The State and Future of JavaScript

Douglas Crockford
Yahoo!

The State and Future of ECMAScript

Douglas Crockford
Yahoo!

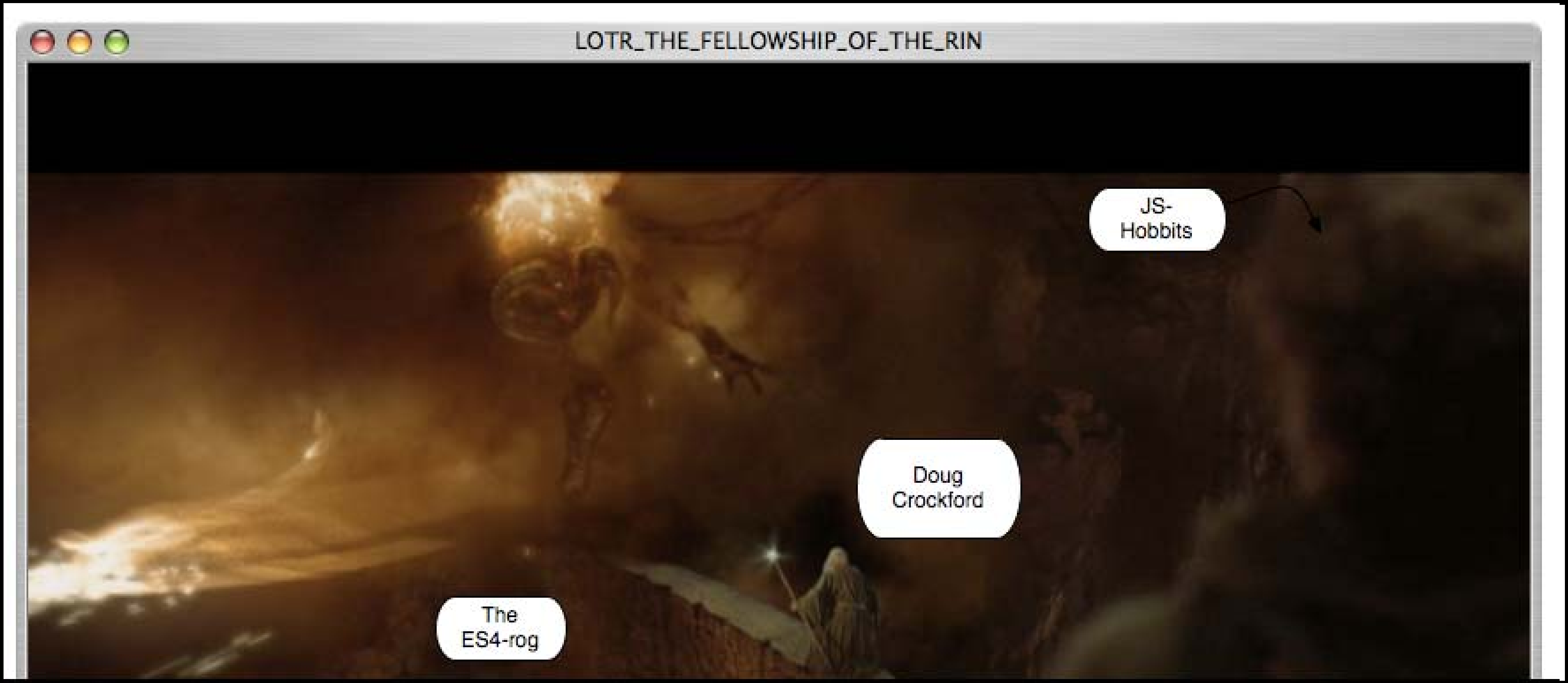
The World’s Most Popular Programming Language

The World's Most Popular Programming Language

The World's Most Unpopular Programming Language

1999 and Beyond!

The Fellowship of the Ring



12 Angry Men

ECMA TC39

- At the time that I joined, there were no members who used the language.
- There were only 4 other active members: Adobe, Mozilla, Opera, and Microsoft.
- Opera was not actually a member of ECMA, so their participation was in violation of ECMA rules.
- The committee was pursuing ES4, a design that began at Netscape in 1999.
- It had been abandoned, but restarted because of interest in Ajax.

I wasn’t convinced

- The new language didn’t solve any of our problems.
- There was strong interest in the web community to make the language suck less.
- I was concerned that it was going to suck more.
- Good intentions have been proven to be ineffective.

They didn’t want to hear it

- Refused to argue about the system as a whole.
- Refused to argue about the necessity of individual features.
- They had been at work for many years.
- They would be done next year.

- I was too late.

However...

- It turned out that the Microsoft member had similar concerns.
- But he was also concerned that if Microsoft stood up that it would be accused to anticompetitive behavior. It turned out that this concern was well founded.
- I convinced him that Microsoft should do the right thing.

The showdown in Redmond

- Microsoft wanted to play hardball, setting up a paper trail, beginning grievance procedures…
- I insisted that we keep it technical.
- Microsoft formally refused to accept ES4 in whole or in part.
- The committee was not in consensus.

ES4 v ES3.1

- I proposed an ES3.1 project, a minimalist's alternative to ES4.
- The argument went public.
- ES4 was positioned as the official new standard, and that the ES3.1 project was an unfair attempt to subvert the standards process.
- Yahoo was accused of conspiring with Microsoft to protect IE's dominance.

More Jurors

- I tried to encourage other companies to participate.
- Many sat on the side line, and told me privately that they were glad I was fighting the fight, but they did not want to get involved because they did not want to be attacked like Microsoft.

Meanwhile…

- We agreed to disagree.
- We informally split into two committees, but continued to hold joint meetings.
- It was very uncomfortable and unproductive.
- Eventually other companies joined: IBM, Google, Apple, Opera, Dojo, and Company 100.

ECMA was worried

- ECMA could not tolerate two proposals.
- ECMA demanded consensus.
- The ECMA Secretary General and the ECMA President began attending our meetings.
- We were able to agree on a subset relationship between ES3.1 and ES4, but we could not agree on what that meant.
  - ES4: The ES3.1 could only pick features from ES4.
  - ES3.1: ES4 had to adopt everything that ES3.1 adopted.

Later…

- ES4 was not converging. It slipped a year per year since the project began in 1999.
- ES4 began jettisoning features in an attempt to get back on schedule.
- Ultimately, the project was abandoned.
- ES3.1 was completed and became the candidate for ECMAScript Fifth Edition.
- It goes before the General Assembly in December.

One company has stated its intention to vote against it.

Company: IBM

Issue: Decimal arithmetic

0.1 + 0.2 !== 0.3

The most replicated bug.

IEEE 754

- Improved uniformity of floating point results between different machine architectures.

- Well suited for applications using very large and very small numbers, including astronomy, chemistry, and physics.

Binary floating point is poorly suited for most other applications.

Including everything that we do.

Binary floating point cannot exactly represent decimal fractions.

It can handle dollars and quarters, but accumulates errors on pennies, nickels, and dimes.

Given the applications of ECMAScript, IEEE 754 turns out to have been a bad choice.

A bad but popular choice that was adopted by virtually every programming language designed in the 90s and 00s.

Just because something is a standard doesn't mean it is the right choice for every application.

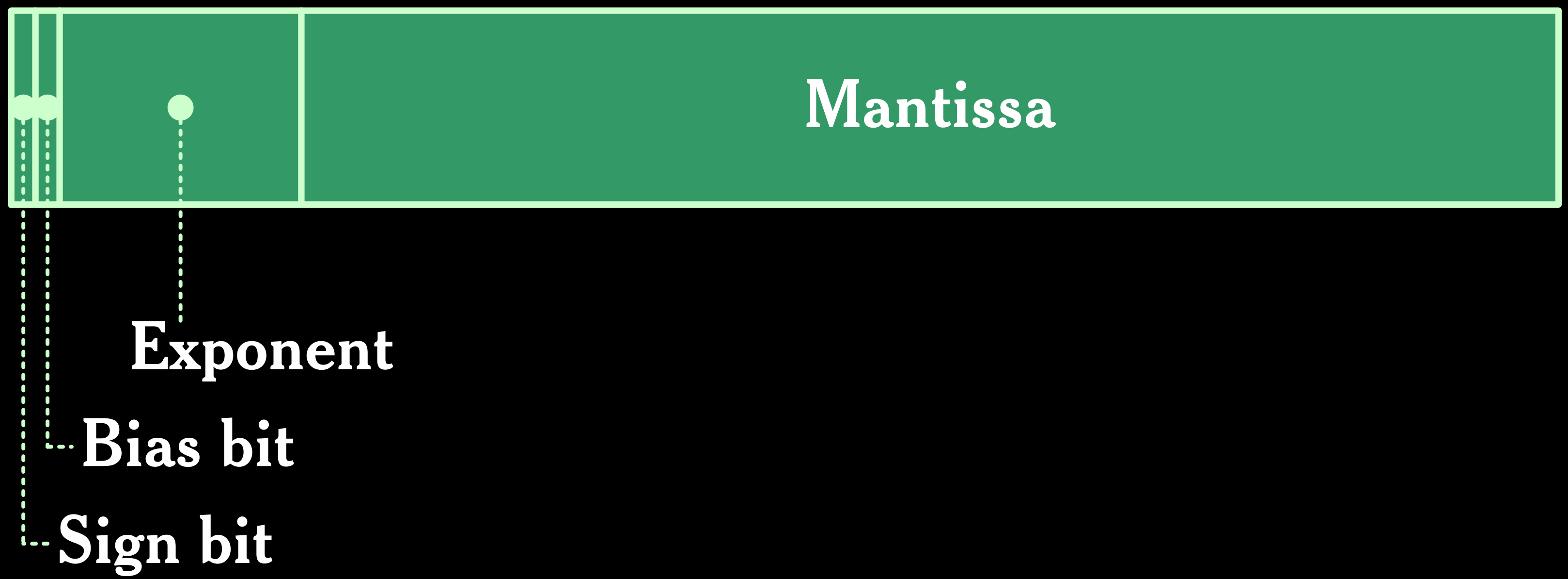
Like XML, for example.

IBM is promoting IEEE 754r

- It adds decimal modes to IEE 754.
- It corrects the problems with representation of decimal fractions.
- IBM tried to get it into the Third Edition (1999).
- TC39 decided to defer it until the Fourth Edition.

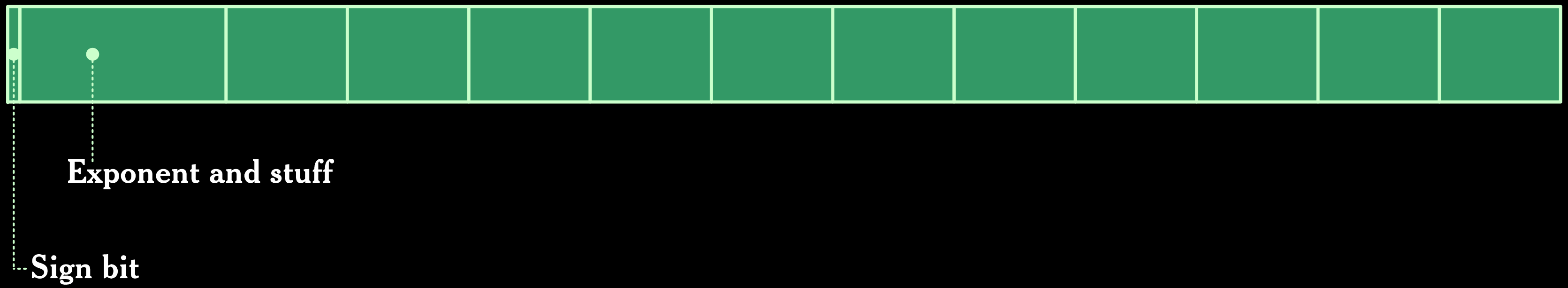
- It was ultimately rejected by ES4 and ES3.1.
- This was one of the few things we could agree on.

IEEE 754



$$\text{Number} = \text{Mantissa} * 2^{(\text{Bias} - \text{Exponent})}$$

IEEE 754r DEC128



$$\text{Number} = \text{Mantissa} * 10^{\text{Exponent}}$$

It will produce results that are more correct, but some applications may be dependent on incorrect results.

This is the way of the web.

We considered adding a `"use decimal";` pragma.

This was rejected because the new format is hundreds of times slower than the old one.

We considered adding a decimal library.

It was ugly, unlikely to solve any real problems in actual use, and introduced mixed type problems.

We considered adding another number type.

Changing the number of number types from 1 to 2 is a violent change. This may ultimately be the right approach, but it will take a long time to get right.

TC39 is looking seriously at decimal for a later edition.

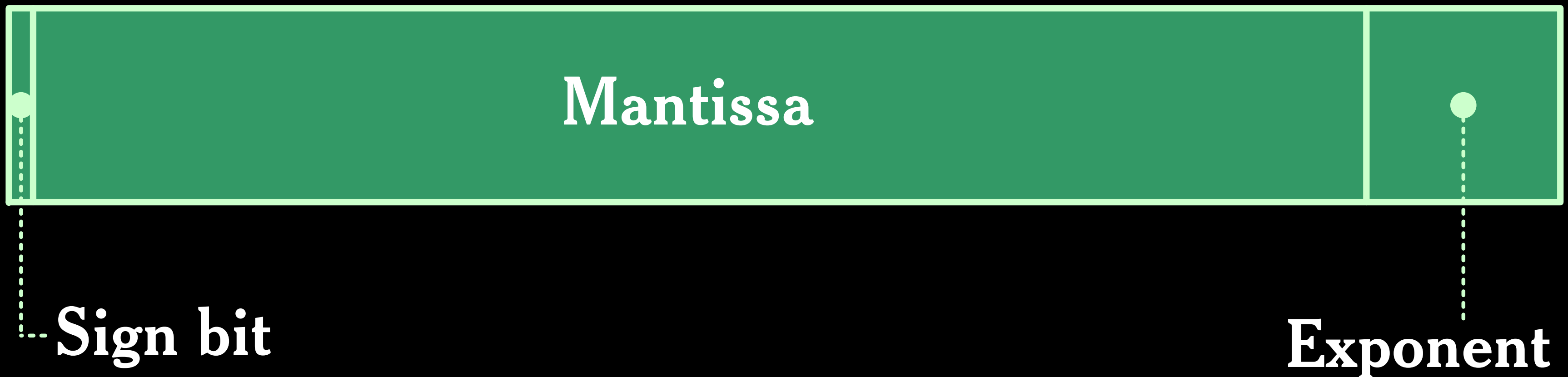
There was been no discussion of selection of a more suitable decimal format.

IBM: IEEE 754r DEC128
or nothing.

9E6

- Numbers are 64 bits, scaled by 9 million.
- Advantages:
  - Addition and subtraction at integer speeds.
  - 6 exact decimal digits
  - A repeating 7th decimal digit (*n* / 9000000)
- Disadvantages:
  - 6 or 7 decimal places may not be enough for some applications
  - 1 trillion is too small for some applications

DEC64



$$\text{Number} = \text{Mantissa} * 10^{\text{Exponent}}$$

DEC64

- Advantages
  - Much faster than IEEE754r DEC128
  - Easy to implement in software
- Disadvantages
  - Slower than 9E6
  - Range is only around 10^{143}

TC39's goal is to repair the language, not to add IEEE 754r.

IBM's goal is to add IEEE 754r, not to repair the language.

IBM has stated that they will vote against any language standard that does not include IEEE 754r.

Appeal to IBM

- It is irresponsible to inflict damage on a language and its community to solve an unrelated chicken/egg problem.
- If your nay vote fails, then you will have done nothing except show contempt for the web development community.
- If your nay vote succeeds, then you will cause significant damage to the web and the open standards movement.
- IBM, please vote aye.

Fifth Edition

- The Fifth Edition defines two languages:
  - The Default Language
  - The Strict Language
- You should use the ES5/Strict for reliability, or the ES3 for compatibility.
- Do not write in the ES5/Default.
  It will be abandoned soon.

Harmony

- The code name of the next proposal is Harmony, not ES6.
- We want to avoid giving proposals edition numbers because it gives the false appearance of inevitability or momentum.
- Harmony will be built on the Strict Language.
- Harmony will probably have incompatible syntax, so programs written in the Harmony language will fail on all pre-Harmony browsers. Hopefully the IE6 problem will be gone by the time our work is done.

Design Pressure

- To suck less.
- To be more like the other languages.
- To be more expressive.
- To be a better compilation target.
- To be more secure.
- To be better at math.

Correct the {block scope} problem.

New `let` and `const` statements to replace `var`.

Better support for variadic functions

The `arguments` array has lots of problems.

Syntactic Sugar

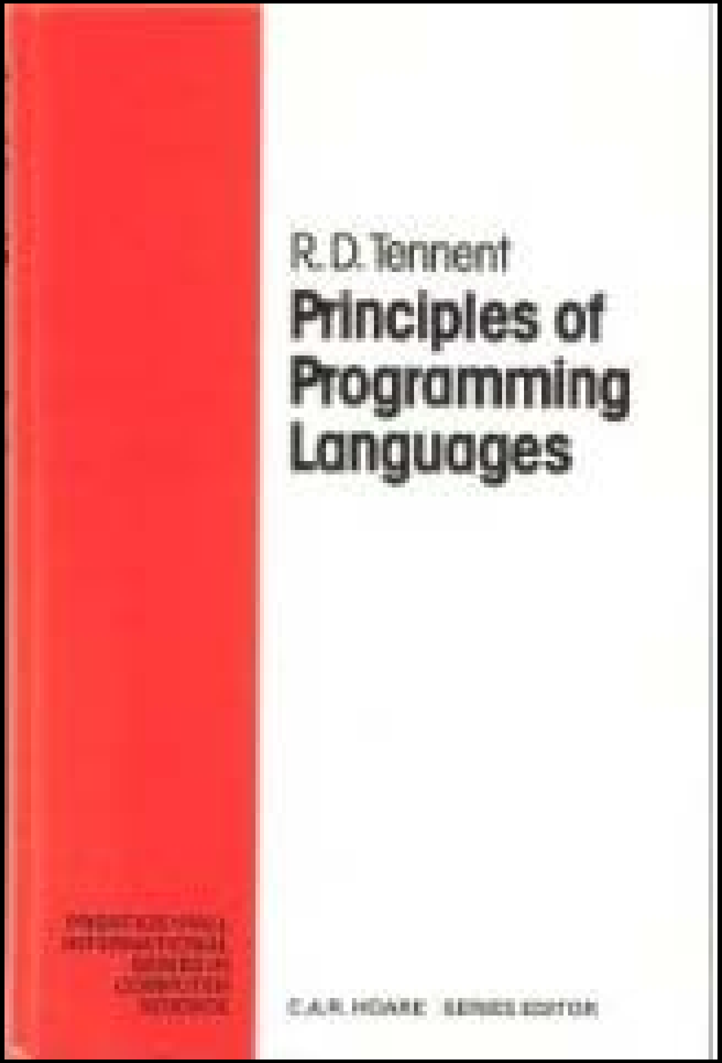
- We might be adding things like classes as sugar on top of the existing language.
- This could be done with macros, but TC39 is wary of macros.

“Syntactic sugar causes cancer of the semicolons” – Alan Perlis

- Functions are attractive (classes as closures), but there is a problem.

Tennent's Principle of Correspondence

- R. D. Tennent
- *Principles of Programming Languages*
- 1981



R. D. Tennent
Principles of Programming Languages

[illegible]

C.A.R. HOARE SERIES EDITOR

Tennent's Principle of Correspondence

```
function booga() {
    var x = 3,
        y = 4;
    return x + y;
}

function wooga() {
    return (function (x, y) {
        return x + y;
    }(3, 4));
}
```

Tennent's Principle of Correspondence

- Any expression or block can be placed in an immediate function.
- Except: implied parameters (`this`, `arguments`) and `var` and disruptive statements (`return`, `break`, `continue`, `throw`).
- Tennent's book does not demand that a language have full correspondence.

Tennent’s Principle of Correspondence

- **Many existing languages only managed to combine countless “features” into a jumble that is neither easy to implement nor a pleasure to use.**
- **Side effects are often confusing to program readers because they are unexpected: the familiar expressions of conventional arithmetic and algebra do not have side effects.**

Ten Years

is too long between editions.

The next edition is planned for 2.5 years, but will probably take longer, because it always does.

Lessons

A clear separation between research and standard setting

A standard is the last place where you want to see innovation.

**If you have a great new idea,
don’t tell it to a standards body.**

**Instead, implement it,
and then show it to the world.**

Don’t promise what you can’t deliver.

At this point, The Fifth Edition is just a candidate, which may fail.

A change to a widely used standard is an act of violence

Any changes had better be worth it.

Standards are hard.

Tremendous care and precision.

You can’t please everybody.

Check your expectations.

The process can produce heartbreak and disappointment.

Check your motives.

Patents and Open Standards Are Incompatible.

It is time to close the Patent Office.

The success of an enterprise should depend on the quality of its goods and services and its ability to execute efficiently, and not on a capricious government office.

The Patent System made sense in the 18th and 19th Centuries.

It has long outlived its usefulness.

Nobody wins except the lawyers.

Getting a lot of people to agree on anything, particularly on something good, is a wonderful and valuable thing.

ECMAScript: The Fifth Edition. The Best Web Standard in the History of the World!

There is still lots of room for improvement.

I expect that the future editions will be even better.

I expect that the future editions will be even better.

Even if they still suck a little.