1. New tools:
    - Real Time Error Monitoring Tool: Raygun
    - Continuous Integration Tool: Jenkins
2. Notable Features:
   Raygun:
    - See the exact line of code that caused an error
    - Monitor deployments to determine what caused a spike or decrease in error count
    - Full stack trace information and diagnostic details for every error occurrence or crash
    - Filter through your errors by date, time, version, tag, host, OS, browser, custom tags, and more
    - Reduce noise with configurable filters for machine name, version, IP address, hostname, and more
    - Groups errors by affected users (this also helps reduce noise)
    - 180-day data retention
    - Support for all major languages and frameworks
    - Easy setup using lightweight SDKs
    - Works seamlessly alongside Real User Monitoring and APM for full visibility into your users digital experience.
    - Supports JS framework and integrates with github which works well for our purposes.

   Jenkins:
    - Jenkins is open-source and free to use. It is a preferred CI/CD tool by early-stage startups as well as large corporations since it has been under development since a long time.
    - It is a rich plugin ecosystem. At the time of this article, there were close to 1500+ plugins available for use.
    - Jenkins can be integrated with popular cloud platforms such as Amazon EC2, Google Cloud, VMWare vSphere, Digital Ocean, and more.
    - Jenkins Pipelines can be extremely useful for realizing CD requirements for large-scale projects.
    - It can be installed easily on any operating systems4. User friendly interface that is easy to configure and with easy upgrades.
3. Getting Started Instructions
   Raygun:
    - For Raygun, I was unable to find a very useful official tutorial published by the developers themselves. However, I was able to find this website, https://code.tutsplus.com/tutorials/getting-started-with-raygun-insights-and

-crash-reporting-for-app-developers--cms-24852. This website provides a very useful tutorial on how to install and begin using raygun, including example code for how to utilize it with Javascript. This would be very useful for us to use Raygun for our purposes. I also found this page, https://raygun.com/documentation/product-guides/crash-reporting/integrations/github/ which provides documentation on how to successfully connect Raygun to your github repository.

Jenkins:

- For Jenkins, I found this very thorough and in depth resource which provides several insights and tutorials into Jenkins, https://www.lambdatest.com/blog/what-is-jenkins/. This includes multiple tutorials on how to install and configure Jenkins and would be very useful. The official Jenkins website does have a page specifically for listing user documentation, and provides a full library on how to perform various functions using Jenkins.

4. Date and Market Share of the product:

Raygun:

- For Raygun the first commit I was able to find was in 2014. On their website they claim over 100,000 developers are currently using Raygun. They also brag that among their customers are companies such as Coke, HBO, and Microsoft.

Jenkins:

- For Jenkins the fist commit that I was able to find was in 2013. According to a 3rd party website that I was able to find, Jenkins is the oldest player in the industry and commands a market share of 71%. On the website for Jenkins, they do not brag about specific companies that utilize their product, however, they do list a series of industry awards that they have won in recent years.

Step 2: Runtime Analysis

Array Size	**Insert Method**	**Append Method**
extraLargeArray	2.0644878 s	7.3727 ms
largeArray	17.5852 ms	989.9 µs
mediumArray	360.2 µs	315.7 µs
smallArray	85.3 µs	191.3 µs
tinyArray	67.1 µs	184.4 µs

The insert function scales with much worse time complexity. Although it initially executes faster when the arrays are smaller, as the array scales in size, the performance of this method slows down massively. The append method scales much better. When changing the size of the array from 1000 to 10000, the append method scales by a factor of roughly 3. This is much faster than the equivalent method for the insert method which scales by a factor of roughly 54.

Bonus Question: Looking into the time complexity of the two array methods we used, (unshift and push), reveals that the methods themselves have default time complexities. For push, this is O(1), or a constant time complexity. For unshift, this is O(n) or a linear time complexity. However in our code, both of these are called in for loops where the length is determined by the array size, a loop with a time complexity of O(n). This essentially multiplies both of our existing time complexities by n, which leaves us with push having O(1*n) or O(n) a linear time complexity, and unshift with O(n*n), or a quadratic time complexity. These results make sense when we examine the way these methods scale and the shape of their curves.