There was no puff of white smoke, no final decisive election, But behind closed doors, in a conference in Budapest, key internet engineers discussed the failings and the future of JavaScript.

Developers argue the demands once placed on languages like C/C++ and Java now shift to JavaScript, exposing limitations in performance and maintainability. The latest generation of JavaScript engines may deliver incredible performance gains, but to many these still aren't enough when you look at how sites such as YouTube, Facebook, EA' etc have to scale up.

Prezi, LogMeIn and Ustream organized a conference in Budapest to focus on how to scale JavaScript, MLOC.js. Engineers from Google, Facebook, Mozilla, Groupon and others gathered for a 3-day tete-a-tete to discuss how to work-around the limitations of JavaScript. Their work could change the way we write applications on the web.

Péter Halácsy, co-founder and CTO of Prezi, writes in to *TechCrunch* with this critical point:

"Here's my takeaway: JavaScript is caught in an iron triangle of its own making. We've all grown accustomed to the flexibility and freedom it offers, and we don't want to give that up. At the same time, we want it to be blazing fast, and it turns out that when you reach millions of lines of

JavaScript, maintaining such a flexible language becomes a problem. All three of these areas-- flexibility, performance, and maintainability-- are intertwined. When you make progress in one area, one of the other two, or even both, will suffer.

This tells me that there is no one silver bullet that will solve these problems, but there may be many that do. JavaScript is a dynamic language that excels at being the glue of the web, and that shouldn’t change."

Sure, it's dynamic but if companies still turn to Flash for performance because JavaScript can't match its performance-- maybe, just maybe the world is waiting for another solution. The Next Language...

RedMonk's list of most popular programming languages...

1. JavaScript
2. Java
3. PHP
4. Python
5. Ruby
6. C#
7. C++
8. C
9. Objective-C
10. Shell
11. Perl
12. Scala
13. Haskell
14. ASP
15. Assembly
16. ActionScript
17. R
18. Visual Basic
19. CoffeeScript
20. Groovy

Is the answer on this list? Or still coming? While companies like Google have a massive investment in JavaScript, they don't seem to have a strong commitment.

Every once in a while, some visionary emerges to proclaim: "I'm going to kill JavaScript. Just you watch." GWT tried, and failed. And now DART is here, another knight-errant come to slay the JavaScript giant. It looks like the DART team is just ignoring JavaScript altogether and assuming if they build a better language, people will just drop everything and start using it. Wish them luck-- but we all know how that movie ends.

It's easy enough to think Google is caught up in the pipe dream that JavaScript will magically vanish someday, as if the Web itself would vanish. JavaScript will outlive us all.

For example, Steve Yeggie, an experienced software developer in America well-known for his blogs, says "I think JavaScript will evolve, slowly. It's in much the same boat as C++, with a bunch of competing vendors needing to agree before any progress can be made. But C++ still managed to squeeze out a major upgrade with C++11, so there's hope. I think JavaScript will eventually acquire a type system of sorts, possibly learning some lessons from DART along the way. The lack of a static type system is the heart of the issue. It's the reason the world bashes on JavaScript, but it's also the reason the worldwide JavaScript community ignores Java-based or even Java-like approaches (including Google's Closure library). They obviously feel that a type system like Java's isn't worth the price of admission. DART's optional-types approach seems like the best bipartisan compromise. So even if DART fails, I think it could still have an impact."

Go The Death of JavaScript?