Reusable Inline Caching for JavaScript Performance

**Jiho Choi**, Thomas Shull, and Josep Torrellas

Department of Computer Science
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

PLDI 2019
Two Contradicting Trends on Webpage Load Performance

- Ever higher user expectation in page load time: < 1 sec in 2015
- Increased program complexity: 28 JavaScript (JS) files per page
- Our goal: Reduce JS initialization time
Problem: Limited Reuse of Compilation Results Across JS Executions

• JS compilers currently have limited reuse of compilation results across executions
  – Code cache for bytecodes to avoid parsing overhead
  – Snapshot support to avoid built-in object creation

• Compilation results are typically **Context-Dependent** (i.e., contain memory addresses specific to an execution context)
  – Optimized codes embed memory addresses for built-in objects, object types, etc.
  – Different executions may create different shapes of objects depending on control paths taken

• Result: Overheads of optimization (e.g., type specialization) repeat every execution
Overhead of Type Specialization During JS Initialization

- On average, 36% of instructions are spent on type specialization during initialization of popular JS libraries.

- This overhead is the target of our paper.
Overhead of Type Specialization During JS Initialization

- On average, 36% of instructions are spent on type specialization during initialization of popular JS libraries.

This overhead is the target of our paper.

In the next few slides:
- Why statically-typed languages do not have this overhead
- How type specialization works in JS compilers
Why Statically-Typed Languages are Fast: Type Declarations

• Static type information is crucial to generating efficient code

Type Declaration

[C Code]
struct A {
    int x;
    int y;
};
A o;
if (...) o.x = 1;
o.y = 2;

[Code for Object Access Sites]
// o.x = 1;
o[0] = 1
// o.y = 2;
o[1] = 2

Type declaration provides compiler with static object layout
JS Has No Types

- Properties can be added and removed at any time
- Objects are simply hash tables from property names to values
- Flexible for programmers but challenging for compilers

```javascript
var o = {};
if (...) o.x = 1;
o.y = 2;
```

Shape of `o`?

Compile

How to generate code when shape of `o` is unknown?

(If branch not taken)

```
[Object o]
y: 2
```

OR

(If branch taken)

```
[Object o]
x: 1
y: 2
```
Compilers Dynamically Track Types Internally

- Properties can be added and removed at any time
- Objects are simply hash tables from property names to values
- Flexible for programmers but challenging for compilers

```javascript
var o = {};
if (...) o.x = 1;
o.y = 2;
```

Hash tables

Compilers introduce type system behind the scenes: **Hidden Classes**
Naïve Implementation of Property Access with HC: Hash Table Lookup

**[Object o]**

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>2</td>
</tr>
</tbody>
</table>

hc_ptr

**[HC_A]**

- Property: y
- Offset: 0

**[Object o]**

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>2</td>
</tr>
</tbody>
</table>

hc_ptr

**[HC_B]**

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
</tr>
</tbody>
</table>

**[JavaScript Code]**

```javascript
var o = {};
if (...) o.x = 1;
o.y = 2;
```

**[Code for o.y = 2]**

```javascript
offset = hash_table_lookup(o.hc_ptr, "y")
o[offset] = 2
```

Property access always entails **hash table lookup** to get offset
State-of-the-art Compilers Do Better: Inline Caching (IC)

- For each object access site: IC records previously seen HCs and generates specialized codes for them (Handlers)

- Compiler records (HC, Handler) in ICVector

```
[ICVector]

<table>
<thead>
<tr>
<th>Incoming HC</th>
<th>HC_A</th>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handler</td>
<td>Handler_A</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

- ICVector is initially empty and is dynamically filled as new HCs are encountered in an object access site

- Initial execution of an object access site is still slow, but the subsequent accesses become very efficient
How IC Works

[JavaScript Code]
var o;
if (...) o.x = 1;
o.y = 2;

Compile

[Dispatcher]
for each entry in ICVector:
if (o.hc_ptr == entry.HC):
jump entry.Handler
jump Runtime

[Handler_A]
o[0] = 2
ret
Use of ICVector

[Object o]
y : 2
hc_ptr

[HC_A]
<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>0</td>
</tr>
</tbody>
</table>

OR

[Object o]
x : 1
y : 2
hc_ptr

[HC_B]
<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
</tr>
</tbody>
</table>

[JavaScript Code]
var o;
if (...) o.x = 1;
o.y = 2;

[Code for o.y = 2]
call Dispatcher

[Dispatcher]
for each entry in ICVector:
  if (o.hc_ptr == entry.HC):
    jump entry.Handler
    jump Runtime

[ICVector]
<table>
<thead>
<tr>
<th>Incoming HC</th>
<th>HC_A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handler</td>
<td>Handler_A</td>
</tr>
</tbody>
</table>

[Handler_A]
o[0] = 2
ret
Use of ICVector

[JavaScript Code]
```javascript
var o;
if (...) o.x = 1;
o.y = 2;
```

[Code for o.y = 2]
call Dispatcher

[Dispatcher]
for each entry in ICVector:

Recap:
- **Type specialization**: Optimizing code to be fast for specific hidden class
- **Inline Cache (IC)**: Actual optimized code and data structure
- **IC Hit**: Execution encounters one of recorded hidden classes ➔ Fast
- **IC Miss**: Requires hash table lookup and type specialization ➔ Very Slow
IC Miss is Very Expensive and Repeats Every Execution

- IC miss handling takes 1000s of instructions
- Overheads of IC miss handling are especially high during JS initialization
  - Many object access sites are executed for first time causing IC misses

Can we reuse IC information across executions?
Opportunities for Reuse in IC

• Instrumented Google V8 JS engine to profile initialization of popular JS libraries and identified three opportunities for reuse

• **Opportunity #1**: Initialization of JS library is mostly deterministic
  – Spend most of time creating and testing library object

• **Opportunity #2**: Same HC is used in multiple object access sites
  – It causes an IC miss at each site when it is first encountered
  – On average, a HC is shared by 5 object access sites

• **Opportunity #3**: Large portion of IC info is *Context-Independent* (i.e., doesn’t contain memory addresses)
  – 60% of handlers are context-independent
Contributions

• Characterized IC miss handling during initialization of popular JS libraries

• Proposed **Reusable Inline Caching (RIC):**
  – Extracts context-independent portion of IC information and reuses it across executions to avoid IC misses

• Evaluated initialization of 7 popular JS libraries in Google V8
  – Reduced # of IC misses by 53%
  – Reduced # of instructions by 15%
  – Reduced initialization time by 17%
Key Ideas of Reusable Inline Caching (RIC)

- RIC extracts IC information from **Initial Run** and reuses it in subsequent executions (**Reuse Runs**)

- IC information extracted from **Initial Run**
  - **Triggering Sites**: Object access sites that create HCs
  - **Dependent Sites of a Triggering Site**: Object access sites that use the HC created by the Triggering Site
  - **Context-independent handlers**

- Operations performed in **Reuse Run** to avoid IC misses
  - Upon IC miss on a **Triggering Site**, fill ICVector for **Dependent Sites**
  - Also, dynamically check that **Reuse Run** takes the same control path as **Initial Run**
Example:
Step 1: Initial Run

[JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

[ICVector]

| Site   | S1 | S2 | L1 | ...
|--------|----|----|----|-----
| Incoming HC_addr |    |    |    | ...
| Handler           |    |    |    | ...

[Hidden Class]

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>

[Handler]
Example:
Step 1: Initial Run

[JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

[ICVector]

| Site   | S1 | S2 | L1 | ...
|--------|----|----|----|----
| Incoming HC Addr |     |    |    |...
| Handler         |    |    |    |...

[Hidden Class]

[A]

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Handler]
Example:
Step 1: Initial Run

[JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2;     // S2
4: print(o.y); // L1

[ICVector]

<table>
<thead>
<tr>
<th>Site</th>
<th>S1</th>
<th>S2</th>
<th>L1</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incoming HC Addr</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Handler</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

[Hidden Class]

[A]

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>

[B]

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>
y | 0 |

[Handler]

[H1]

obj.HC = B
obj[0] = 2
Example:
Step 1: Initial Run

**[JavaScript Code]**
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2;  // S2
4: print(o.y);  // L1

**[ICVector]**
<table>
<thead>
<tr>
<th>Site</th>
<th>S1</th>
<th>S2</th>
<th>L1</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incoming HC Addr</td>
<td></td>
<td>A</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Handler</td>
<td></td>
<td>H1</td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

**[Hidden Class]**

<table>
<thead>
<tr>
<th>[A]</th>
<th>[B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Offset</td>
</tr>
<tr>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>

| y | 0 |

**[Handler]**

[H1]
obj.HC = B
obj[0] = 2

IC Miss
Example:
Step 1: Initial Run

[JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

[ICVector]

| Site | S1 | S2 | L1 | ...
|------|----|----|----|-----
| Incoming HC_Addr | A | | | ...
| Handler | H1 | | | ...

[Hidden Class]

[A]          [B]

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>y</td>
<td>0</td>
</tr>
</tbody>
</table>

[Handler]

[H1]
obj.HC = B
obj[0] = 2

[H2]
ret obj[0]
Example:
Step 1: Initial Run

[JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

[Hidden Class]
[A]
<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>
[B]
<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>
y       | 0      |

[Handler]
[H1]
obj.HC = B
obj[0] = 2

[H2]
ret obj[0]
Example:
Step 1: Initial Run

[JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

[ICVector]
| Site | S1 | S2 | L1 | ...
|------|----|----|----|---
| Incoming HC Addr | A | B | ... |
| Handler | H1 | H2 | ...

[Hidden Class]

<table>
<thead>
<tr>
<th>[A]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[B]</td>
</tr>
<tr>
<td>Property</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>y</td>
</tr>
</tbody>
</table>

[Handler]

[H1]
obj.HC = B
obj[0] = 2

[H2]
ret obj[0]
### Example:
Step 2: Extract RIC Information

#### [JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

#### [ICVector]

| Site   | S1 | S2 | L1 | ...
|--------|----|----|----|------
| Incoming HC_addr | A | B | ... |
| Handler   | H1 | H2 | ... |

#### [Hidden Class]

<table>
<thead>
<tr>
<th>[A]</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Offset</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[B]</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Offset</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

#### [Handler]

[H1]
- obj.HC = B
- obj[0] = 2

[H2]
- ret obj[0]

#### [RIC Information]

<table>
<thead>
<tr>
<th>[Hidden Class Validation Table]</th>
<th>[Triggering Object Access Site Table]</th>
<th>[Context-Independent Handlers]</th>
</tr>
</thead>
<tbody>
<tr>
<td>HC_ID</td>
<td>HC_addr</td>
<td>V</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DS:** Dependent Site, **S/B:** Site or Built-in HC
Example:
Step 2: Extract RIC Information

[JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

[ICVector]
| Site  | S1 | S2 | L1 | ...
|-------|----|----|----|---
| Incoming HC Addr | A  | B  | ...
| Handler         | H1 | H2 | ...

[Hidden Class]
[A]
<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>
[B]
| Property | Offset |
|          | y      |
|          | 0      |

[RIC Information]
[H1]
obj.HC = B
obj[0] = 2

[H2]
ret obj[0]

[Context-Independent Handlers]

[Hidden Class Validation Table]
<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC_Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Triggering Object Access Site Table]
<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DS: Dependent Site, S/B: Site or Built-in HC
Step 2: Extract RIC Information

**[JavaScript Code]**
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2;  // S2
4: print(o.y);  // L1

**[ICVector]**

| Site  | S1 | S2 | L1 | ...
|-------|----|----|----|---
| Incoming HC_addr | A | B | ...
| Handler | H1 | H2 | ...

**[Hidden Class]**

<table>
<thead>
<tr>
<th>[A]</th>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>[B]</th>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>

**[Handler]**

```
[H1]
obj.HC = B
obj[0] = 2
[H2]
ret obj[0]
```

**[RIC Information]**

```
DS: Dependent Site, S/B: Site or Built-in HC
```

**[Hidden Class Validation Table]**

<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC_addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**[Triggering Object Access Site Table]**

<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>(-, A)</td>
</tr>
<tr>
<td>S2</td>
<td>(A, B)</td>
</tr>
</tbody>
</table>

**[Context-Independent Handlers]**
Example:
Step 2: Extract RIC Information

[JavaScript Code]
1: var o = {};  
2: if (...) o.x = 1; // S1  
3: o.y = 2;  // S2  
4: print(o.y);  // L1

[ICVector]

<table>
<thead>
<tr>
<th>Site</th>
<th>S1</th>
<th>S2</th>
<th>L1</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>o.x</td>
<td>A</td>
<td>B</td>
<td></td>
<td>…</td>
</tr>
<tr>
<td>o.y</td>
<td>H1</td>
<td>H2</td>
<td></td>
<td>…</td>
</tr>
</tbody>
</table>

[Hidden Class]

[A]

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[B]

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>0</td>
</tr>
</tbody>
</table>

[Handler]

[H1]

obj.HC = B

[H2]

ret obj[0]

[RIC Information]

[Hidden Class Validation Table]

<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC_Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>(L1, H2)</td>
<td></td>
</tr>
</tbody>
</table>

[Triggering Object Access Site Table]

<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>(-, A)</td>
</tr>
<tr>
<td>S2</td>
<td>(A, B)</td>
</tr>
</tbody>
</table>

[Context-Independent Handlers]

DS: Dependent Site, S/B: Site or Built-in HC

ILLOINOIS
Example:
Step 2: Extract RIC Information

[JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

[ICVector]

| Site | S1 | S2 | L1 | ...
|------|----|----|----|----
| Incoming HC Addr | A | B | ...
| Handler | H1 | H2 | ...

[Hidden Class]

<table>
<thead>
<tr>
<th>[A]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
</tr>
<tr>
<td>---</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
</tr>
<tr>
<td>---</td>
</tr>
</tbody>
</table>

[Handler]

[H1]

obj.HC = B
obj[0] = 2

[RIC Information]

[Hidden Class Validation Table]

<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC.Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>(L1, H2)</td>
<td></td>
</tr>
</tbody>
</table>

[Triggering Object Access Site Table]

<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>(-, A)</td>
</tr>
<tr>
<td>S2</td>
<td>(A, B)</td>
</tr>
</tbody>
</table>

[Context-Independent Handlers]

[H2]

ret obj[0]

DS: Dependent Site, S/B: Site or Built-in HC
Example:
Step 2: Extract RIC Information

**[JavaScript Code]**
1: var o = {};  
2: if (...) o.x = 1; // S1  
3: o.y = 2; // S2  
4: print(o.y); // L1

**[ICVector]**
| Site    | S1 | S2 | L1 | ...
|---------|----|----|----|-----
| Incoming HC Addr | A  | B  |    | ...
| Handler | H1 | H2 |    | ...

**[Hidden Class]**

**[RIC Information]**

**[Context-Independent Handlers]**

**[Hidden Class Validation Table]**

**[Triggering Object Access Site Table]**

**[Hidden Class]**

**[A]**

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>

**[B]**

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>

RIC information is completely **context-independent** and, thus, **reusable** across executions.

**[Default Vector]**

**[ICVector]**

<table>
<thead>
<tr>
<th>HC ID</th>
<th>HC Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>(L1, H2)</td>
</tr>
</tbody>
</table>

**[S/B] HC ID Transition**

<table>
<thead>
<tr>
<th>S/B</th>
<th>HC ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>(-, 0)</td>
</tr>
<tr>
<td>S2</td>
<td>(0, 1)</td>
</tr>
</tbody>
</table>

**[Context-Independent Handlers]**

**[H2]**

`ret obj[0]`

DS: Dependent Site, S/B: Site or Built-in HC
Example: Step 3: Reuse Run

[JavaScript Code]
1: var o = {};  
2: if (...) o.x = 1; // S1  
3: o.y = 2; // S2  
4: print(o.y); // L1

[ICVector]
<table>
<thead>
<tr>
<th>Site</th>
<th>S1</th>
<th>S2</th>
<th>L1</th>
<th>…</th>
</tr>
</thead>
</table>

[Hidden Class]

[C]
<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>

[Handler]

[RIC Information]

[Hidden Class Validation Table]

<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC_Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>(L1, H2)</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>(L1, H2)</td>
</tr>
</tbody>
</table>

[Triggering Object Access Site Table]

<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>(-, 0)</td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>(0, 1)</td>
<td></td>
</tr>
</tbody>
</table>

[Context-Independent Handlers]

[H2]
| ret obj[0] | |

DS: Dependent Site, S/B: Site or Built-in HC
Example: Step 3: Reuse Run

[JavaScript Code]
1: var o = {};  
2: if (...) o.x = 1; // S1  
3: o.y = 2; // S2  
4: print(o.y); // L1

[ICVector]
<table>
<thead>
<tr>
<th>Site</th>
<th>S1</th>
<th>S2</th>
<th>L1</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>0</td>
<td>C</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>Addr</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

[Hidden Class]
<table>
<thead>
<tr>
<th>[C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
</tr>
</tbody>
</table>

[Handler]

[RIC Information]

[Hidden Class Validation Table]

<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC.Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>(L1, H2)</td>
</tr>
</tbody>
</table>

[Triggering Object Access Site Table]

<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>(0, 1)</td>
</tr>
</tbody>
</table>

[Context-Independent Handlers]

[H2] ret obj[0]

DS: Dependent Site, S/B: Site or Built-in HC
Example:
Step 3: Reuse Run

[JavaScript Code]
1: var o = {};  
2: if (...) o.x = 1; // S1  
3: o.y = 2; // S2  
4: print(o.y); // L1  

[ICVector]

<table>
<thead>
<tr>
<th>Site</th>
<th>S1</th>
<th>S2</th>
<th>L1</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Incoming HC Addr</th>
<th>...</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Handler</th>
<th>...</th>
</tr>
</thead>
</table>

[Hidden Class]

[C]

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>

[Handler]

[RIC Information]

[Hidden Class Validation Table]

<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC_Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>(L1, H2)</td>
</tr>
</tbody>
</table>

[Triggering Object Access Site Table]

<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>(-, 0)</td>
</tr>
<tr>
<td>S2</td>
<td>(0, 1)</td>
</tr>
</tbody>
</table>

[Context-Independent Handlers]

[H2]

ret obj[0]

DS: Dependent Site, S/B: Site or Built-in HC
Example:
Step 3: Reuse Run

[JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

[ICVector]

| Site   | S1 | S2 | L1 | ...
|--------|----|----|----|------
| Incoming Addr |    |    |    |      
| Handler |    |    |    |      |

[Hidden Class]

[C]
<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>

[D]
<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>

[Handler]

[H3]
obj.HC = D
obj[0] = 2

[RIC Information]

[Hidden Class Validation Table]

<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC_Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>1</td>
<td>(L1, H2)</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Triggering Object Access Site Table]

<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>(-, 0)</td>
</tr>
<tr>
<td>S2</td>
<td>(0, 1)</td>
</tr>
</tbody>
</table>

[Context-Independent Handlers]

[H2]
ret obj[0]

DS: Dependent Site, S/B: Site or Built-in HC
Example: Step 3: Reuse Run

[JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2;       // S2
4: print(o.y);    // L1

[IC Vector]
| Site | S1 | S2 | L1 | ...
|------|----|----|----|-----
| HC   |    |    |    |      
| Addr |    |    |    |      
| Handler |    | H3 |    |      

[Hidden Class]
<table>
<thead>
<tr>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

[D]
<table>
<thead>
<tr>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

[RIC Information]
obj.HC = D
obj[0] = 2

[Hidden Class Validation Table]
<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC_Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>(L1, H2)</td>
</tr>
</tbody>
</table>

[Triggering Object Access Site Table]
<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>(-, 0)</td>
</tr>
<tr>
<td>S2</td>
<td>(0, 1)</td>
</tr>
</tbody>
</table>

[Context-Independent Handlers]
ret obj[0]

DS: Dependent Site, S/B: Site or Built-in HC
## Example: Step 3: Reuse Run

### JavaScript Code
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

### ICVector

| Site | S1 | S2 | L1 | ...
|------|----|----|----|----
| Incoming Addr | C |    |    | ...
| Handler      | H3 |    |    | ...

### Hidden Class

[C]

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>

[D]

<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>0</td>
</tr>
</tbody>
</table>

### Handler

[H3]

obj.HC = D
obj[0] = 2

### RIC Information

Example:

Step 3: Reuse Run

### Hidden Class Validation Table

<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC_Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>1</td>
<td>(DS, Handler)</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>(L1, H2)</td>
</tr>
</tbody>
</table>

### Triggering Object Access Site Table

<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>0</td>
<td>(0, 1)</td>
</tr>
</tbody>
</table>

### Context-Independent Handlers

[H2]

ret obj[0]

---

DS: Dependent Site, S/B: Site or Built-in HC
Example:
Step 3: Reuse Run

**[JavaScript Code]**
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

**[ICVector]**

| Site   | S1 | S2 | L1 | ...
|--------|----|----|----|----
| Incoming HC Addr | C |     |    | ...
| Handler         | H3 |     |    | ...

**[Hidden Class]**

<table>
<thead>
<tr>
<th>C</th>
<th>[Property Offset]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D</th>
<th>[Property Offset]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**[Hidden Class Validation Table]**

<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC.Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td><strong>1</strong></td>
<td><strong>D</strong></td>
<td><strong>1</strong></td>
<td><strong>(L1, H2)</strong></td>
</tr>
</tbody>
</table>

**[Triggering Object Access Site Table]**

<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>(-, 0)</td>
</tr>
<tr>
<td>S2</td>
<td>(0, 1)</td>
</tr>
</tbody>
</table>

**[Context-Independent Handlers]**

<table>
<thead>
<tr>
<th>H2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ret obj[0]</td>
</tr>
</tbody>
</table>

**[RIC Information]**

- IC Miss
- IC Miss
- DS: Dependent Site, S/B: Site or Built-in HC
Example:
Step 3: Reuse Run

**[JavaScript Code]**
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

**[ICVector]**
| Site | S1  | S2  | L1  | ...
|------|-----|-----|-----|------
| Incoming Addr | C  |     |     | ...
| Handler | H3  |     |     | ...

**[Hidden Class]**

**[Handler]**

[H3]
obj.HC = D
obj[0] = 2

**[RIC Information]**

**[Hidden Class Validation Table]**

<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC_Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>D</td>
<td>1</td>
<td>(L1, H2)</td>
</tr>
</tbody>
</table>

**[Triggering Object Access Site Table]**

<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>(-, 0)</td>
</tr>
<tr>
<td>S2</td>
<td>(0, 1)</td>
</tr>
</tbody>
</table>

**[Context-Independent Handlers]**

[H2]
ret obj[0]

DS: Dependent Site, S/B: Site or Built-in HC
Example:
Step 3: Reuse Run

[JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2;   // S2
4: print(o.y); // L1

[ICVector]

| Site | S1 | S2 | L1 | ...
|------|----|----|----|---
| Incoming Addr | C | D | ...
| Handler | H3 | H2 | ...

[Hidden Class]

[C]
Property | Offset
--- | ---

[D]
Property | Offset
--- | ---
y | 0

[Handler]

[H3]
obj.HC = D
obj[0] = 2

[RIC Information]

[Hidden Class Validation Table]

<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC_Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>D</td>
<td>1</td>
<td>(L1, H2)</td>
</tr>
</tbody>
</table>

[Triggering Object Access Site Table]

<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>(-, 0)</td>
</tr>
<tr>
<td>S2</td>
<td>(0, 1)</td>
</tr>
</tbody>
</table>

[Context-Independent Handlers]

[H2]
ret obj[0]

DS: Dependent Site, S/B: Site or Built-in HC
Example:
Step 3: Reuse Run

[JavaScript Code]
1: var o = {};
2: if (...) o.x = 1; // S1
3: o.y = 2; // S2
4: print(o.y); // L1

[ICVector]
| Site | S1 | S2 | L1 | ...
|------|----|----|----|----
| Incoming HC Addr | C | D | ...
| Handler | H3 | H2 | ...

[Hidden Class]
[C]
<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
</table>

[D]
<table>
<thead>
<tr>
<th>Property</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>0</td>
</tr>
</tbody>
</table>

[Hidden Class Validation Table]
<table>
<thead>
<tr>
<th>HC_ID</th>
<th>HC_Addr</th>
<th>V</th>
<th>(DS, Handler)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>D</td>
<td>1</td>
<td>(L1, H2)</td>
</tr>
</tbody>
</table>

[Triggering Object Access Site Table]
<table>
<thead>
<tr>
<th>S/B</th>
<th>HC_ID Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>(-, 0)</td>
</tr>
<tr>
<td>S2</td>
<td>(0, 1)</td>
</tr>
</tbody>
</table>

[Context-Independent Handlers]
[H2]
ret obj[0]

[RIC Information]
obj.HC = D
obj[0] = 2

DS: Dependent Site, S/B: Site or Built-in HC
Recap of RIC Example

- When handling an IC miss at **Triggering Site** → RIC updates **Dependent Sites** sharing the same HC
  - Saves future IC misses at **Dependent Sites**

- On average, a **Triggering Site** has 4 **Dependent Sites**

- A **Validation Step** confirms reused information is correct by dynamically checking that **Reuse Run** follows the same control path as **Initial Run**
  - See paper for details
Experimental Setup

• Evaluated initialization of 7 popular JS libraries in Google V8

• Evaluated two configurations
  – *Baseline*: **Reuse Run** reuses only bytecodes from *Initial Run*
  – *RIC*: **Reuse Run** reuses bytecodes and RIC information extracted from *Initial Run*
Evaluation Results

- On average, reduced # of IC misses by 53%
- On average, reduced # of instructions by 15%
Execution Time

- On average, reduced initialization time of JS libraries by 17%
- Saved up to 56 ms (17 ms on average)
Conclusion

• Characterized JS library initialization
  – Execution is mostly deterministic across runs
  – Same hidden class is encountered by multiple object access sites
  – Most handlers are context-independent

• Proposed **Reusable Inline Caching (RIC)** to reuse IC information across JS executions

• Evaluated 7 popular JS libraries in Google V8
  – Reduced # of IC misses by 53%
  – Reduced # of instructions by 15%
  – Reduced initialization time by 17%
Reusable Inline Caching for JavaScript Performance

Jiho Choi, Thomas Shull, and Josep Torrellas
Department of Computer Science
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

PLDI 2019