and can tell testers how users are engaging with the application.

Getting realistic use cases and user stories can help frame the tester's mindset for exploratory testing. This mindset can be extended as testers identify scenarios that are best suited for automated regression and write end-to-end tests from the customer's perspective. In this way, testing becomes a channel for helping to improve product quality, and better acts as an advocate for the customer.

Responding to Change: Virtuous Feedback Loops

The final principle of the Agile manifesto highlights responding to change. Feedback from testers helps development teams by providing them with more information with which to respond to the changes that they may have introduced to the product. Testers have many opportunities to provide feedback to the entire business, and to involve the business in the process of getting that feedback.

For example, testers can invite the sales or marketing teams to test a major feature before release, getting valuable feedback from them as they see the feature for the first time, while also providing them with an experience of being a part of the testing process. The feedback garnered from this group testing event, provided to the development team, can yield interesting bugs, as well as insights into how end-users might engage with the finished product.

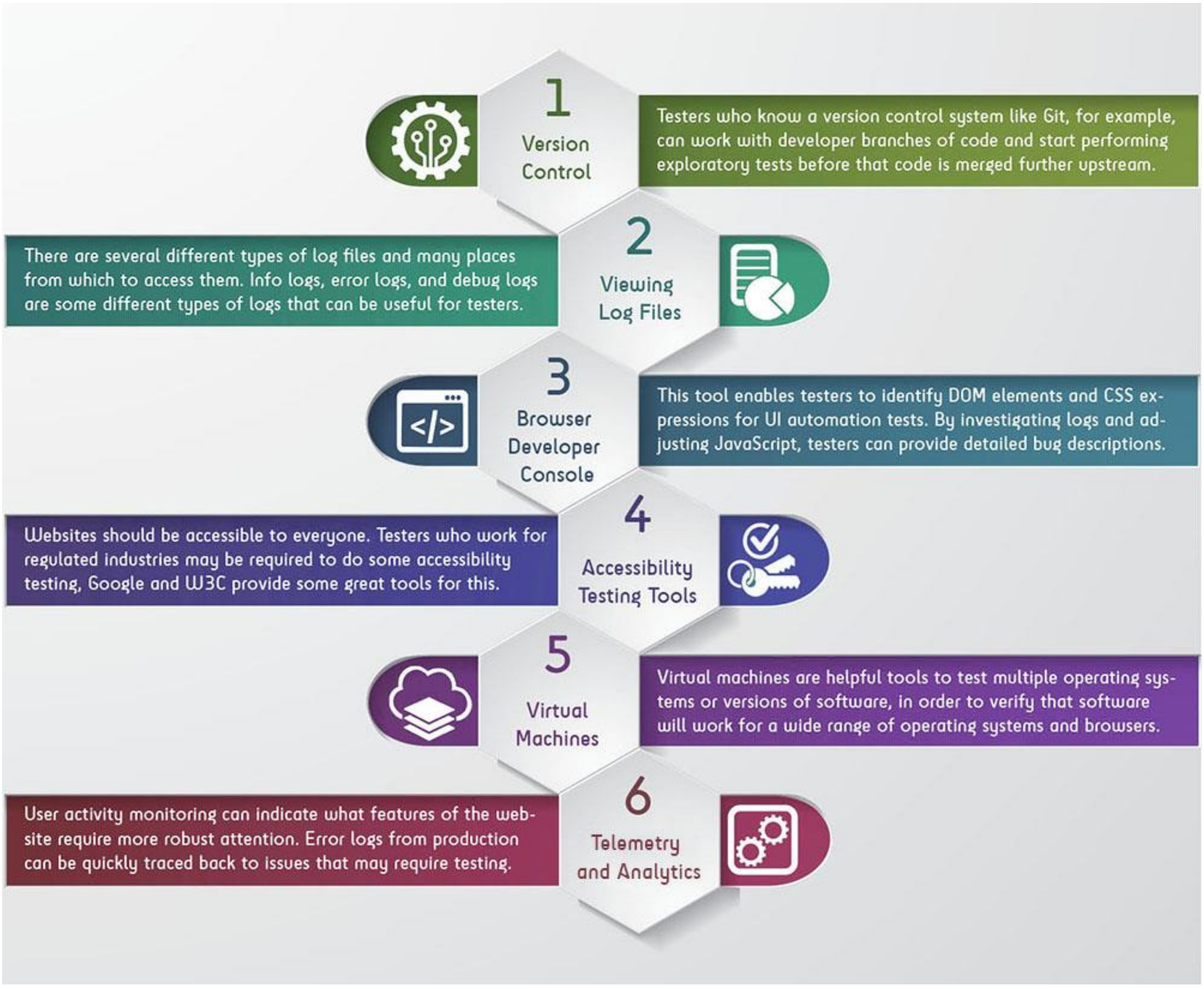
Everyday Agile Practices

Every tester can look to the principles of the Agile manifesto, and engage in any number of practices that will improve their daily work. Some of them might be transformative, like having lunch while introducing a new feature; others might be small, such as using test charters as a collaborative documentation tool. But they are powerful nonetheless, because they make testing work more meaningful.

Like the Content? Want More?
Sign up for our weekly Testers Email here on our blog!

SIGN UP HERE!

6 Technical Testing Skills that aren’t Automation


1
Version Control
Testers who know a version control system like Git, for example, can work with developer branches of code and start performing exploratory tests before that code is merged further upstream.
2
Viewing Log Files
There are several different types of log files and many places from which to access them. Info logs, error logs, and debug logs are some different types of logs that can be useful for testers.
3
Browser Developer Console
This tool enables testers to identify DOM elements and CSS expressions for UI automation tests. By investigating logs and adjusting JavaScript, testers can provide detailed bug descriptions.
4
Accessibility Testing Tools
Websites should be accessible to everyone. Testers who work for regulated industries may be required to do some accessibility testing, Google and W3C provide some great tools for this.
5
Virtual Machines
Virtual machines are helpful tools to test multiple operating systems or versions of software, in order to verify that software will work for a wide range of operating systems and browsers.
6
Telemetry and Analytics
User activity monitoring can indicate what features of the website require more robust attention. Error logs from production can be quickly traced back to issues that may require testing.

When testers hear the words "technical testing", our minds immediately jump to "automation." Testers are often unaware that many highly valuable technical testing skills have nothing to do with automation. It is invaluable for any tester to be able to skilfully use the different tools that help build software. There are hundreds of tools, free and paid, that can be in the tester's arsenal; here, we look at six of the most popular tools and skills that aren't automation.

Version Control

Version control systems, like Git and SVN, are ubiquitous among software developers, with good reason. Using version control means that software can be worked on, simultaneously, by multiple developers, without them stepping on each other's toes. If something goes wrong with a deploy, the software can easily be rolled back to the last working version until a fix is generated.

What does version control mean for testers? Testers who know a version control system like Git, can work with developer branches of code and start performing exploratory tests before that code is merged further upstream. For example, if a developer has made a branch for a new search feature, a tester who knows Git can get notifications about the status of the developer's work. In some circumstances, testers who have sufficient test environments can pull a feature branch into a test environment before it is merged into master, and perform tests against the branch. This is a practical way to "shift left", and give testing feedback earlier in the software development life. There are many blogs, videos, and resources to learn Git. One helpful, free resource is ProGit.

Viewing Log Files

Log files are stores of data waiting to be mined. Accessing logs can seem difficult at first, but there are several different types of log files, and many places from which to access them. Info logs, error logs, and debug logs are just some different types of logs that can be useful for testers. Info logs keep track of information such as different services starting and stopping on the product, or the processes that happen during that time. Error and debug logs have information including diagnostic messages, to help developers and testers diagnose potential issues. Some error logs may be accessed through the web console, while others will require access to internal systems where developer or production code is running.

Browser Developer Console

An overlooked and underused tool in testing is the developer console, available in all web browsers. For testers who are developing UI automation tests, this tool is necessary to identify DOM elements. Sometimes, building a CSS expression for difficult-to-locate elements is necessary and developer tools make this possible. By investigating logs and adjusting JavaScript, testers can investigate bugs and provide detailed descriptions of the problems they see. There are many tutorials available that help beginners learn to use the console methods that are useful to conduct a deep investigation of a website.

Accessibility Testing Tools

Websites should, ideally, be accessible to everyone. Testers who work for regulated industries may be required to do some accessibility testing. Developers who do not face a disability, such as vision or hearing impairment, may not have accessibility concerns at the forefront of their mind. Fortunately, there are some great tools that are easy to implement to check for basic accessibility requirements. The W3C markup validation service is a fast and simple way to validate all the markup on a webpage to ensure that it meets basic W3C accessibility standards. In addition, Chrome offers some free browser tools, available on GitHub, to use in developing and checking websites for accessibility.

Virtual Machines

Virtual machines and containers are helpful tools to test multiple operating systems or multiple versions of software. Often, testers are required to verify that software will work for users with a wide range of operating systems and browsers, but having several computers with different operating systems may not be realistic for some companies. Additionally, most computers will only support one version of each browser, but some companies may support older browser versions. Virtual machines are often a good solution for companies that support older machines or browser versions. Virtual machine software like Virtual Box is free, and companies like Microsoft even offer VM images of their IE and Edge browsers for download and test.

Telemetry and Analytics

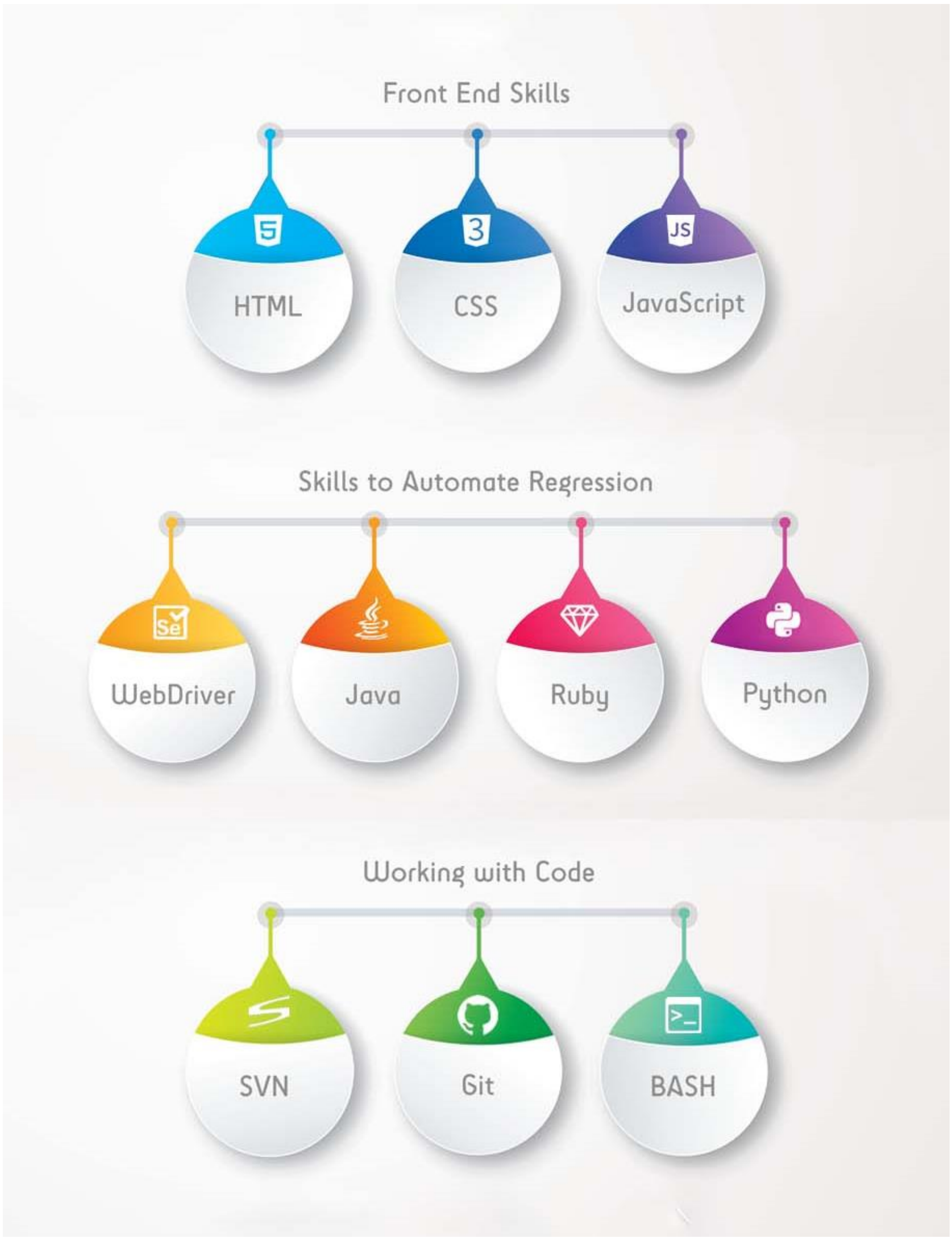
DevOps skills and tools are your window into the backend of the product. Often, the DevOps team has access to monitoring tools, and monitor performance, user activity, errors in production, and other behavioral changes in the website that might indicate security concerns. Some of these changes, for instance, include how much traffic is coming into the site at any given time. Very high spikes in traffic can indicate potential problems allowing a denial of service (DDoS) attack to occur.

How can you use this DevOps information in your tests? User activity monitoring can indicate what features of the website are updated, changed, or require more robust attention during regression. Error logs from production can be quickly traced back to issues which may require testing, and subsequent fixes. Security issues can be mitigated earlier in the testing process if some security tools are used. ZAP Proxy tool is a free tool provided and maintained by OWASP. With the tool, you can scan your site for security concerns, helping prevent these issues from reaching production.

Tools applied in appropriate situations enhance problem solving by providing more information, more efficiently, with less test effort. Testers, who regularly employ technical skills to enhance their testing, can spend more test cycles investigating interesting problems, and less time doing shallow tasks. Check out the resources below to learn the skills highlighted in this article below:

ProGit
Chrome Dev Tools
Accessibility Testing Skills
Microsoft Virtual Machines
Virtual Box
ZAProxy

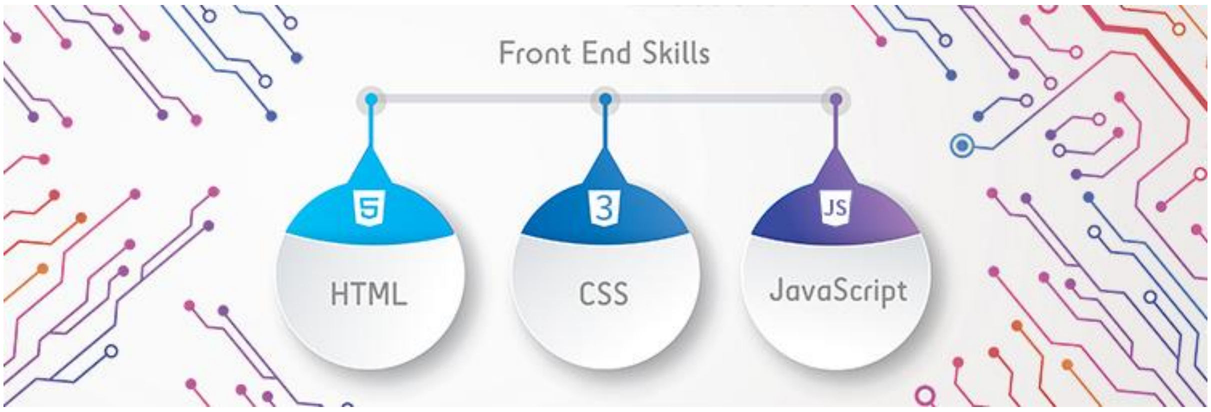
Top Programming Skills for Testers



With literally thousands of programming languages and new technologies being created daily, figuring out what technical skills to learn can be overwhelming for testers. Fortunately, many web and mobile apps tend to work with a similar tech stack. This means that learning some core skills can help testers work in a variety of environments, large and small. Let's talk about the most useful skills for software testers, and where to find information and training.

“Do the thing you think you cannot do.” - Eleanor Roosevelt

Front End Skills: HTML, CSS, JavaScript



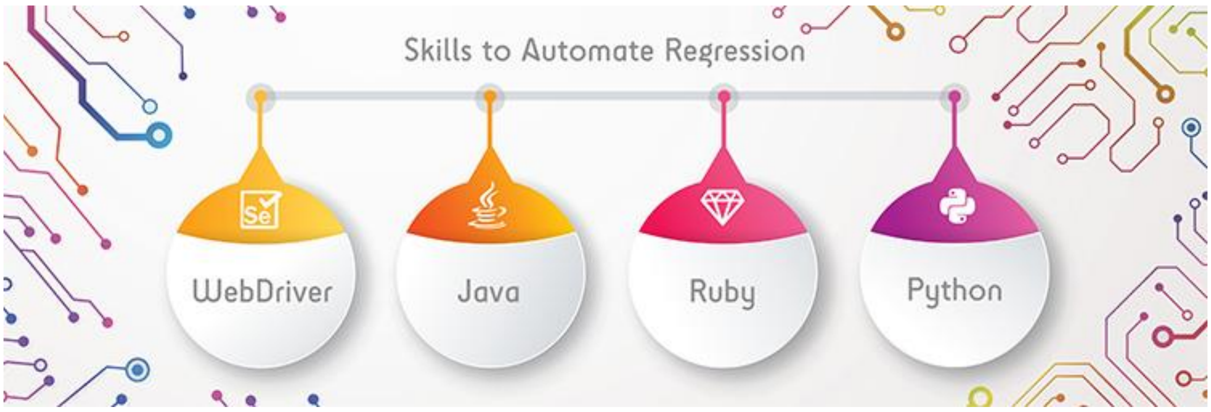
Any tester wishing to know how to write automated UI tests will need to be familiar with HTML/CSS/JavaScript. Understanding these languages, and how they work together, enables a software tester to learn more about how a web application is built.

Knowing HTML and CSS helps testers to interpret code from a browser console as they investigate a page. It is a good starting point as HTML is the foundation behind all web pages. It's used to add structure, text, images, and other types of media. CSS is the language used to style HTML content to create visually appealing web pages.

Knowing JavaScript is also helpful because it can be executed from within Selenium scripts, if needed.

Let's say you are a software tester and you are testing a form submission on a webpage. When you submit the form, you see an error. A tester who knows HTML/CSS/JS can open the development tools in their browser, select the console option, and repeat their actions to reproduce the error. In the console, they will be able to see the JavaScript error that is thrown, and then use this information to either further investigate the issue or make a thorough bug report to developers.

Skills to Automate Regression: WebDriver, Ruby, Java, Python



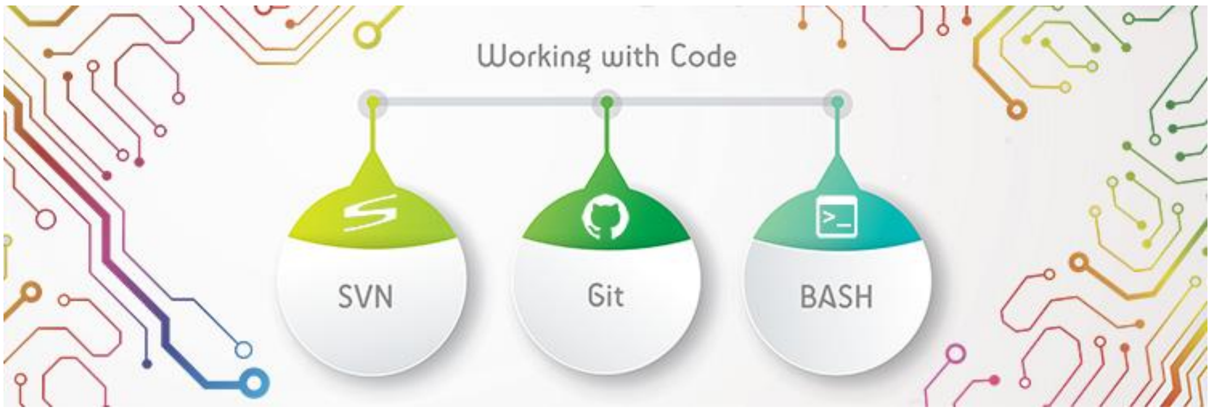
More often, testers are being asked to create automated UI tests for web applications. While there are many options, Selenium WebDriver (WebDriver) tends to be the most popular API for driving a browser. Ruby, Python, and Java are popular language choices for people wishing to work with Gherkin style syntax and WebDriver. Another reason these languages tend to be popular is that many web apps use them in other capacities, making it easier for the tester to get help from developers in creating and maintaining the test suite.

For example, a tester may be asked to design an end-to-end regression test of a website that includes logging into the site. Using HTML and CSS skills, the tester can open the browser console elements, locate the login box on the page, and inspect the element to find the id of the element. The tester can then use Ruby, Java, or Python to write commands for the WebDriver API. Some sample code in Python might look like this:

```
driver.get("https://www.website.com")
element = driver.find_element_by_id("login-id")
element.send_keys("yourname")
```

The above code goes to the website, locates the textbox that accepts the login, and then enters the login name into the text box. Learning the WebDriver API along with a language like Ruby, Java, or Python are key skills for building automated regression tests.

Working with Code: SVN, Git, and BASH



Sometimes, a tester is lucky enough to work in a large or mature company where they can develop their tests against an environment that just "exists". More typically, however, some environment setup is required. Also, testers who are writing UI automation will need to check in their own code. For maximum work independence and flexibility, testers should know a version control system also. Two popular version control systems are SVN and Git. Both SVN and Git can easily be learned online with free resources.

In addition, testers should get to know the command prompt/command line and some simple Bash Commands to move around both their computer and their code base. Bash Commands can help the tester navigate files easily while writing tests. Additionally, it is generally necessary to use the command prompt to run automated UI tests and investigate the results of failed scenarios or features.

For example, a tester might be starting a job at a company whose code is kept in GitHub. They can do a 'git clone' command to get the code onto the computer. Bash Commands would help the tester move files around, save changes to his test code, and push test changes up to Git for review. Some sample code might look like this:

`git clone` (gets a copy of the repository)
`cd` (bash command "cd" means "change directory" to your company's repository)
`git checkout -b` (makes a local copy of the code just for you)
`git add .` (save your changes)
`git push origin/your_shiny_new_branch` (sends your branch to the repository)

There are, of course, many tech skills beyond this that software testers can learn, and there are no rules about where to start. Different companies can and will employ different technologies. However, these top tech skills can help software testers build a core understanding of web technologies that they can use to branch out into more varied technologies during their careers.

Places to Learn Programming Skills Online for Free

Find out more at TeachCode.Org.

HTML and CSS
Learn HTML and CSS - This course takes approximately 10 hours to complete.

JavaScript
Learn JavaScript - This course will teach you the most fundamental concepts in programming JavaScript. It takes approximately 10 hours to complete.

Computer Programming - Learn the basics, starting with Intro to programming. With the Khan Academy, you can learn how to use the JavaScript language and the Processing.js library to create fun drawings and animations. There are also courses available that will enable you to combine HTML and JS for interactive webpages.

WebDriver
Online Selenium Tutorial for beginners in Java - Learn Selenium WebDriver automation step by step hands-on practical examples.

Ruby
Learn Ruby - In this course you can gain familiarity in Ruby around basic programming concepts, including variables, loops, control flow, and object-oriented programming. You will also get the opportunity to test your understanding in a final project which you'll build locally. The course takes approximately 9 hours.

Java
Learn Java - In this course you'll learn fundamental programming concepts, including object-oriented programming in Java. You will also get the opportunity to build 7 Java projects, like a basic calculator, to help you practice along the way. This course takes approximately 4 hours to complete.

Python
Learn Python - This course is a great introduction to both fundamental programming concepts and the Python programming language. It takes approximately 13 hours.

Learn Python the Hard Way - This is the 3rd Edition of Learn Python the Hard Way. You can visit the companion site to the book at http://learnpythonthehardway.org/ where you can purchase digital downloads and paper versions of the book. The free HTML version of the book is available here.

Command Line
Learn the Command Line - Learning to use the Command Line will help you to discover all that your computer is capable of and accomplish a wider set of tasks more effectively and efficiently. This course takes approximately 3 hours.

SVN
Learn SVN - Apache Subversion which is often abbreviated as SVN, is a software versioning and revision control system distributed under an open source license.

Git
Learn Git - This course teaches you to save and manage different versions of code projects. It takes approximately 2 hours to complete.

Written by Jess Ingrassellino. *Jess is a software engineer in New York. She has perused interests in music, writing, teaching, technology, art and philosophy. She is the founder of TeachCode.org*

Like the Content? Want More?
Sign up for our weekly Testers Email here on our blog!

SIGN UP HERE!

30 Days of Agile Testing

Ready to ramp-up the Agility some more? Follow the 30-day plan below to start shedding waste from your development and testing processes. Share your pains, gains, wins and fails with the software testing community by live tweeting your experiences, remembering to include the #30DaysOfTesting hashtag!

- ☐ **1** Buy an agile testing related book and share something you've learnt by day 30
- ☐ **2** Create a mindmap, document, diagram or sketchnote about what you think agile testing is
- ☐ **3** Find a video on YouTube about agile testing, then watch it!
- ☐ **4** Read the agile manifesto and reflect on the implications for your role
- ☐ **5** Pair with a developer on a feature
- ☐ **6** Map out what your exploratory testing looks like, compare it to what other testers do
- ☐ **7** Find a visual way of representing your tests - e.g. a mind map, diagram, model, etc.
- ☐ **8** Speak to a developer about a bug you found instead of logging it in the tracking system
- ☐ **9** Pair with a developer on a code review. Can you identify any risks?
- ☐ **10** Learn where the application logs are and how to read them
- ☐ **11** Find out what customers are saying about your product. What did you learn?
- ☐ **12** What test documentation does your team have? How can you improve it?
- ☐ **13** Learn to use a tool that your developers use - e.g. an IDE
- ☐ **14** How can you deliver greater value to your customer?
- ☐ **15** How can you make your testing processes (more) lean?
- ☐ **16** What barriers do you feel exist in testing in agile?
- ☐ **17** Map out your current team structure. How does it compare to other teams?
- ☐ **18** How can you make testing jobs easier?
- ☐ **19** How can you make jobs for your team easier?
- ☐ **20** Investigate what is in your and your team's tool kit
- ☐ **21** How are you managing your testing, is it really agile?

- ☐ **22** Find out what testing is being done by other team members
- ☐ **23** What agile strategies are there for managing tests? How can you improve yours?
- ☐ **24** Look for a task that can be automated
- ☐ **25** What can't you automate? Communicate that to your team
- ☐ **26** What does your Test Plan look like, what format do you use?
- ☐ **27** Look into zero bug tolerance, is this something your team could do?
- ☐ **28** What learning culture exists in your company? How can you contribute to it?
- ☐ **29** What columns do you have on your work tracker or kanban board?
- ☐ **30** What action does your team take on a red build?
- ☐ **31** BONUS: Debrief your whole team on your last session of testing

www.ministryoftesting.com

Sponsored by:
gurock.com/testrail

TestRail

By Rosie Sherry and Simon Knight

Stay in Touch!

Make sure you keep up to date with the latest news, learning and events in the Software Testing community by checking out our blog posts and articles on the TestRail blog!

Like the Content? Want More?
Sign up for our weekly Testers Email here on our blog!

SIGN UP HERE!

TestRail

http://www.gurock.com/testrail