How JS engine optimize JavaScript code

JavaScript runs in web browsers, node js, reactiveness, IT devices and so on.

There are different types of JavaScript engines. To mention some of them V8, Spider Monkey, Chakra, JavaScript Core( JSC).

The JavaScript Pipeline

It all starts with the JavaScript code you write. The JavaScript engine parses the source code and turns it into an Abstract Syntax Tree (AST). Based on that AST, the interpreter can start to do its thing and produce bytecode. Great! At that point the engine is actually running the JavaScript code.

To make it run faster, the bytecode can be sent to the optimizing compiler along with profiling data. The optimizing compiler makes certain assumptions based on the profiling data it has, and then produces highly optimized machine code.

I f at some point one of the assumptions turns out to be incorrect, the optimizing compiler deoptimizes and goes back to the interpreter.

The interpreter generates unoptimized bytecode quickly, and the optimizing compiler takes a little longer but eventually produces highly-optimized machine code.

Optimizing

Shapes

All JavaScript engines use shapes as an optimization, but they don’t all call them shapes:

V8 🡪Maps

Chakra 🡪Types (typesof)

JavaScriptCore 🡪 Structures

SpiderMonkey 🡪Shapes

JavaScript engines can optimize object property access based on the object’s shape.

Object = {

X=10;

Y=20;

}

|  |
| --- |
| Object |
| “X” |
| “Y” |

We can access the properties for both Object X and Y and looks in the objects key x and y then loads the corresponding property attributes, and finally returns the [[Value]].

Because we don’t want to waste memory we don’t store it in JSObject and to remove unnecessary duplication. As an optimization, engines store the shape of the object separately

|  |
| --- |
| JSObject |
| “10” |
| “20” |

|  |
| --- |
| Shape |
| “X” |
| “Y” |

A= {X:5; Y:10};

B= {X:10; Y:20};

|  |
| --- |
| JSObject A |
| “5” |
| “10” |

|  |
| --- |
| Shape |
| “X” |
| “Y” |



|  |
| --- |
| JSObject B |
| “10” |
| “20” |

vTransition chainsin the JavaScript engine.

|  |
| --- |
| Shape(empty) |



|  |
| --- |
| Shape (X) |
| “X” |



|  |
| --- |
| Shape(X,Y) |
| “X” |
| “Y” |



|  |
| --- |
| JSObject |
| “10” |
| “20” |

Object= {};



Object.X=10;

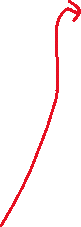
Object.Y= 20;



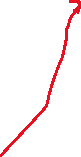
|  |
| --- |
| Shape(empty) |



|  |
| --- |
| Shape(X) |
| “X” |



|  |
| --- |
| Shape(Y) |
| “Y” |



|  |
| --- |
| JSObject object |
| 10 |
| 20 |



We know the advantage when we have multiple objects. No matter how many objects there are, if they have the same shape, we only must store the shape and property information once!



So no need of to store it in JSObject.



Reference https://mathiasbynens.be/notes/shapes-ics