Microsoft Research

Each year Microsoft Research hosts hundreds of influential speakers from around the world including leading scientists, renowned experts in technology, book authors, and leading academics, and makes videos of these lectures freely available.

2016 © Microsoft Corporation. All rights reserved.
Louis Lafreniere
Principal Software Engineer Manager, Chakra

Louis was hired by Microsoft to work on its first shipping 32bit compiler backend for C/C++. His first task was to add the C++ support. After 18 years of squeezing perf out of C++ and working on security features and 3 backends later, Louis jumped on the opportunity to work on Chakra during the early IE9 days to write a JIT for JavaScript. That was before he realized what JavaScript was all about...
Chakra: From Script to Optimized Code

By some accounts, JavaScript is the most popular programming language in the world. Yet it’s also not a language that was designed with fast performance in mind. The Chakra team embarked on a journey to make JavaScript fast in IE9 and the Chakra runtime can today power everything from tiny scripts on a page to modern applications and services running on Azure. In this talk, we’ll cover the basic architecture of the engine that transforms JavaScript into fast native code.
Where are we in the Web Platform?
Where are we in the Web Platform?

This is us
Evolution of Chakra

IE3 to IE8
Evolution of Chakra

IE3 to IE8

IE9 to IE11
Evolution of Chakra

1. jscript.dll
   - IE3 to IE8

2. jscript9.dll
   - IE9 to IE11

3. chakra.dll
   - Windows 10

4. ChakraCore.dll
   - Open Source

5. Windows 10 (RS1+)

6. Integrates w/ Windows 10

Node
TypeScript
DocDB
Architecture of Chakra
Chakra and ChakraCore

Chakra
- COM API
- Projection
- Diagnostic API
- Static Library
- SCA

ChakraCore
- JSRT API
- TTD
- PAL
- Codex
- Common Datastructures

Front End
- Parser
- Byte Code Generator

Backend
- Global Optimizer
- Register Allocator
- Lowerer
- Encoder

Library
- Global
- Array
- Number
- String
- Date
- Function
- Regex
- Promise
- JSON
- Typed Array

Execution Machinery
- Interpreter
- Types
- Language
- Inline Cache
Chakra and ChakraCore
Making script executable

Script Document

Script

Function foo(a, b)
{
    return a+b;
}

Hosting

ActiveScript
ActiveScriptDirect
ActiveScriptBycode
Making script executable

```
Function foo(a, b) 
  { 
    return a + b; 
  }
```

```
program 
Scopes: knapBlock(Global) 
Scopes: knapFnDecl 
fn def 1 name foo 
Scopes: knapBlock(Parameter) 
Scopes: knapBlock(Function) 
var a 
var b 
List 
  return + 
  id: a 
  id: b 
<endcode>
```
Making script executable

Script Document → Script: Function foo(a, b) { return a+b; } → AST

Hosting:
- NativeScript
- NativeScriptDirect
- NativeScriptBytecode

UVB Buffer → Parser → AST

Byte Code Generator → Bytecode

Interpreter

Library

Type System

AST:
- program
  - Scopes: knobBlock(Global)
  - Scopes: knobFnDecl
    - fn decl: name foo
  - Scopes: knobBlock(Parameter)
    - Scopes: knobBlock(Function)
      - var a
      - var b
      - List
        - return
        - id: a
        - id: b
        - <endcode>

Bytecode:
- R1 LdRoot
- R2 Argln_A In1
- R3 Argln_A In2
- Add_A R0 R2 R3
- Ret
Making script executable

Script Document

Script

Function foo(a, b) {
    return a + b;
}

AST

program

Scopes: knobBlock(Global)
Scopes: knobFnDecl

fn: def fn name foo
Scopes: knobBlock(Parameter)
Scopes: knobBlock(Function)

var a
var b

lit

return
+ id: a
+ id: b
@endcode

Bytecode

R1 LdRoot
R2 ArgIn_A In1
R3 ArgIn_A In2
Add_A R0 R2 R3
Ret

Hosting

ActiveScript
ActiveScriptDirect
ActiveScriptBytecode

Parser

AST

Byte Code Generator

Bytecode

Interpretor

JIT

Library

Type System
/* Examples of a+b */

> 1 + 1
2
/* Examples of a+b */

> 1 + 1
2
> 1 + ""
'1'
/* Examples of a+b */

> 1 + 1
2

> 1 + ""
'1'

> "a" + "b"
'ab'
/* Examples of a+b */

> 1 + 1
2

> 1 + 
'1'

> "a" + "b"
'ab'

> "a" + 1
'a1'
/* Examples of a+b */

> 1 + 1
2

> 1 + ""
'1'

> ''a'' + ''b''
'ab'

> ''a'' + 1
'a1'

> {} + {}
'[object Object][object Object]'
/* Examples of a+b */

> 1 + 1
2
> 1 + ""
'1'
> "a" + "b"
'ab'
> "a" + 1
'a1'
> {} + {}
'[object Object][object Object]'
/* Examples of a+b */

> 1 + 1
2

> 1 + ""
'1'

> "a" + "b"
'ab'

> "a" + 1
'a1'

> {} + {}
'[object Object][object Object]'

> {} + ''
'

> '' + {}
'[object Object]'
/* Examples of a+b */

> 1 + 1
2
> 1 + ""
'1'
> "a" + "b"
'ab'
> "a" + 1
'a1'
> {} + {}
'[object Object][object Object]'
> {} + ''
'
> '' + {}
'[object Object]'
/* Examples of a+b */

> 1 + 1
2

> 1 + ""
'1'

> "a" + "b"
'ab'

> "a" + 1
'11'

> {} + {} 
'[{object Object}[object Object]']

> {} + ''
0

> '' + {}
'[{object Object}]'

> ({} + '')
'[{object Object}]

> 0 + []
'0'
To get \( a+b \) correct...
Optimizing and Executing code
Lifecycle of code

```javascript
function Sum(numbers) {
  var total = 0;
  for (var i = 0; i < numbers.length; i++) {
    total += numbers[i];
  }
  return total;
}
```
Lifecycle of code

```javascript
function Sum(numbers) {
    var total = 0;
    for (var i = 0; i < numbers.length; i++) {
        total += numbers[i];
    }
    return total;
}
```

Bytecode

```
R1 LdRst
R2 LdL_A, H 0
R3 LdL_A, W 1
R4 LdN_A R5
LdW_A R6
LdR_A R5 R2
LdR_A R6 R2
LoopStart:
LdL_A R7 = R4.length
LdL_B R7 = R4[0] R1
Add_A R5 R5 R7
Inc_A R6 R6
R7 LoopEnd
```

UI Thread

JIT Thread

JIT
function Sum(numbers) {
    var total = 0;
    for (var i = 0; i < numbers.length; i++)
    {
        total += numbers[i];
    }
    return total;
}
Lifecycle of code

```javascript
function Sum(numbers) {
    var total = 0;
    for (var i = 0; i < numbers.length; i++) {
        total += numbers[i];
    }
    return total;
}
```

UI Thread

JIT Thread

JIT

Profile Data

Bytecode Interpreter

JIT Loop Body

if (isInt(i))
    bailout()
    i = Total32(i)
if (isInt(total))
    bailout()
    total.j = Total32(total)
if (isNaN(number))
    bailout()
    i = number-length
    d = number-data
loop:
    if (i,j > 0)
        break
    total.j += d[i]
    i++
    goto loop
    i = Total(i)
    total = Total(total.i)
    return
function Sum(numbers) {
    var total = 0;
    for (var i = 0; i < numbers.length; i++) {
        total += numbers[i];
    }
    return total;
}

if (typeof(n))
    bailout()
    i, = Total32(i)
if (typeof(total))
    bailout()
    total, = Total32(total)
if (typeof(number))
    bailout()
    l, number-length
    d, number-data
    loop
        if (i, = 0)
            break
        total, += d[i,]
        i++
        go loop
    i, = Total32(i)
    total, = Total32(total)
    return
Lifecycle of code

```javascript
function Sum(numbers) {
    var total = 0;
    for (var i = 0; i < numbers.length; i++) {
        total += numbers[i];
    }
    return total;
}
```

Loop:
- `if (isArrayOfNumber(numbers))`:
  - `i = number.length`
  - `l = Js::GetLength(number)`
  - `if (i >= l)`: break
  - `else`:
    - `l = 1`
    - `if (i > l)`: break
    - `else`:
      - `if (Js::GreaterThanEq(i, l))` break
- `if (isArrayOfNumber(numbers) && isInt(i) && i < number.length)`:
  - `n = number.data[i]`
  - `a = Js::GetElement(number, i)`
  - `if (isInt(n) && isInt(total))`:
    - `total += a`
    - `else`:
      - `total = Js::Add(total, n)`
      - goto Loop
  - `return total`
Life cycle of code

```javascript
function Sum(numbers) {
  var total = 0;
  for (var i = 0; i < numbers.length; i++) {
    total += numbers[i];
  }
  return total;
}
```
function Sum(numbers) {
    var total = 0;
    for (var i = 0; i < numbers.length; i++) {
        total += numbers[i];
    }
    return total;
}
function Sum(numbers) {
    var total = 0;
    for (var i = 0; i < numbers.length; i++) {
        total += numbers[i];
    }
    return total;
}

Sum([1, 2, 3, 4, 5])
Lifecycle of code

```javascript
function Sum(numbers) {
    var total = 0;
    for (var i = 0; i < numbers.length; i++) {
        total += numbers[i];
    }
    return total;
}
Sum([1, 2, 3, 4, 5])
```
Lifecycle of code

```javascript
function Sum(numbers) {
    var total = 0;
    for (var i = 0; i < numbers.length; i++) {
        total += numbers[i];
    }
    return total;
}

Sum([1, 2, 3, 4, 5])
```

```javascript
if (!isFinite(number))
    bailout();

i = 0
l = numbers.length
d = numbers.length

loop
    if (i > l - 1)
        break
    n = array[d]
    if (!isNumber(n))
        bailout()
    n_f = ToFloat(n)
    total_f += n_f
    i++
go to loop
return total = ToVar(total_f)
```