Alternative Tools Research

Jenkins.io | https://www.jenkins.io/

Pros

- Integrates with existing tools
- Built by developers
- Continuous Integration
    - Rather than waiting for “nightly builds”, developers are able to test their code with other teams code in real time
- Customers get new code faster because it is always being updated and always being tested
- Open Source
- Millions of users (one of the most popular tools @ 165k active installations)
- Because it has millions of users, it has a phenomenal support team
- Built in functionality to connect with repos and has build tasks for Maven, Git, Puppet, Nagios, Ansible etc

Jenkins was started in 2004 and is an open source tool with *millions* of users, phenomenal support and is consistently updating to keep up with today's requirements. Jenkins is easy to set up - done in just 3 steps - 1)download the tool from the website 2) deploy it 3) install necessary plugins.

Market Share - 71%
Active Development? - Yes
Earliest Commit - 14 yrs ago
Companies using Jenkins

- eBay
- Etsy
- Facebook
- GitHub
- Intuit

- LinkedIn

Atlassian Bamboo | https://www.atlassian.com/software/bamboo

Pros

- Runs multiple builds in parallel for faster compilation
- Build in functionality to connect with repos and has build tasks for Ant, Maven etc
- Designed to make build flow easy to setup
- Allows devs to use CI/CD methodologies
- Built in Git branching workflows
- User-friendly

Atlassian Bamboo has been around since 2002, while not nearly as possible as Jenkins. In fact, some have said that Atlassian Bamboo is dead, though they continue to serve their customers today. Atlassian Bamboo takes a few more steps than Jenkins to setup - after installing Java, one must select a dedicated user to run Bamboo, then download Bamboo from the website to create an installation directly and after setting up your user page, you are then able to configure Atlassian Bamboo to your needs.

Market Share - 1.3%
Active Development? Yes
Earliest Commit - Hard to find
Companies using Atlassian Bamboo

- Honeywell
- Penske
- Nike
- JPMorgan Chase

extraLargeArray results

- Insert - 953.0792 ms
- Append - 2.964 ms

doubTimer

tinyArray	smallArray	medArray	lgArray	xlArray
doub timer: 0.119ms	doub timer: 0.243ms	doub timer: 0.361ms	doub timer: 0.305ms	doub timer: 0.192ms

getSizedTimer

tinyArray	smallArray	medArray	lgArray	xlArray
gsTimer: 0.183ms	gsTimer: 0.257ms	gsTimer: 0.307ms	gsTimer: 7.345ms	gsTimer: 39.269ms

doublerInsert

tinyArray	smallArray	medArray	lgArray	xlArray
doub timer: 0.254ms	doub timer: 0.336ms	doub timer: 0.488ms	doub timer: 0.212ms	doub timer: 0.159ms

doubTimer Explanation

I believe that JavaScript realizes over time that it continues to do the same thing, and it acts on almost “muscle memory”. The information stays the same that it is collecting / the actions stay the same that it is performing, it is just doing them more and more as the array grows.

getSizedTimer Explanation

I actually ran this one backwards - starting with the xlArray. It clearly dropped immensely as the arrays became smaller, so I ran it a second time going from tinyArray to xlArray and I received similar results.

doublerInsert Explanation

I realized afterwards that I was supposed to do both of the ‘doubler’ functions, but this was actually awesome to see. While I thought that the doublerAppend was the most efficient, it's clear to see that the doublerInserts function is even more efficient. I believe that this is because unshift is only changing the beginning of the array, so shifting everything over. Rather than push where an item is then added to the end of an array.

Summary

It is clear to see that the doublerInsert function is the most efficient as it has the best times and continues to get better and better. I feel that this is because it is using ‘unshift’ which adds items to the beginning of the array and shifts everything else over one index. I could be wrong but I feel that ‘push’ is slower because it has to find the end of the array, then move each item behind it? I know how push works but I would imagine that this is why it is slower.