

YAHOO!

CROCKFORD ON JAVASCRIPT

DOUGLAS CROCKFORD, JAVASCRIPT ARCHITECT, YAHOO! INC.



YAHOO!

CROCKFORD ON JAVASCRIPT

DOUGLAS CROCKFORD, JAVASCRIPT ARCHITECT, YAHOO! INC.

Part 5

The End of All Things

**I believe that children are our
future.**

**I believe that children are our
future.**

And also robots.

Children and robots.

The Browser and Security

XSS and Beyond

**What can an attacker do if he
gets some script into your
page?**

**An attacker can request
additional scripts from any server
in the world.**

**Once it gets a foothold, it can
obtain all of the scripts it needs.**

**An attacker can read the
document.**

**The attacker can see everything
the user sees.**

**An attacker can make requests
of your server.**

**Your server cannot detect that the
request did not originate with your
application.**

If your server accepts SQL queries, then the attacker gets access to your database.

**SQL was optimized for
SQL Injection Attacks**

An attacker has control over the display and can request information from the user.

The user cannot detect that the request did not originate with your application.

**An attacker can send information
to servers anywhere in the world.**

**The browser does not prevent
any of these.**

**Web standards require these
weaknesses.**

**The consequences of a successful
attack are horrible.**

Harm to customers.

Loss of trust.

Legal liabilities.

HTML5

A big step in the wrong direction.™

**Cross site scripting attacks
were invented in 1995.**

**We made no progress on the
fundamental problems except for a
baby step in December 2009.**

Why is there XSS?

- **The web stack is too complicated.**
Too many languages, each with its own encoding, quoting, commenting, and escapement conventions.
Each can be nested inside of each other.
Browsers do heroic things to make sense of malformed content.
- **Template-based web frameworks are optimized for XSS injection.**

Why is there XSS?

- The JavaScript global object gives every scrap of script the same set of powerful capabilities.
- As bad as it is at security, the browser is a vast improvement over everything else.
- The browser does distinguish between the interests of the user and the interests of the site.
- The browser failed to anticipate that there could be additional interests.

And then there is the Mashup Problem

- *A mashup* is the combining of programs representing multiple interests.
- The browser confuses those interests.
- So an advertiser on a page gets the same privileges as an Ajax library or an analytics files, which is the same as the main applications, which is the same as any XSS code that falls into the page.
- Advertising is a self-inflicted XSS attack.

Safe JavaScript Subsets

Deny access to the global object and the DOM.

Caja. <http://code.google.com/p/google-caja/>

ADsafe. <http://www.ADsafesafe.org/>



ECMAScript Fifth Edition Strict

December 2009

ES5/Strict makes it possible to have static verification of third party code without over-constraining the programming model.

The best of both Caja and ADsafe.

There is still more work to be done

- ES5/Strict can protect the page from the widgets, but it cannot protect the widgets from the page.
- ES5/Strict cannot protect the page from XSS script injection.
- ES5/Strict is an important step toward a programming model in which multiple interests can cooperate for the user's benefit without compromising each other or the user.

**The browser makers are now
in a race to implement
ES5/Strict.**

**On the day when all 5 major
browsers support ES5/Strict...**

IE6
MUST
DIE!

Many popular approaches to security fail.

- Security by inconvenience. (TSA)
- Security by obscurity.
- Security as identity.
- Security by vigilance.

Working with the grain.

Pass By Reference

When values are immutable,
Pass By Reference looks like
Pass By Value

By reference

```
variable = {};
```

```
a = variable;
```

```
b = variable;
```

```
alert(a === b);
```

**Don't confuse values with
variables.**

```
function funky(o) {  
    o = null;  
}
```

```
var x = {};
```

```
funky(x);
```

```
alert(x);
```

```
function swap(a, b) {  
    var t = a;  
    a = b;  
    b = t  
}  
var x = 1, y = 2;  
swap(x, y);  
  
alert(x);
```

Confusion of values and variables

```
function do_it(inputs, name) {  
    ...  
    eval(name + ' = ' + result);  
    window[name] = result;  
}
```

```
function do_it(inputs, obj, name) {  
    ...  
    obj[name] = result;  
}
```

```
function do_it(inputs, callback) {  
    ...  
    callback(result);  
}
```

```
function do_it(inputs, callback) {  
    ...  
    callback(result);  
}
```

```
do_it(my_inputs, function (result) {  
    my_object.blah = result;  
});
```

```
function storer(obj, name) {  
    return function (result) {  
        obj[name] = result;  
    };  
}
```

```
do_it(my_inputs, storer(my_object, 'blah'));
```

```
function storer_maker(obj) {  
    return function(name) {  
        return function (result) {  
            obj[name] = result;  
        };  
    };  
}
```

```
my_storer = storer_maker(my_object);  
do_it(my_inputs, my_storer('blah'));
```

```
function once(func) {  
    return function () {  
        var f = func;  
        func = null;  
        return f.apply(this, arguments);  
    };  
}
```

```
do_it(my_inputs,  
      once(storer(my_object, 'blah')));
```

```
function revoker(func) {  
  return {  
    revocable: function () {  
      return func.apply(this,  
        arguments);  
    },  
    revoke: function () {  
      func = null;  
    }  
  };  
}
```

```
function sequence() {  
    var functions = arguments;  
    return function () {  
        var args = arguments;  
        functions.forEach(  
            function (func) {  
                func.apply(null, args);  
            });  
    };  
}
```

```
do_it(my_inputs, once(sequence(  
    storer(my_object, 'blah'),  
    storer(other_object, 'wow'),  
    alert  
)))
```

Information Hiding Need to know

David Parnas

**On the Criteria to Be Used in
Decomposing Systems into
Modules. (1972)**

Information Hiding
Need to know

A reference is a capability.

Capability Hiding
Need to do

**There are exactly three ways
to obtain a reference in an object
capability security system.**

- 1. By Creation.**
- 2. By Construction.**
- 3. By Introduction.**

1. By Creation

If a function creates an object, it gets a reference to that object.

2. By Construction

An object may be endowed by its constructor with references.

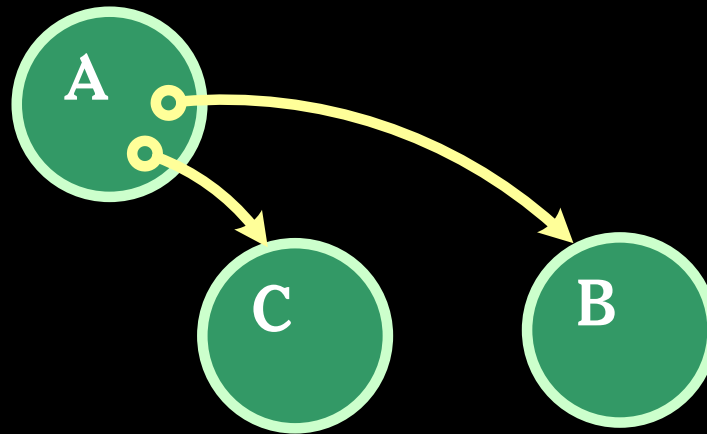
This can include references in the constructor's context and inherited references.

3. By Introduction

A has a references to B and C.

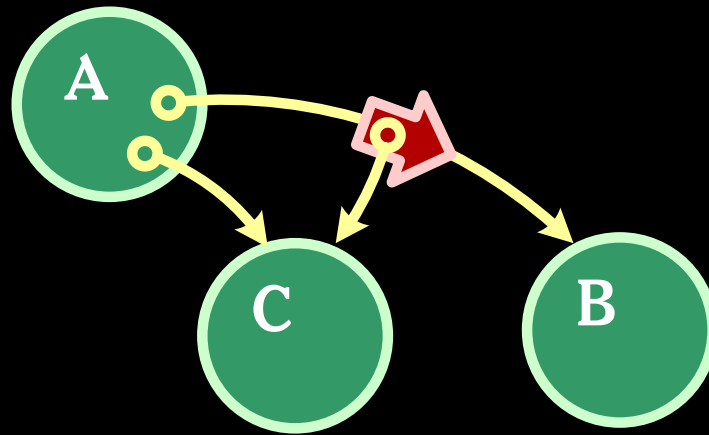
B has no references, so it cannot communicate with A or C.

C has no references, so it cannot communicate with A or B.



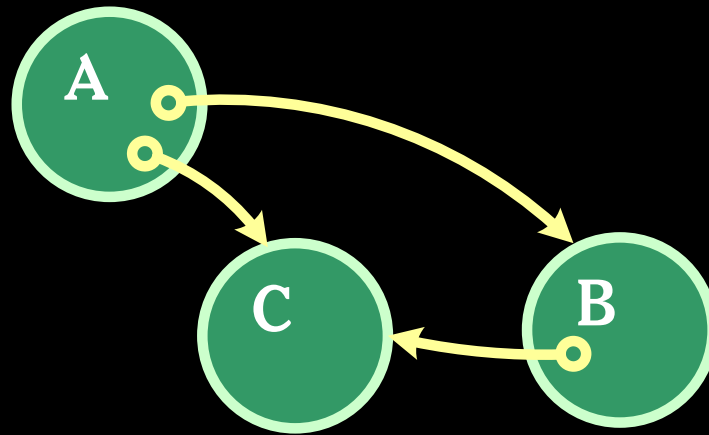
3. By Introduction

A calls B, passing a reference to C.



3. By Introduction

B is now able to communicate with C.



It has the *capability*.

**If references can only be obtained
by Creation, Construction, or
Introduction, then you may have a
safe system.**

**If references can be
obtained in any other
way, you do not have a
safe system.**

**The Lazy Programmer's Guide
to Secure Computing
Marc Stiegler**

**[http://www.youtube.com/watch?
v=eL5o4PFuxTY](http://www.youtube.com/watch?v=eL5o4PFuxTY)**

Performance

Optimize your optimizing.

$$t = c * n$$

t total time

c cost per iteration

n number of iterations

Big O Notation

$O(1)$	Constant
$O(\log n)$	Logarithmic
$O(n)$	Linear
$O(n \log n)$	Loglinear
$O(n^2)$	Exponential
$O(n!)$	Factorial

$$t = c * n^2$$

Improvements in c are insignificant for large n .

Performance Threshold

- **Distraction.**
- **Frustration.**
- **Session failure.**
- **Impractical.**
- **Infeasible.**
- **Impossible.**

**To get under the Distraction
limit, it is necessary to give
immediate feedback.**

**To get under the Frustration
limit, it is necessary to keep n
small.**

Just in time data delivery.

Don't fiddle.

**Find where the time is being spent.
It is counter productive to speed
up the places where time is not
being spent.**

**Every bit of waste you squeeze
out helps performance.**

Surprisingly, this is not true.

Arrays

- Should be $O(1)$.
- Can be $O(\log n)$ or $O(n)$.
- A loop that should be $O(n)$ can be $O(n^2)$

Wow!

Just say no.

Don't Tune For Quirks

- **Some browsers have surprising inefficiencies.**
- **A trick that is faster on Browser A might be slower on Browser B.**
- **The performance characteristics of the next generation may be significantly different.**
- **Avoid short-term optimizations.**

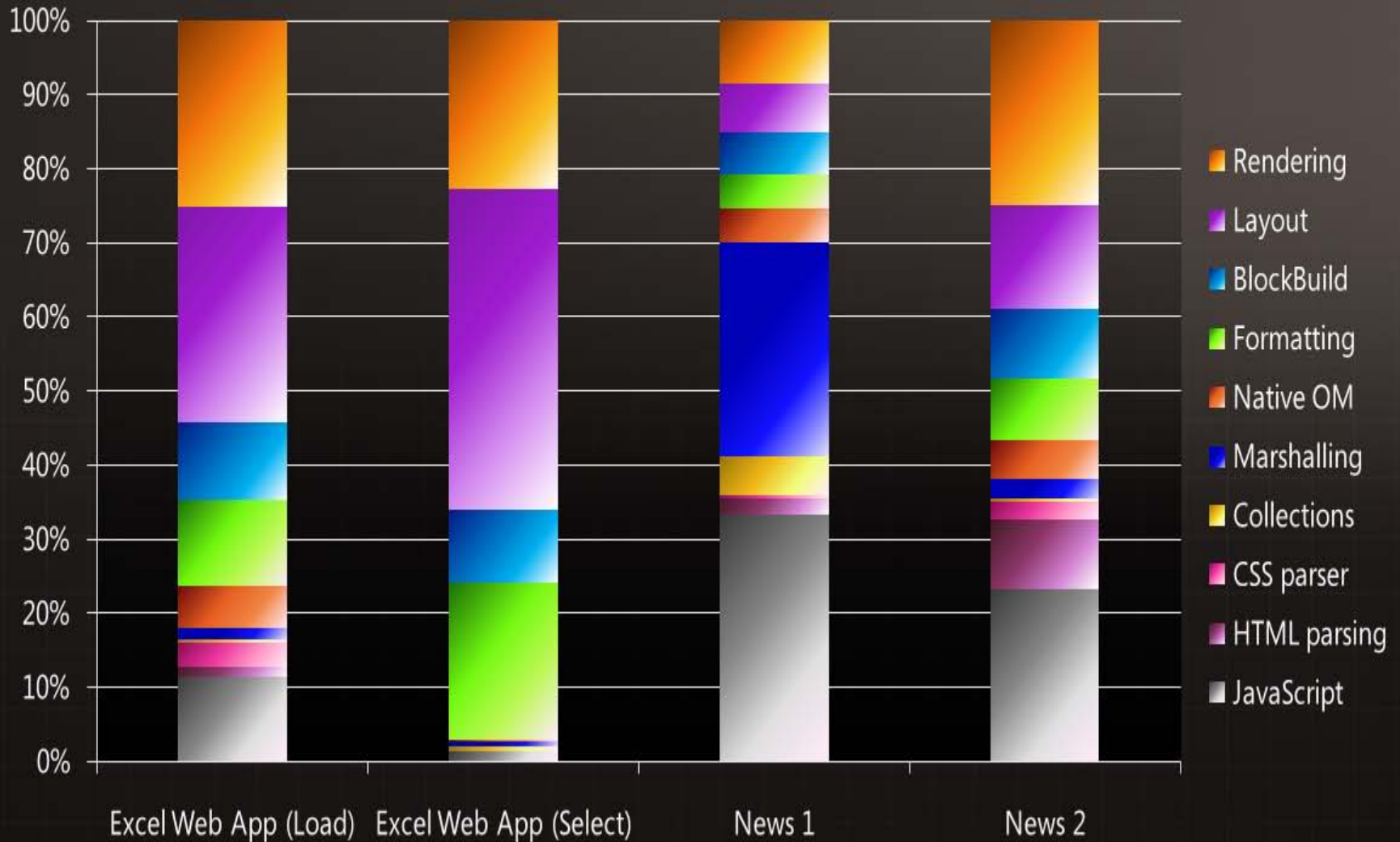
Don't Optimize Without Measuring

- **Intuitions are often wrong.**

```
start_time = Date.now();  
code_to_measured();  
end_time = Date.now();  
elapsed_time = end_time - start_time;
```

- **A single trial is unreliable. Timers can be off by as much as 15 msec.**
- **Even accurate measurements can lead to wrong conclusions.**

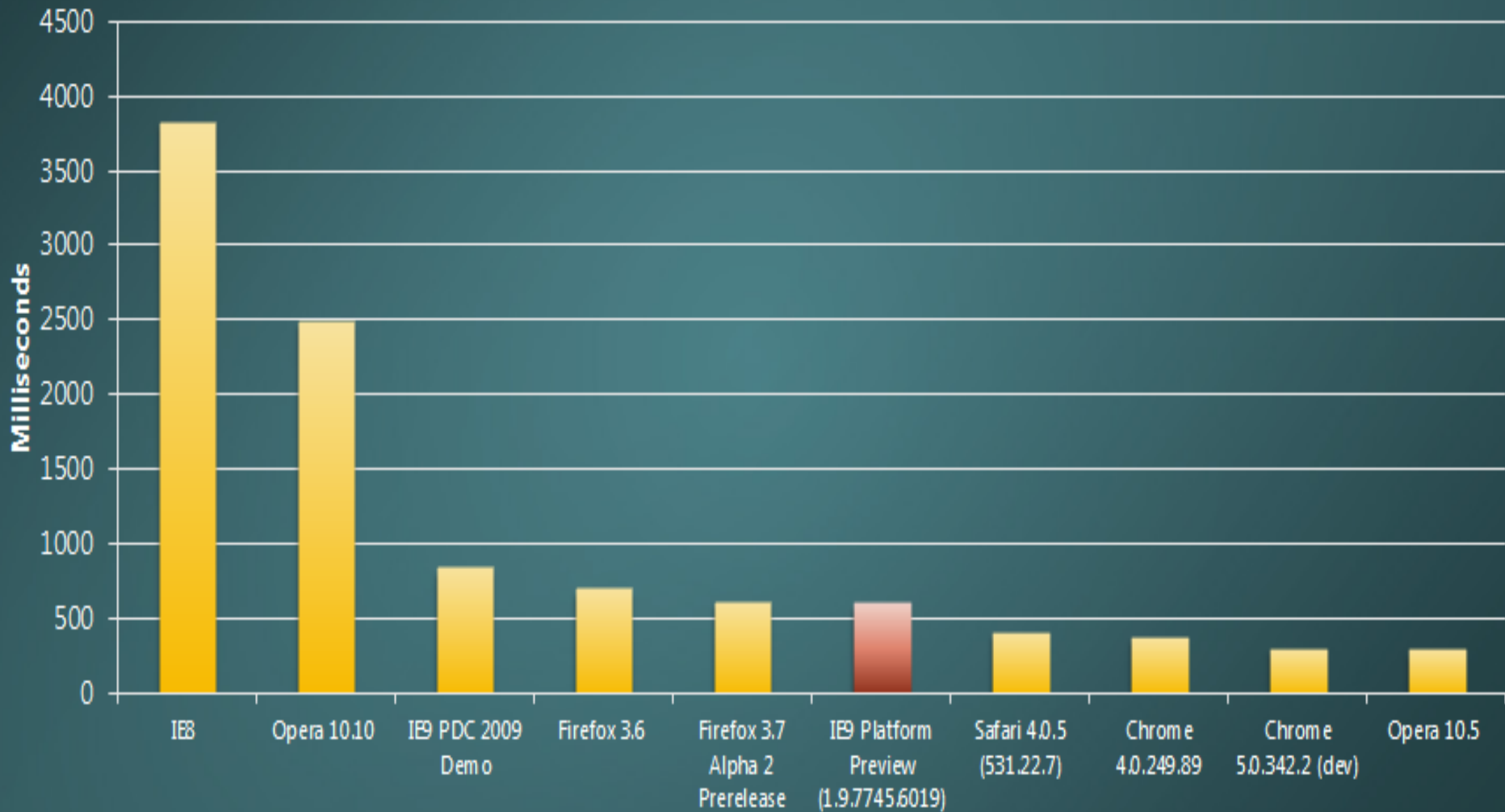
MULTI-SYSTEM PERFORMANCE ENGINEERING



Speed Tracer

Google Chrome Extension

Lies, damn lies, and benchmarks.



Clean, elegant code.

Tonight is a special night.

**Tonight is my 10th anniversary
with JavaScript.**

- I made every mistake that can be made.
- Mistake #1, I didn't bother to learn the language first.
- I tried to write in a Java style, even knowing that it wasn't Java.
- It roundhouse kicked me in the head.

- **After a lot of struggle, I eventually figured out that it was a functional language, and then I stopped fighting it.**
- **We delivered on time and under budget.**

JSLint.com

- I hung out on `comp.lang.javascript`.
- “Help me.”
- Could JSLint have caught that mistake?
- Yes, but it requires discipline:
 Avoid forms that look like errors.

with statement

- It can be extremely useful.
- It can also mask terrible errors.
- Is it possible in static analysis to distinguish the useful cases from the terrible errors?
- No.
- Is it really that useful?
- Can we live without it?

with statement

- **Then it is a Bad Part. Get rid of it.**
- **Remove all of the unessential features and forms that cannot be distinguished from common errors.**
- **Look at what is left. Is it a complete language?**

**It wasn't just a sufficient
language, it was a superior
language.**

Language Subsetting

Only a madman would use all of C++.

- **The ECMAScript Technical Committee (ECMA TC39) is constrained in its ability to remove or repair the bad features of the language.**
- **Dangerous, not useless.**
- **The web is a weirdly fragile thing.**
- **But you can remove bad features simply by refusing to use them.**

(global) Object

- **The object that dares not speak its name.**
- **It is the container for all global variables and all built-in objects.**
- **On browsers, `window` is the global object.**
- **We need to tame or eliminate the global object.**

Implied Global

- Any variable which is not properly declared is assumed to be global by default.
- This makes it easy for people who do not know or care about encapsulation to be productive, but it makes applications less reliable.
- This statement may use a global variable `i`

```
for (i = 0; i < n; i += 1) {
```

Global variables should be as rare as hen's teeth, and should stick out like a sore thumb.

ALLCAPS

+ +

++

x += 1

x++

++

x += 1

~~x++~~

++x

Running with scissors

We make tradeoffs.

Bugs are an inevitable hazard.

We should shift the odds in our favor.

Programming style should be conventions that promote good tradeoffs.

Avoidance of ambiguity, confusion, potential errors.

JSLint is my proofreader.

**JSLint makes it easier for me to
write good code.**

Future of ECMAScript

Growing Role

**Servers, mobile,
consumer electronics**

The Kitchen Sink

Dumbing down
Gunking up

New Languages

Write well.

The End of All Things