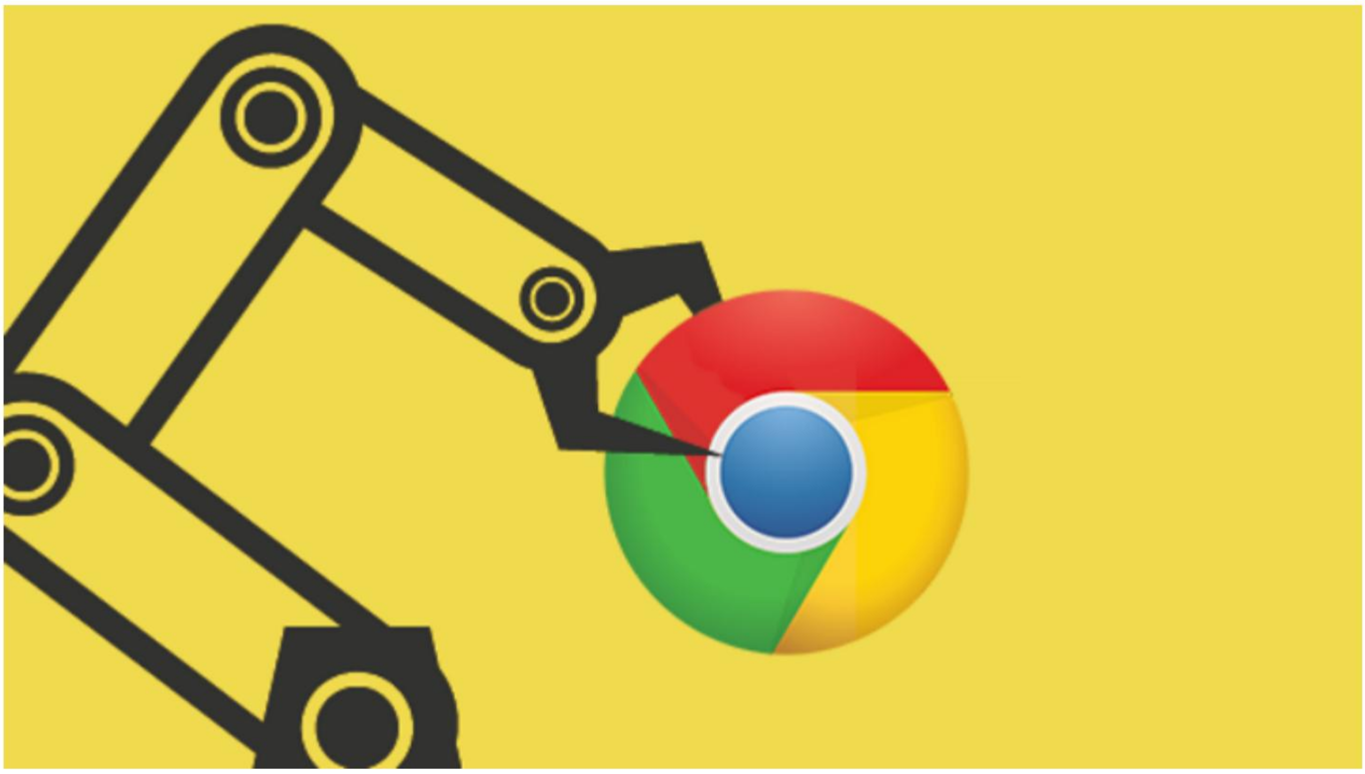
"Chrome V8 Source Code" 38. replace technical details and performance influencing factors



1 Introduction

Strings are an important data type in JavaScript. Their importance is not only reflected in the fact that strings are the most widely used data type, but also in the fact that V8 uses a large number of technical means to modify and optimize string operations. The next few articles will focus on the related operations of strings. This article first explains the source code and related data structures of String.prototype.replace, and then demonstrates the calling, loading and execution process of String.prototype.replace through test cases. We will look at how factors such as string type and length affect the implementation and efficiency of replace. **Note** (1) Sea of Nodes is the leading knowledge of this article. Please refer to Cliff's 1993 paper From Quads to Graphs. (2) The environment used in this article is: V8 7.9, win10 x64, VS2019.

2 String.prototype.replace source code

The test case code is as follows:

```
var str="Visit Microsoft!Visit Microsoft!"; var res=str.replace("Microsoft","Runoob");
console.log(res);
```

replace() is implemented using TF_BUILTIN. The function name of replace() in V8 is StringPrototypeReplace, and the number is 594. The source code is as follows:

Machine Translated by Google

```
TF_BUILTIN(StringPrototypeReplace, StringBuiltinsAssembler) {
Label out(this);
TNode<Object> receiver = CAST(Parameter(Descriptor::kReceiver));
Node* const search = Parameter(Descriptor::kSearch);
Node* const replace = Parameter(Descriptor::kReplace);
TNode<Context> context = CAST(Parameter(Descriptor::kContext));
TNode<Smi> const smi_zero = SmiConstant(0);
RequireObjectCoercible(context, receiver, "String.prototype.replace");
MaybeCallFunctionAtSymbol(/*omitted.....*/);
TNode<String> const subject_string = ToString_Inline(context, receiver);
TNode<String> const search_string = ToString_Inline(context, search);
TNode<IntPtrT> const subject_length =
LoadStringLengthAsWord(subject_string);
TNode<IntPtrT> const search_length = LoadStringLengthAsWord(search_string);
    {
      Label next(this);
      GotoIfNot(WordEqual(search_length, IntPtrConstant(1)), &next);
      GotoIfNot(IntPtrGreaterThan(subject_length, IntPtrConstant(0xFF)), &next);
      GotoIf(TaggedIsSmi(replace), &next);
      GotoIfNot(IsString(replace), &next);
      TNode<Uint16T> const subject_instance_type =
          LoadInstanceType(subject_string);
      GotoIfNot(IsConsStringInstanceType(subject_instance_type), &next);
      GotoIf(TaggedIsPositiveSmi(IndexOfDollarChar(context, replace)), &next);
      Return(CallRuntime(Runtime::kStringReplaceOneCharWithString, context,
                                      subject_string, search_string, replace));
      BIND(&next); }
TNode<Smi> const match_start_index =
        CAST(CallBuiltin(Builtins::kStringIndexOf, context, subject_string,
search_string, smi_zero));
    {
      Label next(this), return_subject(this);
      GotoIfNot(SmiIsNegative(match_start_index), &next);
      GotoIf(TaggedIsSmi(replace), &return_subject);
      GotoIf(IsCallableMap(LoadMap(replace)), &return_subject);
      ToString_Inline(context, replace);
      Goto(&return_subject);
      BIND(&return_subject);
      Return(subject_string);
      BIND(&next); }
TNode<Smi> const match_end_index = SmiAdd(match_start_index,
SmiFromIntPtr(search_length));
VARIABLE(var_result, MachineRepresentation::kTagged, EmptyStringConstant());
    {
      Label next(this);
      GotoIf(SmiEqual(match_start_index, smi_zero), &next);
      TNode<Object> const prefix =
          CallBuiltin(Builtins::kStringSubstring, context, subject_string,
                                 IntPtrConstant(0), SmiUntag(match_start_index));
      var_result.Bind(prefix);
      Goto(&next);
      BIND(&next); }
Label if_iscallablereplace(this), if_notcallablereplace(this);
```

Machine Translated by Google

```
52. GotoIf(TaggedIsSmi(replace), &if_notcallablereplace);
53. Branch(IsCallableMap(LoadMap(replace)), &if_iscallablereplace,
54.                  &if_notcallablereplace);
55. BIND(&if_iscallablereplace);
56. {
57.          Callable call_callable = CodeFactory::Call(isolate());
58.          Node* const replacement =
59.                 CallJS(call_callable, context, replace, UndefinedConstant(),
60.                             search_string, match_start_index, subject_string);
61.          TNode<String> const replacement_string =
62.                 ToString_Inline(context, replacement);
63.          var_result.Bind(CallBuiltin(Builtins::kStringAdd_CheckNone, context,
64.                                                    var_result.value(), replacement_string));
65.          Goto(&out); }
66. BIND(&if_notcallablereplace);
67.       {
68.          TNode<String> const replace_string = ToString_Inline(context, replace);
69.          Node* const replacement =
70.                 GetSubstitution(context, subject_string, match_start_index,
71.                                           match_end_index, replace_string);
72.          var_result.Bind(CallBuiltin(Builtins::kStringAdd_CheckNone, context,
73.                                                    var_result.value(), replacement));
74.          Goto(&out);}
75. BIND(&out);
76. {
77.          TNode<Object> const suffix =
78.                 CallBuiltin(Builtins::kStringSubstring, context, subject_string,
79.                                     SmiUntag(match_end_index), subject_length);
80.          TNode<Object> const result = CallBuiltin(
81.                 Builtins::kStringAdd_CheckNone, context, var_result.value(), suffix);
82.            Return(result);
83.        }}
```

The code receiver in line 3 of the above code represents the test case string "Visit Microsoft!Visit Microsoft!";

The 4th line of code search represents the test case string "Microsoft";

Line 5 of code replace represents the test case string "Runoob";

Line 9 of code uses MaybeCallFunctionAtSymbol() to implement the replace function when the search is a regular expression.

The replace source code is divided into five sub-functions and explained separately below:

(1) Function 1: The receiver length is greater than 0xFF, the search length is greater than 1, and replace is ConsString. These three conditions are true at the same time.

hour

Lines 10-13 of code convert the data types of search and replace and calculate their lengths;

Line 16 of code determines whether the length of search is equal to 1. If not, jumps to line 26;

The 17th line of code determines whether the length of the receiver is greater than 0xFF, and if it is less than or equal to it, it jumps to line 26;

Lines 18-19 determine whether replace is a small integer or replace is not a string. If the result is true, jump to line 26;

Lines 20-22 determine whether replace is of type ConsString. If the result is false, jump to line 26 ; **prompt that** ConsString is not an independent

A string, which is a string pair that uses pointers to splice two strings together. In V8, the result of adding two strings is often

A pair of strings of type ConsString.

Line 23 determines the PositiveSmi type;

Line 24 uses Runtime::kStringReplaceOneCharWithString() to complete the replacement.

(2) Function 2: Calculate the prefix string in receiver

Machine Translated by Google

Lines 27-32 calculate the position match_start_index of the first character of search in the receiver, if match_start_index is not negative number, jump to line 39;

Lines 33-35 determine whether replace is SMI or a function, if so, jump to line 37;

Line 40 calculates match_end_index;

In line 44, if match_start_index = 0 (that is, search[0]=receiver[0]), jump to line 49;

Line 45 takes out the characters before the match_start_index position in the receiver and saves them as prefix; that is, getting the "Visit" in the test case string;

Lines 50-54 judge whether replace is SMI or a function, and jump to the corresponding line number based on the judgment result.

(3) Function 3: replace is a function type

Lines 58-63 calculate the result of replace, and concatenate the result after prefix to form a new string var_result;

(4) Function 4: replace is a string type

Lines 58-72 splice replace after prefix to form a new string var_result;

(5) Calculate the suffix string in receiver

Lines 77-82 take out the string suffix after match_end_index in the receiver, splice suffix after var_result to form and return New string, replacement completed.

The following is a brief explanation of the runtime method used in replace:

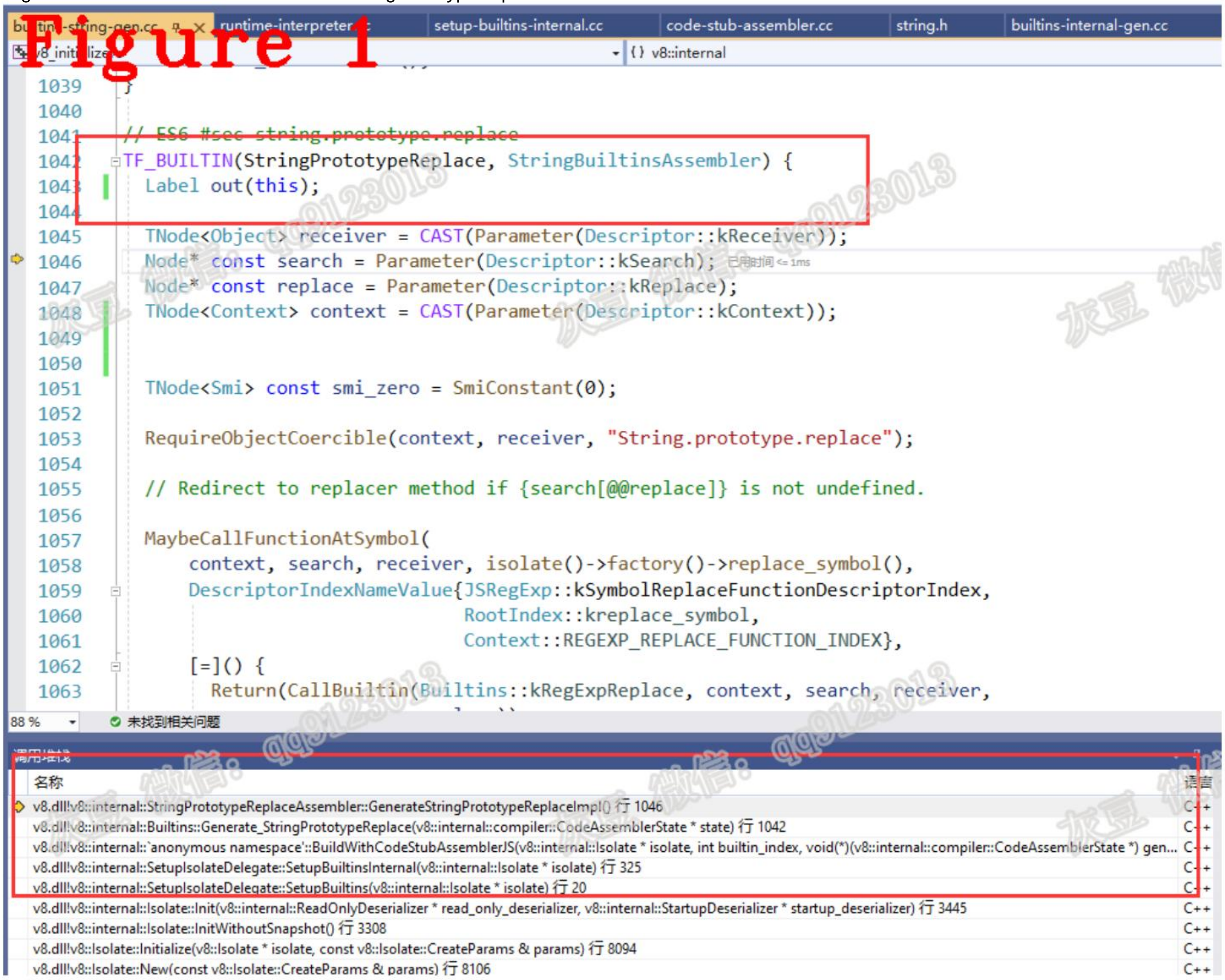
```
1. RUNTIME_FUNCTION(Runtime_StringReplaceOneCharWithString) {
2.         HandleScope scope(isolate);
3.         DCHECK_EQ(3, args.length());
4.         CONVERT_ARG_HANDLE_CHECKED(String, subject, 0);
5.         CONVERT_ARG_HANDLE_CHECKED(String, search, 1);
6.         CONVERT_ARG_HANDLE_CHECKED(String, replace, 2);
7.         const int kRecursionLimit = 0x1000;
8.         bool found = false;
9.         Handle<String> result;
10.          if (StringReplaceOneCharWithString(isolate, subject, search, replace,
&found,
11.                                                                    kRecursionLimit).ToHandle(&result)) {
12.              return *result;
13.          }
14.          if (isolate->has_pending_exception())
15.              return ReadOnlyRoots(isolate).exception();
16.          subject = String::Flatten(isolate, subject);
17.          if (StringReplaceOneCharWithString(isolate, subject, search, replace,
&found,
18.                                                                    kRecursionLimit).ToHandle(&result)) {
19.              return *result;
20.          }
twenty one.  if (isolate->has_pending_exception())
twenty two.      return ReadOnlyRoots(isolate).exception();
twenty three. return isolate->StackOverflow();
twenty four. }
```

The 10th step in the above code executes the StringReplaceOneCharWithString method, which implements the replace function;

When the code in line 14 detects an abnormal situation, it first executes String::Flatten, processes the ConsString characters into a single string, and then executes it again. StringReplaceOneCharWithString method.

Machine Translated by Google

Figure 1 shows the function call stack of StringPrototypeReplace.



```
}

// ES6 #sec-string.prototype.replace
TF_BUILTIN(StringPrototypeReplace, StringBuiltinsAssembler) {
  Label out(this);

  TNode<Object> receiver = CAST(Parameter(Descriptor::kReceiver));
  Node* const search = Parameter(Descriptor::kSearch);
  Node* const replace = Parameter(Descriptor::kReplace);
  TNode<Context> context = CAST(Parameter(Descriptor::kContext));


  TNode<Smi> const smi_zero = SmiConstant(0);

  RequireObjectCoercible(context, receiver, "String.prototype.replace");

  // Redirect to replacer method if {search[@@replace]} is not undefined.

  MaybeCallFunctionAtSymbol(
      context, search, receiver, isolate()->factory()->replace_symbol(),
      DescriptorIndexNameValue{JSRegExp::kSymbolReplaceFunctionDescriptorIndex,
                               RootIndex::kreplace_symbol,
                               Context::REGEXP_REPLACE_FUNCTION_INDEX},
      [=]() {
        Return(CallBuiltin(Builtins::kRegExpReplace, context, search, receiver,
```

```
v8.dll!v8::internal::StringPrototypeReplaceAssembler::GenerateStringPrototypeReplaceImpl() 行 1046    C++
v8.dll!v8::internal::Builtins::Generate_StringPrototypeReplace(v8::internal::compiler::CodeAssemblerState * state) 行 1042    C++
v8.dll!v8::internal::`anonymous namespace'::BuildWithCodeStubAssemblerJS(v8::internal::Isolate * isolate, int builtin_index, void(*)(v8::internal::compiler::CodeAssemblerState *) gen...    C++
v8.dll!v8::internal::SetupIsolateDelegate::SetupBuiltinsInternal(v8::internal::Isolate * isolate) 行 325    C++
v8.dll!v8::internal::SetupIsolateDelegate::SetupBuiltins(v8::internal::Isolate * isolate) 行 20    C++
v8.dll!v8::internal::Isolate::Init(v8::internal::ReadOnlyDeserializer * read_only_deserializer, v8::internal::StartupDeserializer * startup_deserializer) 行 3445    C++
v8.dll!v8::internal::Isolate::InitWithoutSnapshot() 行 3308    C++
v8.dll!v8::Isolate::Initialize(v8::Isolate * isolate, const v8::Isolate::CreateParams & params) 行 8094    C++
v8.dll!v8::Isolate::New(const v8::Isolate::CreateParams & params) 行 8106    C++
```

String.prototype.replace test

The bytecode of the test case is as follows:

```
1. //Omit........
2. 000001A2EE982AC6 @ 16 : 12 01 3.                LdaConstant [1]
000001A2EE982AC8 @ 18 : 15 02 04 4.                StaGlobal [2], [4]
000001A2EE982ACB @ 21 : 13 02 00 5.                LdaGlobal [2], [0]
000001A2EE982ACE @ 24 : 26 f9 6.                   Star r2
000001A2EE982AD0 @ 26 : 29 f9 03 7.                LdaNamedPropertyNoFeedback r2, [3]
000001A2EE982AD3 @ 29 : 26 fa 8.                   Star r1
000001A2EE982AD5 @ 31 : 12 04 9.                   LdaConstant [4]
000001A2EE982AD7 @ 33 : 26 f8 10.                  Star r3
000001A2EE982AD9 @ 35 : 12 05 11.                  LdaConstant [5]
000001A2EE982ADB @ 37 : 26 f7 12. 00               Star r4
0001A2EE982ADD @ 39 : 5f fa f9 03 13.              CallNoFeedback r1, r2-r4
000001A2EE982AE1 @ 43 : 15 06 06 14.               StaGlobal [6], [6]
000001A2EE982AE4 @ 46 : 13 07 08 15.               LdaGlobal [7], [8]
000001A2EE982AE7 @ 49 : 26 f9 16.                  Star r2
000001A2EE982AE9 @ 51 : 29 f9 08                   LdaNamedPropertyNoFeedback r2,
[8]
17. 000001A2EE982AEC @ 54 : 26 fa 18.                     Star r1
000001A2EE982AEE @ 56 : 13 06 02 19.                      LdaGlobal [6], [2]
000001A2EE982AF1 @ 59 : 26 f8 20.                         Star r3
000001A2EE982AF3 @ 61 : 5f fa f9 02 21. 000001A2EE982AF7  CallNoFeedback r1, r2-r3
@ 65 : 26 fb 22. 000001A2EE982AF9 @ 67 : ab 23.           Star r0
Constant pool (size = 9)                                  Return

24. 000001A2EE982A29: [FixedArray] in OldSpace
25.        - map: 0x022d5e100169 <Map>
26.        - length: 9
27. 0: 0x01a2ee9829c9 <FixedArray[8]>
28. 1: 0x01a2ee9828c1 <String[#32]: Visit Microsoft!Visit Microsoft!>
29. 2: 0x01a2ee9828a9 <String[#3]: str>
30. 3: 0x02749a2ab821 <String[#7]: replace>
31. 4: 0x01a2ee982909 <String[#9]: Microsoft>
32. 5: 0x01a2ee982929 <String[#6]: Runoob>
33. 6: 0x01a2ee9828f1 <String[#3]: res>
34. 7: 0x02749a2b3699 <String[#7]: console>
35. 8: 0x02749a2b2cd9 <String[#3]: log>
```

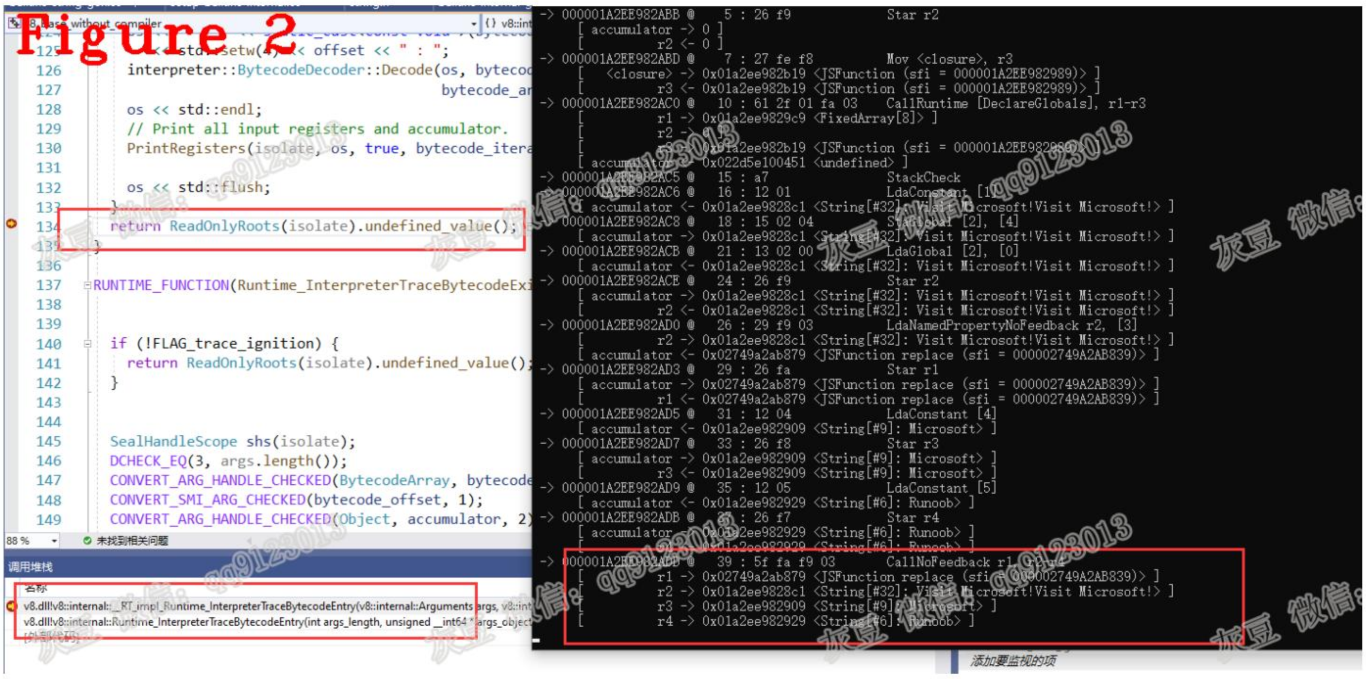
In the above code, lines 2-5 read the string "Visit Microsoft!Visit Microsoft!" and save it to the r2 register;

Lines 6-7 obtain the replace function address and save it to the r1 register;

Lines 8-11 read "Microsoft" and "Runoob" and save them to the r3 and r4 registers respectively;

Line 12 calls the repalce method;

Figure 2 shows the call stack of CallNoFeedback.



```
          << std::setw(4) << offset << " : ";
      interpreter::BytecodeDecoder::Decode(os, bytecod
                                          bytecode_ar

      os << std::endl;
      // Print all input registers and accumulator.
      PrintRegisters(isolate, os, true, bytecode_itera

      os << std::flush;
    }
    return ReadOnlyRoots(isolate).undefined_value();
  }

RUNTIME_FUNCTION(Runtime_InterpreterTraceBytecodeExi


    if (!FLAG_trace_ignition) {
      return ReadOnlyRoots(isolate).undefined_value();
    }


    SealHandleScope shs(isolate);
    DCHECK_EQ(3, args.length());
    CONVERT_ARG_HANDLE_CHECKED(BytecodeArray, bytecode
    CONVERT_SMI_ARG_CHECKED(bytecode_offset, 1);
    CONVERT_ARG_HANDLE_CHECKED(Object, accumulator, 2)
```

```
v8.dll!v8::internal::__RT_impl_Runtime_InterpreterTraceBytecodeEntry(v8::internal::Arguments args, v8::int
v8.dll!v8::internal::Runtime_InterpreterTraceBytecodeEntry(int args_length, unsigned __int64 * args_object
```

```
-> 000001A2EE982ABB @    5 : 26 f9             Star r2
      [ accumulator -> 0 ]
      [          r2 <- 0 ]
-> 000001A2EE982ABD @    7 : 27 fe f8          Mov <closure>, r3
      [   <closure> -> 0x01a2ee982b19 <JSFunction (sfi = 000001A2EE982989)> ]
      [          r3 <- 0x01a2ee982b19 <JSFunction (sfi = 000001A2EE982989)> ]
-> 000001A2EE982AC0 @   10 : 61 2f 01 fa 03    CallRuntime [DeclareGlobals], r1-r3
      [          r1 -> 0x01a2ee9829c9 <FixedArray[8]> ]
      [          r2 -> 0 ]
      [          r3 -> 0x01a2ee982b19 <JSFunction (sfi = 000001A2EE982989)> ]
      [ accumulator <- 0x022d5e100451 <undefined> ]
-> 000001A2EE982AC5 @   15 : a7                StackCheck
-> 000001A2EE982AC6 @   16 : 12 01             LdaConstant [1]
      [ accumulator <- 0x01a2ee9828c1 <String[#32]: Visit Microsoft!Visit Microsoft!> ]
-> 000001A2EE982AC8 @   18 : 15 02 04          StaGlobal [2], [4]
      [ accumulator -> 0x01a2ee9828c1 <String[#32]: Visit Microsoft!Visit Microsoft!> ]
-> 000001A2EE982ACB @   21 : 13 02 00          LdaGlobal [2], [0]
      [ accumulator <- 0x01a2ee9828c1 <String[#32]: Visit Microsoft!Visit Microsoft!> ]
-> 000001A2EE982ACE @   24 : 26 f9             Star r2
      [ accumulator -> 0x01a2ee9828c1 <String[#32]: Visit Microsoft!Visit Microsoft!> ]
      [          r2 <- 0x01a2ee9828c1 <String[#32]: Visit Microsoft!Visit Microsoft!> ]
-> 000001A2EE982AD0 @   26 : 29 f9 03          LdaNamedPropertyNoFeedback r2, [3]
      [          r2 -> 0x01a2ee9828c1 <String[#32]: Visit Microsoft!Visit Microsoft!> ]
      [ accumulator <- 0x02749a2ab879 <JSFunction replace (sfi = 000002749A2AB839)> ]
-> 000001A2EE982AD3 @   29 : 26 fa             Star r1
      [ accumulator -> 0x02749a2ab879 <JSFunction replace (sfi = 000002749A2AB839)> ]
      [          r1 <- 0x02749a2ab879 <JSFunction replace (sfi = 000002749A2AB839)> ]
-> 000001A2EE982AD5 @   31 : 12 04             LdaConstant [4]
      [ accumulator <- 0x01a2ee982909 <String[#9]: Microsoft> ]
-> 000001A2EE982AD7 @   33 : 26 f8             Star r3
      [ accumulator -> 0x01a2ee982909 <String[#9]: Microsoft> ]
      [          r3 <- 0x01a2ee982909 <String[#9]: Microsoft> ]
-> 000001A2EE982AD9 @   35 : 12 05             LdaConstant [5]
      [ accumulator <- 0x01a2ee982929 <String[#6]: Runoob> ]
-> 000001A2EE982ADB @   37 : 26 f7             Star r4
      [ accumulator -> 0x01a2ee982929 <String[#6]: Runoob> ]
      [          r4 <- 0x01a2ee982929 <String[#6]: Runoob> ]
-> 000001A2EE982ADD @   39 : 5f fa f9 03       CallNoFeedback r1, r2-r4
      [          r1 -> 0x02749a2ab879 <JSFunction replace (sfi = 000002749A2AB839)> ]
      [          r2 -> 0x01a2ee9828c1 <String[#32]: Visit Microsoft!Visit Microsoft!> ]
      [          r3 -> 0x01a2ee982909 <String[#9]: Microsoft> ]
      [          r4 -> 0x01a2ee982929 <String[#6]: Runoob> ]
```

Technical summary

(1) The receiver length is greater than 0xFF, the search length is greater than 1 and replace is ConsString. When the three conditions are met at the same time, low efficiency is adopted.

The runtime method is used for processing; **(2)** The principle of replace is to calculate the prefix and suffix, and splice them with newvalue to form the final result.

Okay, that's it for today, see you next time.

Personal abilities are limited and there are shortcomings and mistakes. Criticisms and corrections are welcome.

WeChat: qq9123013 Note: v8 communication Zhihu: www.zhihu.com/people/v8blink

Machine Translated by Google

This article was originally

published by Huidou and reprinted from: https://www.anquanke.com/post/id/

263871 Anquanke - Thoughtful new security media