



# CHAKRA: UNDER THE HOOD

Steve Lucco  
*Technical Fellow*  
*Microsoft*



# Design Principles

- Security
- ECMAScript Compliance
- Balanced Performance
- Transparency

# JIT Security

Data Execution Protection

Codebase Alignment Randomization

Random NOP Insertion

Constant Blinding

JIT Code Allocation Cap

JIT Page Randomization



```
int          3
int          3
push        ebp
mov         ebp, esp
...
xor         eax, eax
xor         ecx, ecx
lea          ecx, [ecx]
$enterLoop:
cmp         ecx, 0x0a
mov         edi, edi
$jge        $exitLoop
mov         edx, 0x02EBCC90
xor         edx, 0x50A2B255
add         eax, edx
jo          $handleOverflow
inc         ecx
jmp         $enterLoop
$exitLoop:
shl         eax, 1
jo          $handleOverflow
inc         eax
mov         esp, ebp
pop         ebp
ret
```

# JIT Hardening Comparison



JIT Hardening Techniques	Chrome	Internet Explorer	Firefox
Codebase Alignment Randomization	✗	✓	✗
Instruction Alignment Randomization	✗	✓	✗
Constant Folding	✓	✓	✗
Constant Blinding	✓	✓	✗
Resource Constraints	✓	✓	✗
Memory Page Protection	✗	✓	✗
Additional Randomization	✓	?	✗
Guard Pages	✓ *	?	✗



Technique was implemented



Technique was not necessary



Technique was not implemented

\* Chrome 14

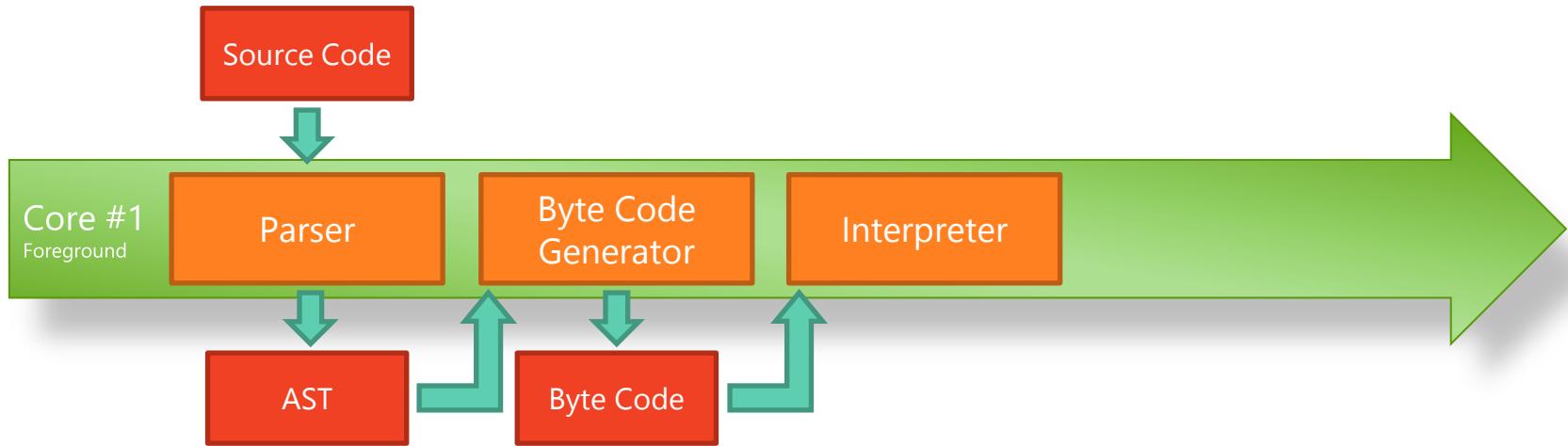
Figure 17. JIT hardening overview

# ECMAScript Compliance

The screenshot shows a web browser window displaying the results of an ECMAScript test suite. The title bar reads "ECMAScript Language test262" and "ECMAScript.org". The main content area has tabs for "Home", "Run" (which is selected), "Results", and "Development". A message at the top says: "Please click on the Run All button to run all the tests. Once you start the test you may pause the test anytime by clicking on the Pause button. You can click on the Results tab once the test is completed or after pausing the test. The Reset button is for restarting the test run. You may run individual tests by clicking the Run button next to the tests listed below. If you wish to run several chapters in sequence, but not the entire test suite, click the Select button for the chapters you wish to run and then click the Run Selected button." Below this, a yellow bar says "Testing complete!". There are buttons for "Run All" and "Run Selected Tests". The text "Tests To run: 11571 | Total tests ran: 11571 | Pass: 11570 | Fail: 1 | Failed to load: 0" is displayed. A list of chapters with their test counts and run buttons follows: Chapter - ch07 (715 tests), Chapter - ch08 (182 tests), Chapter - ch09 (128 tests), Chapter - ch10 (377 tests), Chapter - ch11 (1320 tests). Below this is a detailed test result for "15.5.4.9\_CE": "Tests that String.prototype.localeCompare returns 0 when comparing Strings that are considered canonically equivalent by the Unicode standard." The status is "Fail". At the bottom, it says "Test suite version: ES5.1 | Test suite date: 2012-09-10" and "© Ecma International".

**Highest Pass Rate**

# Balanced Performance: Page Load



## WORLD RECORD SPEED TEST

Share

More info



0:00 / 1:44



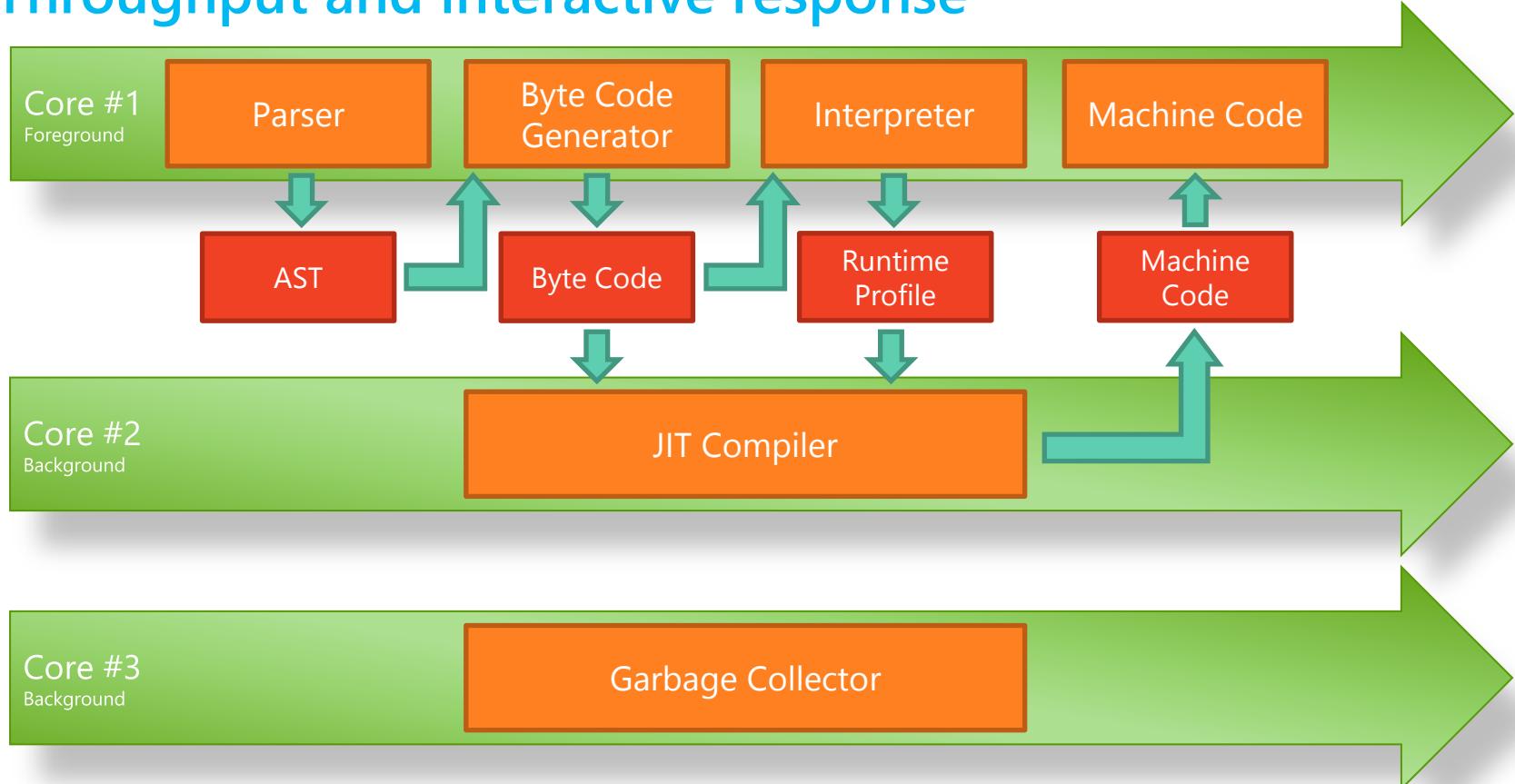
YouTube



# Page Load & App Start-Up

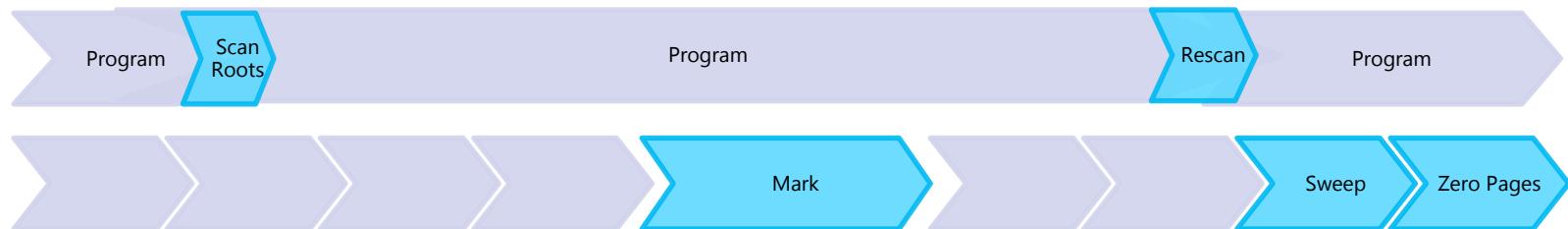
- One of the most visceral elements of user experience
- Internal and third-party reviews show IE has solid page load performance
  - Strangeloop: <http://bit.ly/Sxcw2O>
    - *"Internet Explorer 10 served pages faster than other browsers..."*
  - Tom's Hardware: <http://bit.ly/OY3Bw0>
    - *"Here, Microsoft's own IE9 takes the lead..."*
- Page load design points
  - Interpreter: start execution almost immediately
  - Deferred Parsing: avoid parsing unused code
  - Start-Up Profile Caching: remember which functions were called
  - Background code generation and garbage collection

# Balanced Performance: Throughput and interactive response



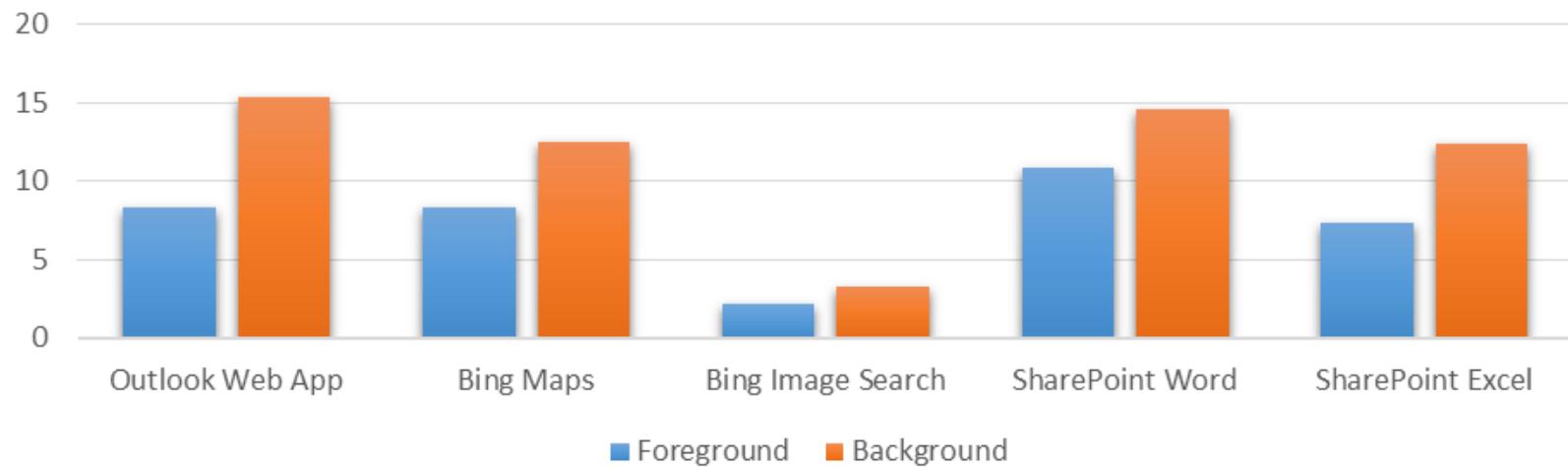
# Chakra's Garbage Collector

- Conservative
  - Can handle object pointers on the native stack; tagged integers lead to very low rate (0.02 per GC) of spurious object references
  - Simplifies interoperation with native code
- Generational
  - partial collections; no separate nursery space
- Mark and Sweep
  - small objects sorted by size into buckets for low fragmentation
  - free-list and bump allocation, currently no compaction or evacuation
- Concurrent



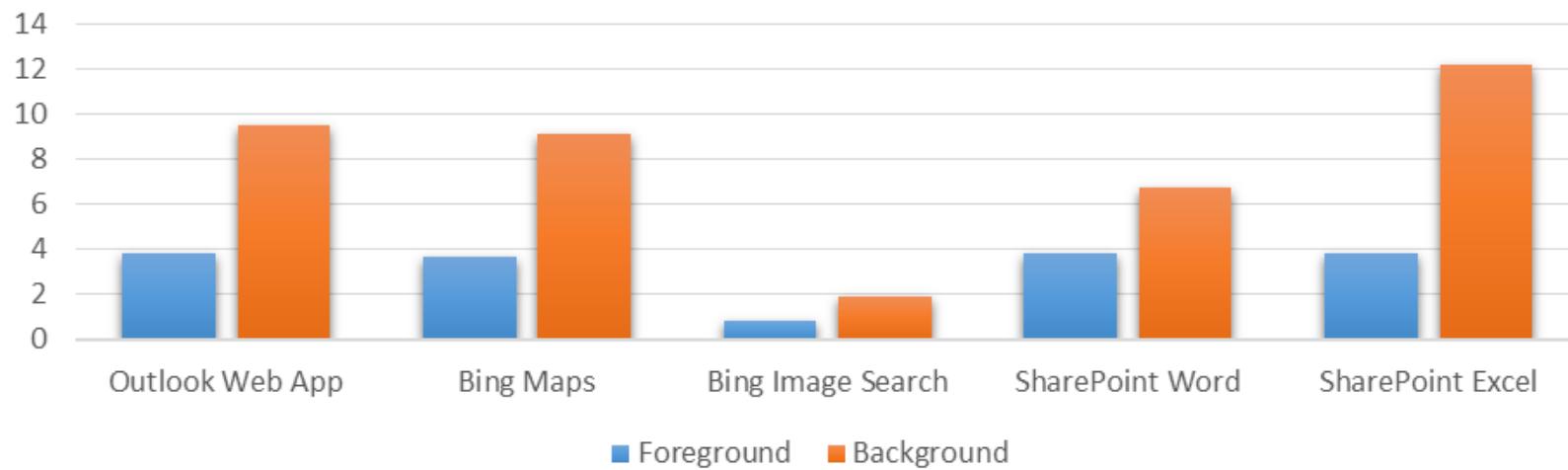
# Interactive Response: Pause Times

Max time in garbage collection  
(in milliseconds)

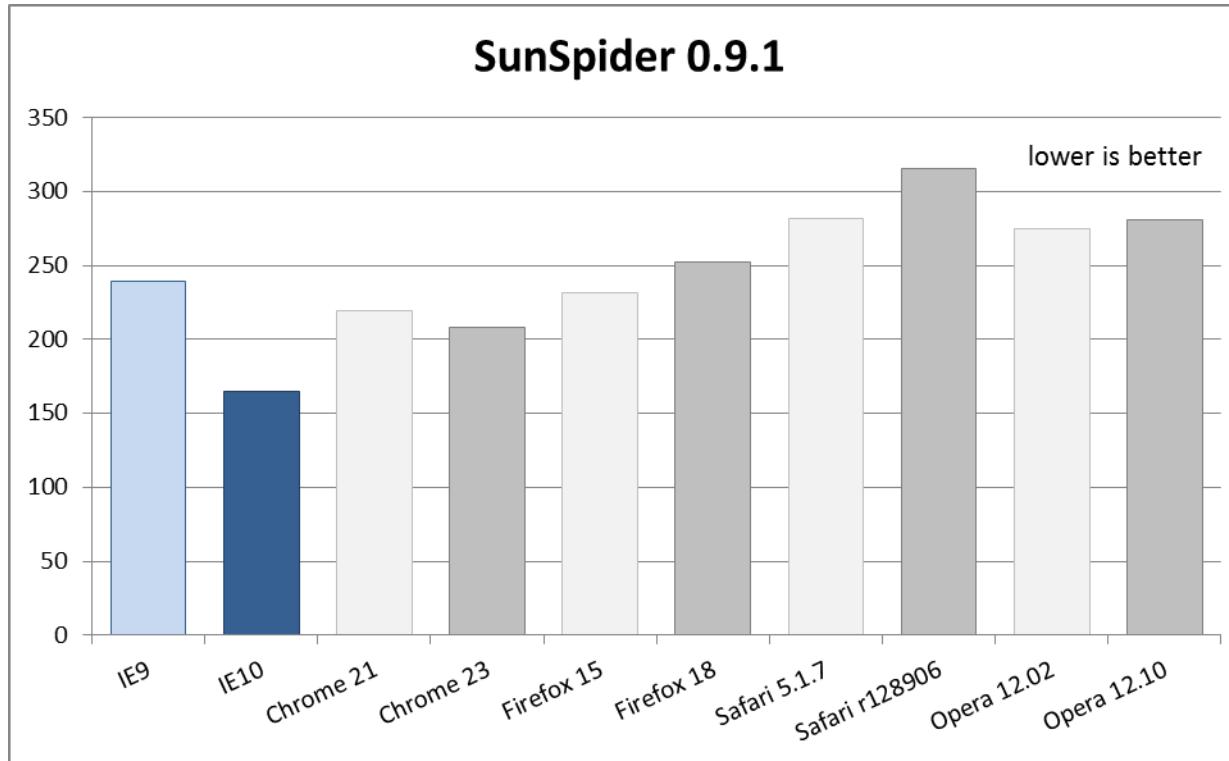


# Interactive Response: Pause Times

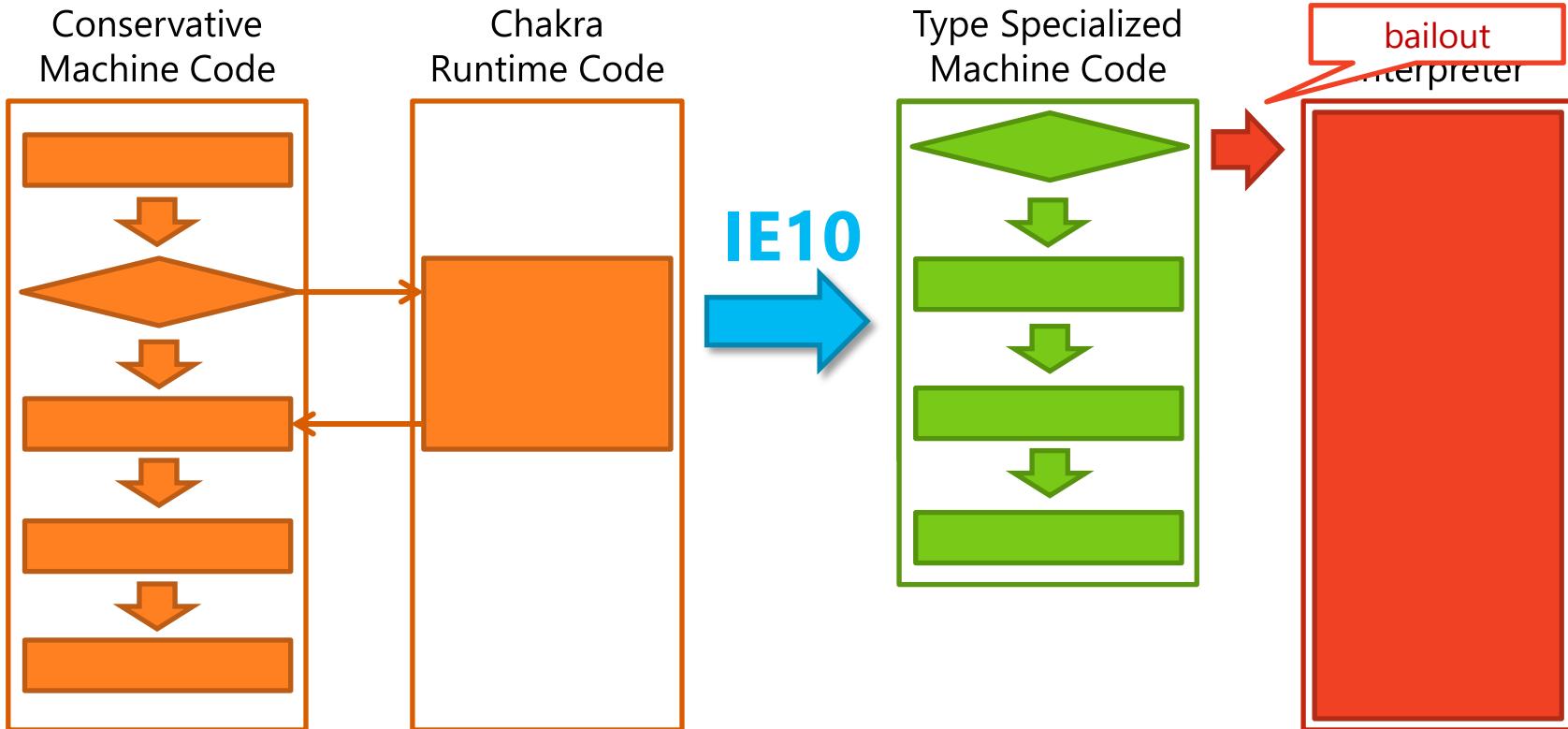
Average time in garbage collection  
(in milliseconds)



# WebKit SunSpider



# Optimistic Profile-Based JIT



# Type Specialized Integer Math in IE10

bitops-bits-in-byte.js

```
function bitsinbyte(b) {  
    var m = 1, c = 0;  
    while(m<0x100) {  
        if(b & m) c++;  
        m <<= 1;  
    }  
    return c;  
}
```



\$enterLoop:

cmp	esi, 0x100
jge	\$exitLoop
mov	ecx, eax
and	ecx, esi
test	ecx, ecx
jeq	\$l1
add	edi, 1
jo	\$bailOut

\$l1:

shl	esi, 1
jmp	\$enterLoop

# Type Specialized Float Math in IE10

3d-cube.js

```
for (; i < NumPix; i++) {  
    Num += NumAdd;  
    if (Num >= Den) {  
        Num -= Den;  
        x += IncX1;  
        y += IncY1;  
    }  
    x += IncX2;  
    y += IncY2;  
}
```



\$enterLoop:

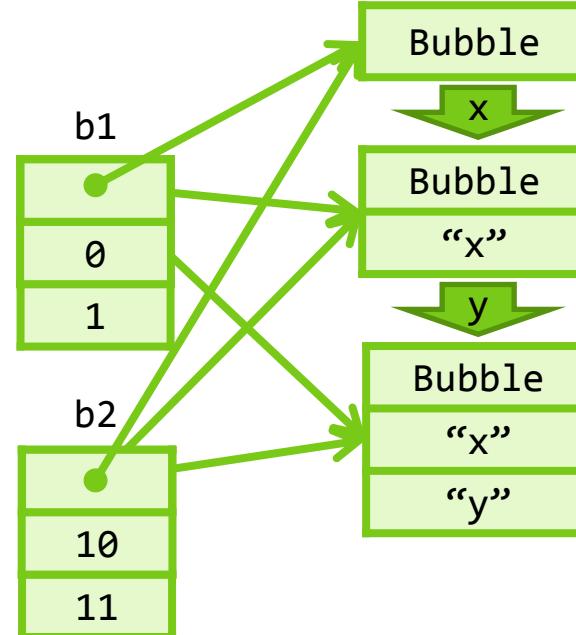
cmp	eax, edx
jge	\$exitLoop
addsd	xmm7, xmm2
comisd	xmm7, xmm6
jb	\$12
subsd	xmm7, xmm6
<b>movsd</b>	<b>xmm2, &lt;-176&gt;</b>
addsd	xmm0, xmm2
addsd	xmm1, xmm3
\$12:	
addsd	xmm0, xmm4
addsd	xmm1, xmm5
add	eax, 1
<b>jo</b>	<b>\$bailOut</b>
movsd	xmm2, <-192>
jmp	\$enterLoop

# Fast Property Access in IE9

```
function Bubble(x, y) {  
    this.x = x;  
    this.y = y;  
}  
var b1 = new Bubble(0, 1);  
var b2 = new Bubble(10, 11);
```

b1.type  b2.type

**monomorphic**

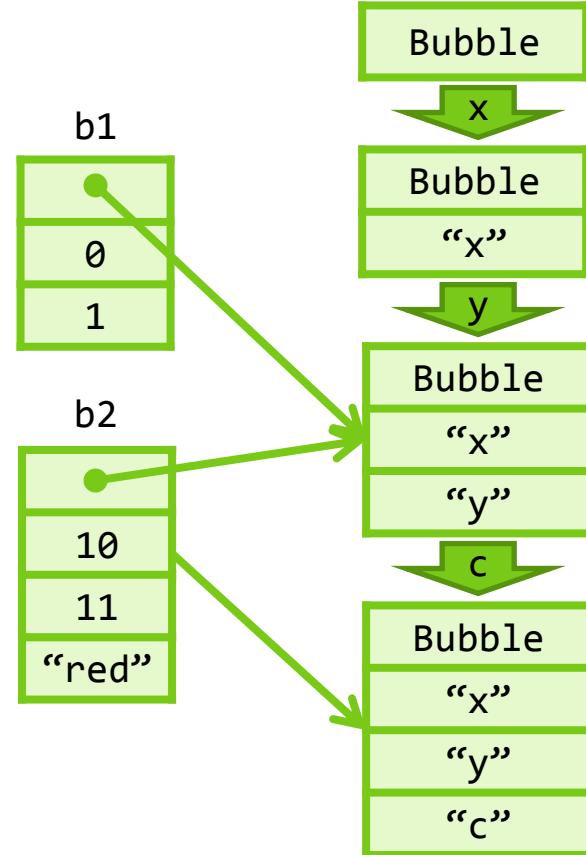


# Fast Property Access in IE9

```
function Bubble(x, y) {  
    this.x = x;  
    this.y = y;  
}  
var b1 = new Bubble(0, 1);  
var b2 = new Bubble(10, 11);  
b2.c = "red";
```

b1.type  b2.type

**polymorphic**



# Faster Property Access in IE10

- Object type specialization
- Polymorphic property caches
- Field hoisting
- Copy propagation
- Streamlined object layout
- Function inlining

# total += o.x + o.y + o.z

```
mov        edi,dword ptr [ebx+88h]
mov        eax,18BF198h
test       edi,1
jne        053F01D7
mov        ecx,dword ptr [edi+8]
cmp        ecx,dword ptr [eax]
jne        053F01D7
movzx     eax,word ptr [eax+6]
mov        eax,dword ptr [edi+eax*4]
mov        edx,18BF1A8h
test       edi,1
jne        053F01F5
mov        ecx,dword ptr [edi+8]
cmp        ecx,dword ptr [edx]
jne        053F01F5
movzx     edx,word ptr [edx+6]
mov        edx,dword ptr [edi+edx*4]
...
mov        eax,18BF1B8h
test       edi,1
jne        053F0231
...
```



```
mov        edi,dword ptr [ebp-0A8h]
test       edi,1
jne        $BailOut
mov        eax,dword ptr [edi+8]
cmp        dword ptr ds:[8E4F20h],eax
jne        $BailOut
mov        eax,dword ptr [edi+1Ch]
...
mov        ecx,dword ptr [edi+20h]
...
mov        eax,dword ptr [edi+24h]
...
```

```
for(...) {  
    total += o.x + o.y + o.z;  
}
```

loop header  
1x

loop body  
100x



```
test    esi, 1  
jne     $bailOut  
mov    eax, dword ptr [esi+8]  
cmp    eax, [0x00480500]  
jne     $bailOut  
mov    eax, dword ptr [esi+28]  
mov    ecx, dword ptr [esi+32]  
mov    edx, dword ptr [esi+36]  
...  
add    eax, ecx  
jo     $bailOut  
...  
add    eax, edx  
jo     $bailOut  
...  
add    ebx, eax  
jo     $bailOut  
...
```

o is {x,y,z}?

o.x

o.y

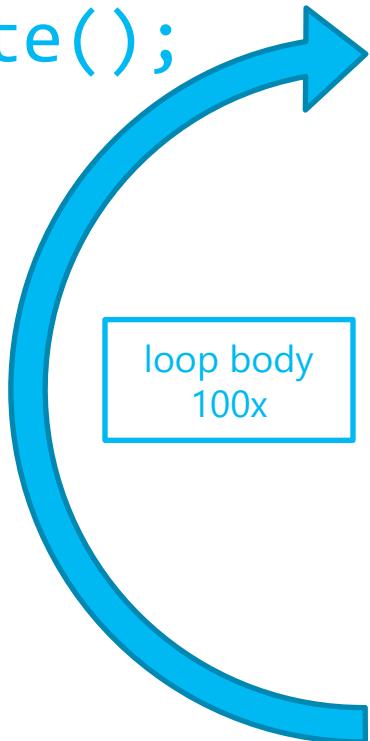
o.z

$t = o.x + o.y$

$t += o.z$

$total += t$

```
for(...) {  
    total += o.x + o.y + o.z;  
    calculate();  
}
```



test	esi, 1	[calculate]	o is {x,y,z}?
jne	\$bailOut		
mov	eax, dword ptr [esi+8]		
cmp	eax, [0x00480500]		
jne	\$bailOut		
mov	eax, dword ptr [esi+28]		
mov	ecx, dword ptr [esi+32]		
mov	edx, dword ptr [esi+36]		
...			
add	eax, ecx		
jo	\$bailOut		
...			
add	eax, edx	t = o.x + o.y	
jo	\$bailOut		
...			
add	ebx, eax	t += o.z	
jo	\$bailOut		
...			
call	[calculate]	total += t	

```
for(...) {  
    total += o.x + o.y + o.z;  
    calculate();  
}
```

loop header  
1x

loop body  
100x



```
test    esi, 1  
jne     $bailOut  
mov    eax, dword ptr [esi+8]  
cmp    eax, [0x00480500]  
jne     $bailOut  
mov    eax, dword ptr [esi+28]  
mov    ecx, dword ptr [esi+32]  
mov    edx, dword ptr [esi+36]  
...  
add    eax, ecx  
jo     $bailOut  
...  
add    eax, edx  
jo     $bailOut  
...  
add    ebx, eax  
jo     $bailOut  
...  
$inlinedCalculate:
```

o is {x,y,z}?

o.x

o.y

o.z

$t = o.x + o.y$

$t += o.z$

$total += t$

# Windows Store Applications

- Bytecode Caching
- GC on Idle/Suspend
- Fast marshaling to native code
  - Native calling conventions and exception handling
  - Generation and caching of method entry points  
(based on meta-data)

# More work to do

- Throughput
  - Array operations; typed arrays
  - Polymorphism and function inlining
- Standards
  - ES6 features; ES5 accessor performance
- Improve GC for games and long-running applications
  - Precise pointers
  - Iterate between sequential and concurrent phases

# Make web development work for any app

- Great JS engine performance
  - Multiple cores, GPU, continued optimization
- APIs, device capabilities, secure component model
- Build tools that enable construction of large-scale Javascript applications