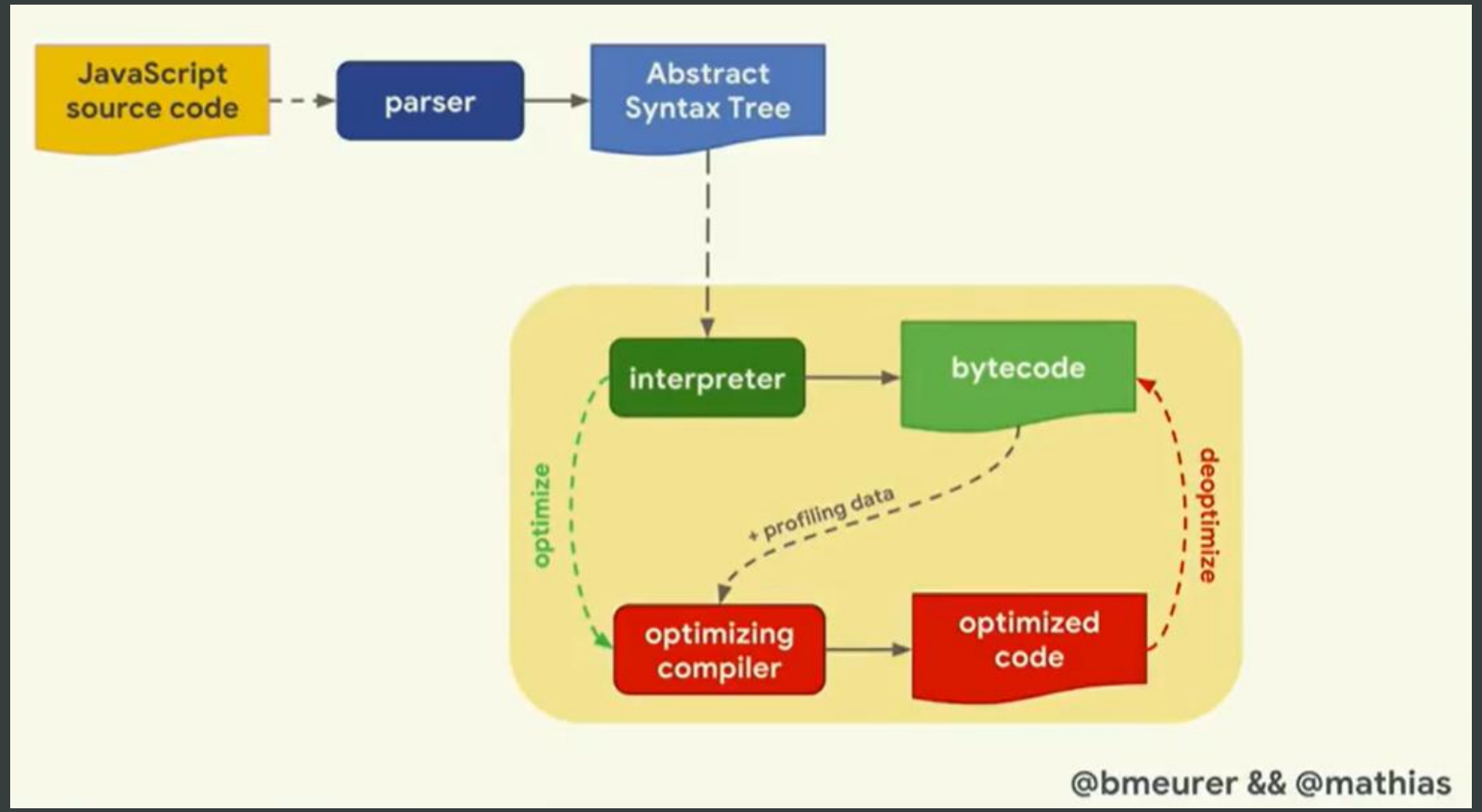
入职报告：
Ignition解释器工作过程

刘铮

V8



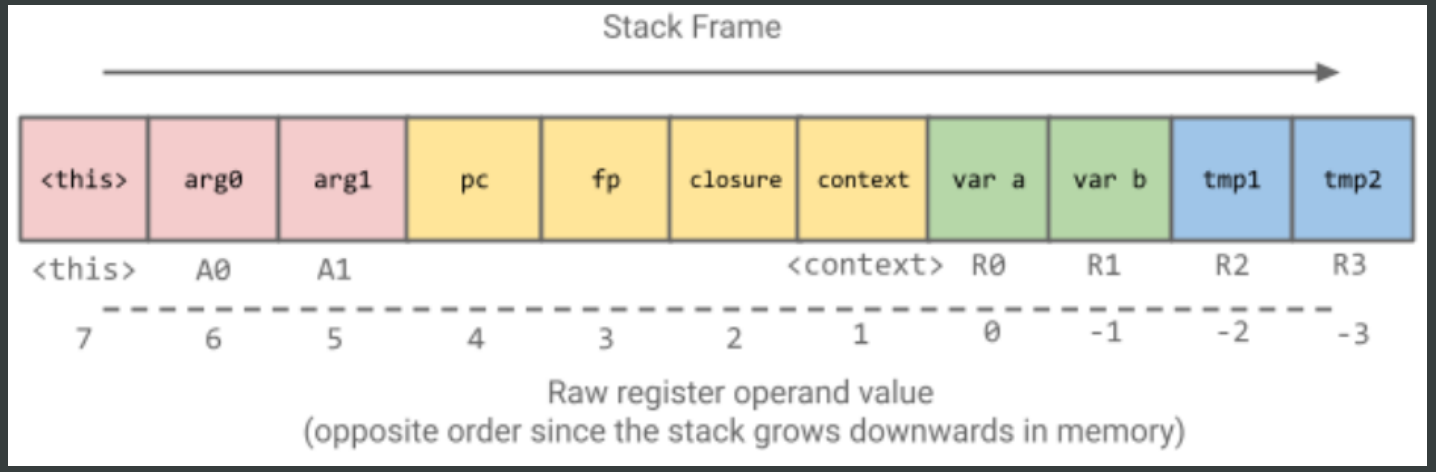
https://www.youtube.com/watch?v=5nmpokoRaZI

生成字节码

- 遍历AST
- 分配寄存器
- 构建常量池

生成字节码

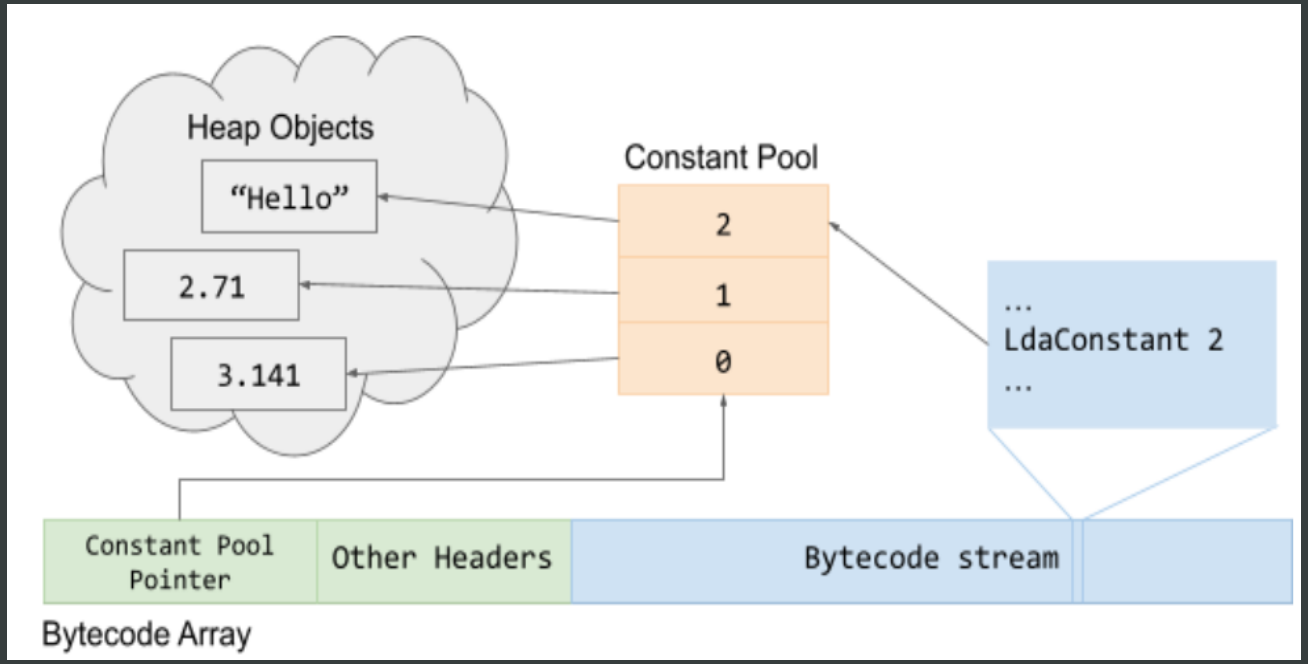
- 遍历AST
- 分配寄存器
- 构建常量池



https://docs.google.com/document/d/11T2CRex9hXxoJwbYqVQ32yIPMh0uouUZLdyrtmMoL44/edit#

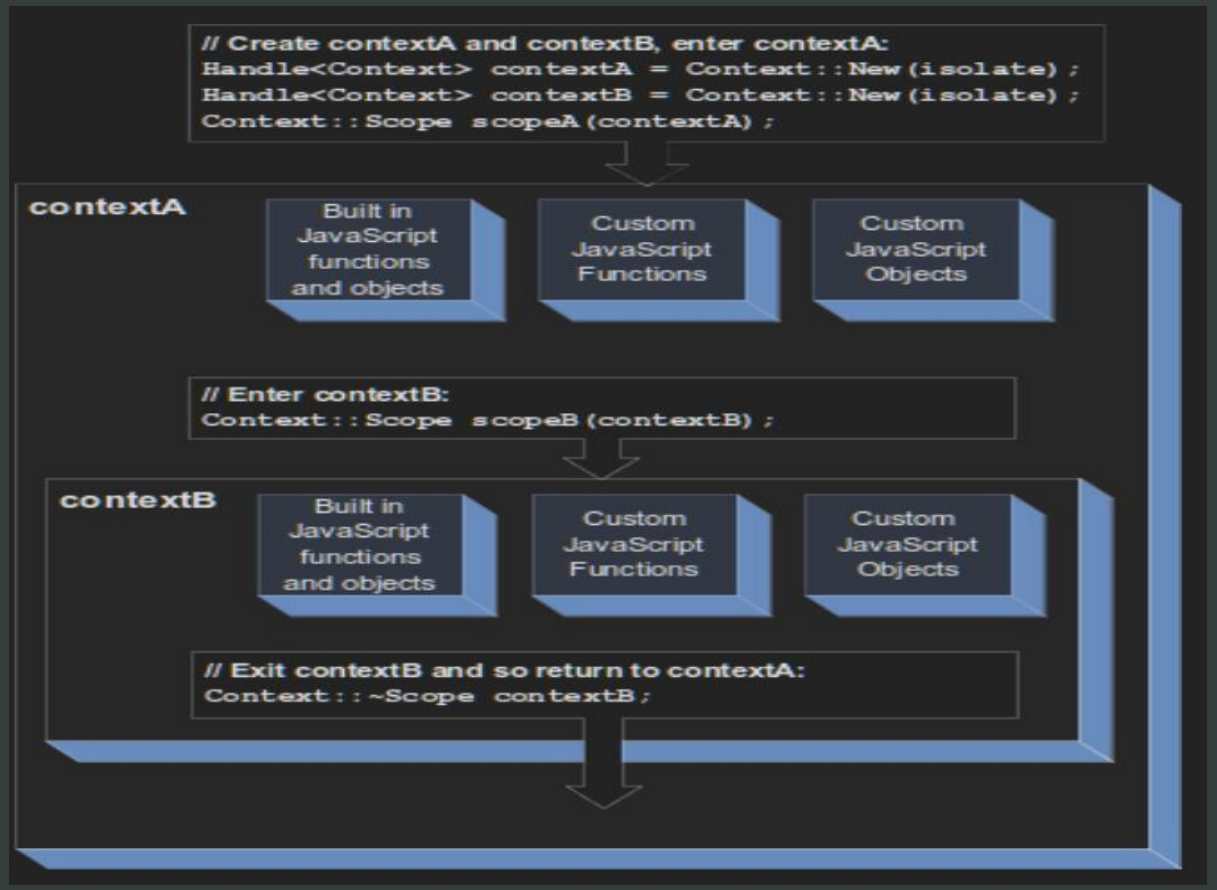
生成字节码

- 遍历AST
- 分配寄存器
- 构建常量池



https://docs.google.com/document/d/11T2CRex9hXxoJwbYqVQ32yIPMh0uouUZLdyrtmMoL44/edit#

跟踪上下文



https://v8.dev/docs/embed

解释执行

- bytecode handler集合

- 只写一次

Global Interpreter Dispatch Table

- 每个isolated实例都有

- 含有code object指针

Bytecode Handler

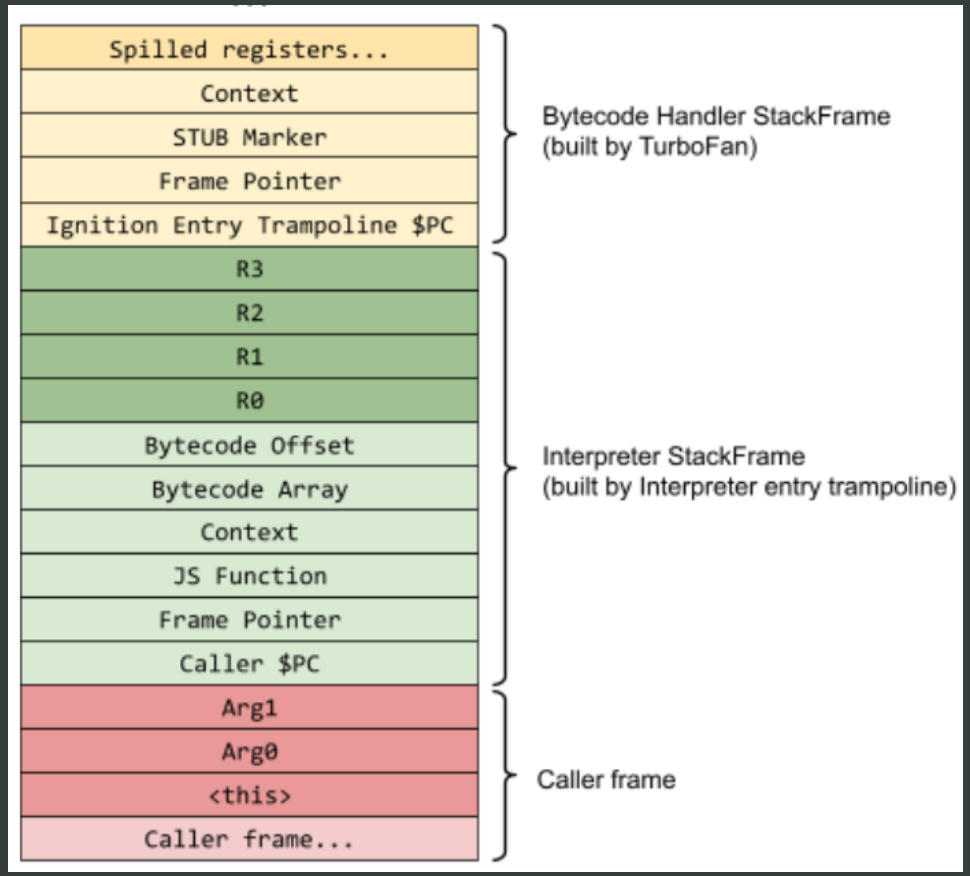
```
// Mov <src> <dst>
//
// Stores the value of register <src> to register <dst>.
void Interpreter::DoMov(InterpreterAssembler* assembler) {
  Node* src_index = __ BytecodeOperandReg(0);
  Node* src_value = __ LoadRegister(src_index);
  Node* dst_index = __ BytecodeOperandReg(1);
  __ StoreRegister(src_value, dst_index);
  __ Dispatch();
}
```

```
// Mov <src> <dst>
//
// Stores the value of register <src> to register <dst>.
IGNITION_HANDLER(Mov, InterpreterAssembler) {
  TNode<Object> src_value = LoadRegisterAtOperandIndex(0);
  StoreRegisterAtOperandIndex(src_value, 1);
  Dispatch();
}
```

http://mshockwave.blogspot.com/2016/03/ignition-interpreter-in-v8-javascript.html

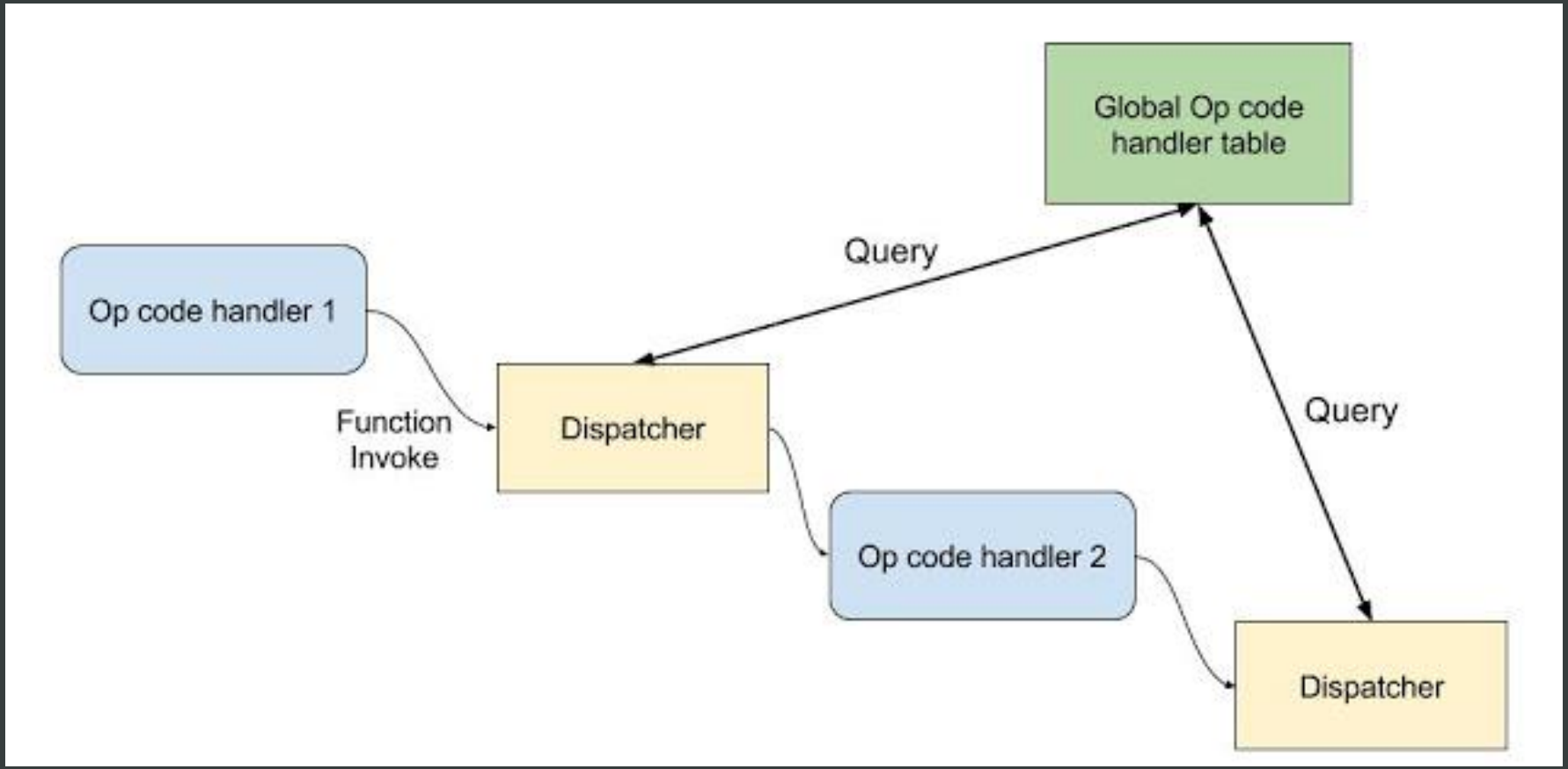
开始

- 函数代码入口：
  - InterpreterEntryTrampoline
- 建立stack frame
- 初始化machine register
- 分派bytecode handler



https://docs.google.com/document/d/11T2CRex9hXxoJwbYqVQ32yIPMh0uouUZLdyrtmMoL44/edit#

执行过程



http://mshockwave.blogspot.com/2016/03/ignition-interpreter-in-v8-javascript.html

执行过程

- Bytecode handler
- Dispatch
- DispatchToBytecodeWithOptionalStarLookahead
- DispatchToBytecode
- DispatchToBytecodeHandlerEntry
- TailCallBytecodeDispatch
- TailCallN
- Next bytecode handler

Disptach

```
// Dispatch to the bytecode.
void InterpreterAssembler::Dispatch() {
  Comment("========= Dispatch");
  DCHECK_IMPLIES(Bytecodes::MakesCallAlongCriticalPath(bytecode_), made_call_);
  // Updates and returns BytecodeOffset() advanced by the current bytecode's
  // size. Traces the exit of the current bytecode.
  TNode<IntPtrT> target_offset = Advance();
  // Load the bytecode at |bytecode_offset|.
  TNode<WordT> target_bytecode = LoadBytecode(target_offset);
  DispatchToBytecodeWithOptionalStarLookahead(target_bytecode);
}
```

DispatchToBytecodeWithOptionalStarLookahead

```
// Dispatches to |target_bytecode| at BytecodeOffset(). Includes short-star
// lookahead if the current bytecode_ is likely followed by a short-star
// instruction.
void InterpreterAssembler::DispatchToBytecodeWithOptionalStarLookahead(
    TNode<WordT> target_bytecode) {

  if (
      // Returns true if the handler for |bytecode| should look ahead and inline a
      // dispatch to a Star bytecode.
      Bytecodes::IsStarLookahead(bytecode_, operand_scale_)) {
    StarDispatchLookahead(target_bytecode);
  }
  DispatchToBytecode(target_bytecode,
                     // Returns the offset from the BytecodeArrayPointer
                     // of the current bytecode.
                     BytecodeOffset());
}
```

DispatchToBytecode

```
// Dispatch to |target_bytecode| at |new_bytecode_offset|.
// |target_bytecode| should be equivalent to loading from the offset.
void InterpreterAssembler::DispatchToBytecode(
    TNode<WordT> target_bytecode, TNode<IntPtrT> new_bytecode_offset) {
  if (FLAG_trace_ignition_dispatches) {
    TraceBytecodeDispatch(target_bytecode);
  }
  //Load a pointer to the target entry in the interpreter dispatch table
  //based on the target bytecode.
  TNode<RawPtrT> target_code_entry = Load<RawPtrT>(
      // Returns a pointer to first entry in the interpreter dispatch table.
      DispatchTablePointer(),
      TimesSystemPointerSize(target_bytecode));

  DispatchToBytecodeHandlerEntry(target_code_entry, new_bytecode_offset);
}
```

DispatchToBytecodeHandlerEntry

```
// Dispatch to the bytecode handler with code entry point |handler_entry|.
void InterpreterAssembler::DispatchToBytecodeHandlerEntry(
    TNode<RawPtrT> handler_entry, TNode<IntPtrT> bytecode_offset) {
  // Propagate speculation poisoning.
  TNode<RawPtrT> poisoned_handler_entry =
      UncheckedCast<RawPtrT>(
          // Poison |value| on speculative paths.
          WordPoisonOnSpeculation(handler_entry));
  TailCallBytecodeDispatch(InterpreterDispatchDescriptor{},
                           poisoned_handler_entry, GetAccumulatorUnchecked(),
                           bytecode_offset,
                           // Returns a pointer to
                           // the current function's BytecodeArray object.
                           BytecodeArrayTaggedPointer(),
                           DispatchTablePointer());
}
```

TailCallBytecodeDispatch

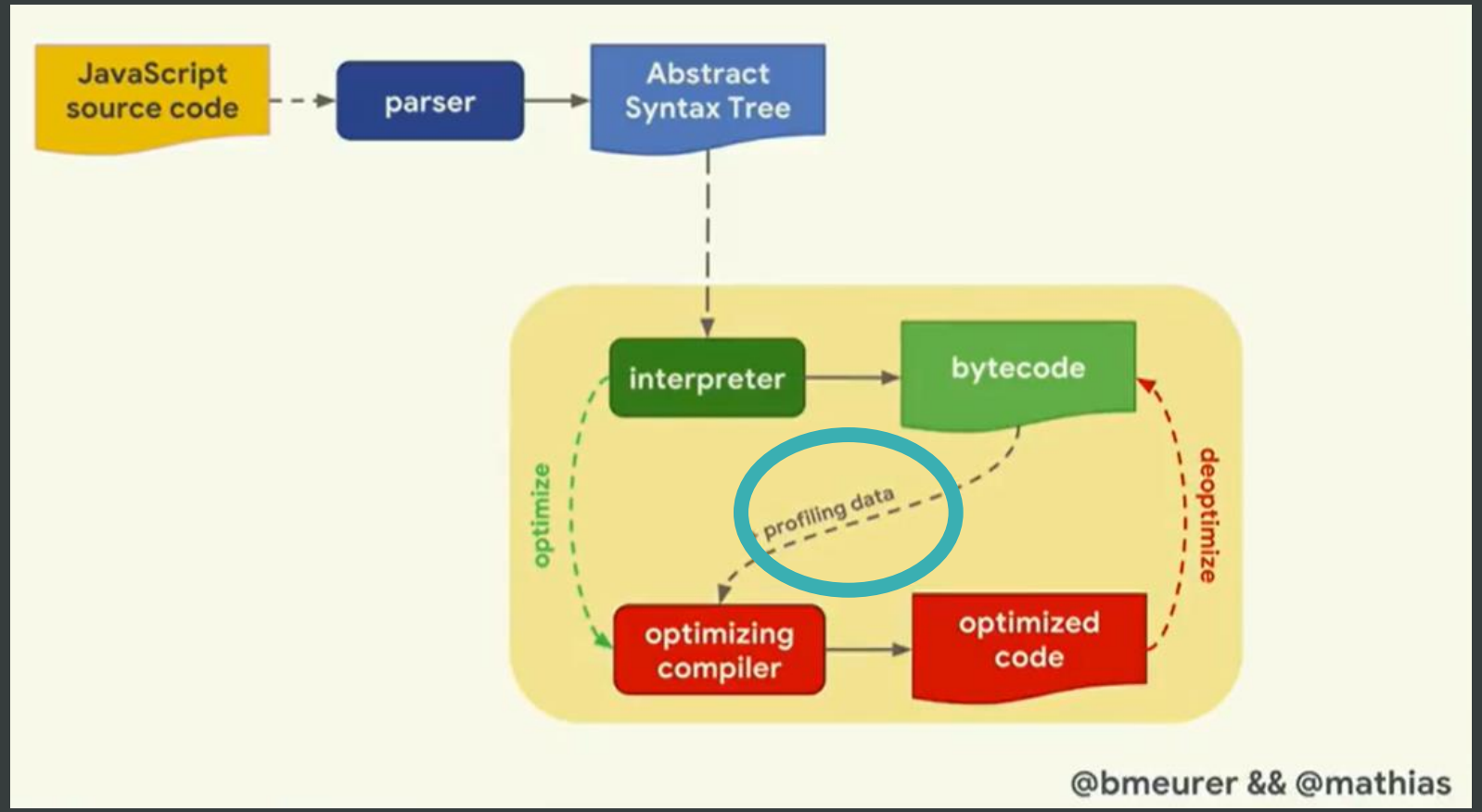
```
template <class... TArgs>
void CodeAssembler::TailCallBytecodeDispatch(
    const CallInterfaceDescriptor& descriptor, TNode<RawPtrT> target,
    TArgs... args) {
  DCHECK_EQ(descriptor.GetParameterCount(), sizeof...(args));
  auto call_descriptor = Linkage::GetBytecodeDispatchCallDescriptor(
      zone(), descriptor, descriptor.GetStackParameterCount());

  Node* nodes[] = {target, args...};
  CHECK_EQ(descriptor.GetParameterCount() + 1, arraysize(nodes));
  raw_assembler()->TailCallN(call_descriptor, arraysize(nodes), nodes);
}
```

TailCallN

```
// Tail call a given call descriptor and the given arguments.
// The call target is passed as part of the {inputs} array.
void RawMachineAssembler::TailCallN(CallDescriptor* call_descriptor,
                                    int input_count, Node* const* inputs) {
  // +1 is for target.
  DCHECK_EQ(input_count, call_descriptor->ParameterCount() + 1);
  Node* tail_call =
      //common(): return a pointer to CommonOperatorBuilder,
      // which builds common operators that can be used at any level of IR
      //TailCall: return an Operator
      MakeNode(common()->TailCall(call_descriptor), input_count, inputs);
  // BasicBlock building: add a tailcall at the end of current execution block {block}.
  schedule()->AddTailCall(CurrentBlock(), tail_call);
  current_block_ = nullptr;
}
```

收集反馈信息



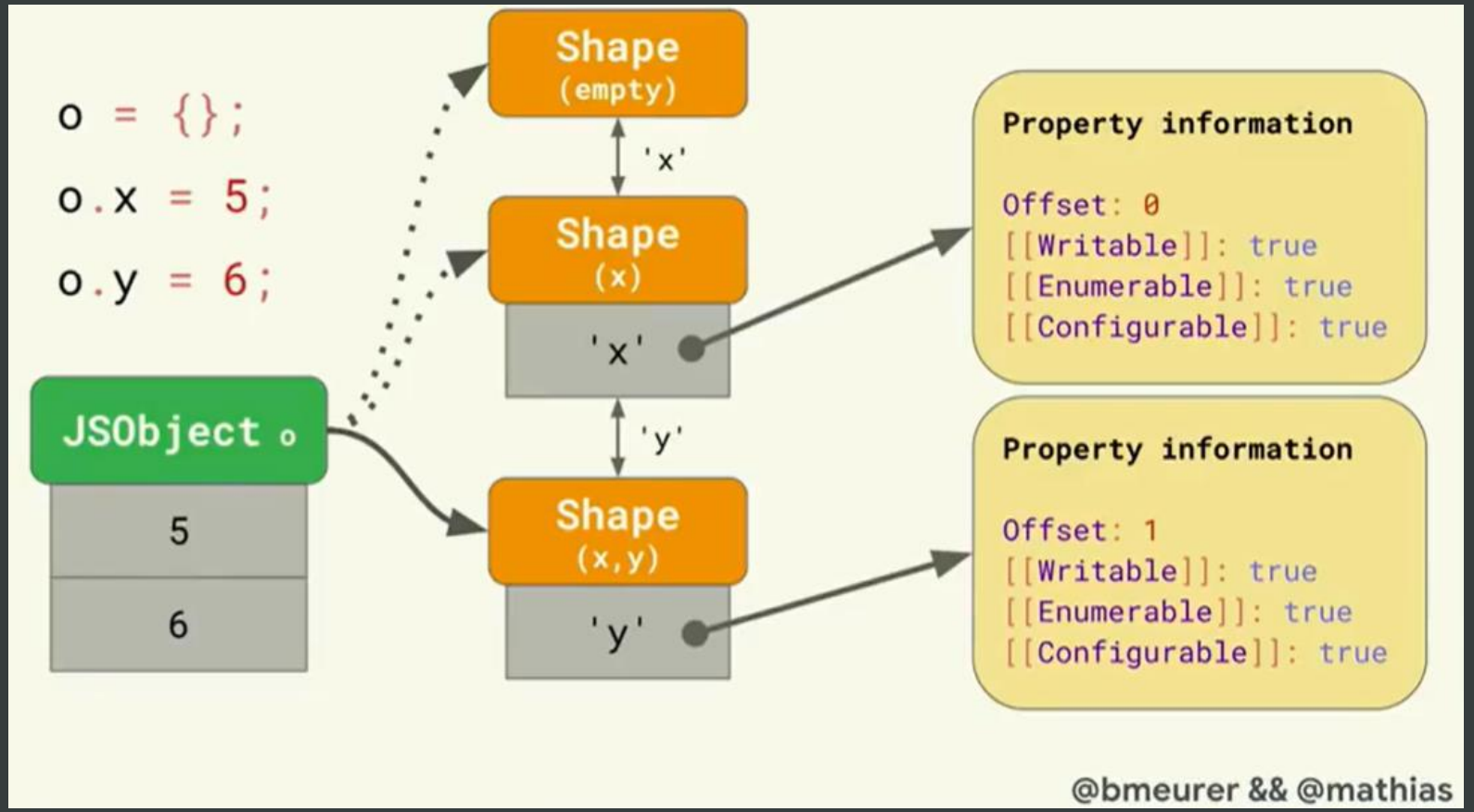
https://www.youtube.com/watch?v=5nmpokoRaZI

TypeFeedbackVector

IC Slot	IC Type	State
...	...	...
10	LOAD	MONO(M)
11	LOAD	UNINITIALIZED

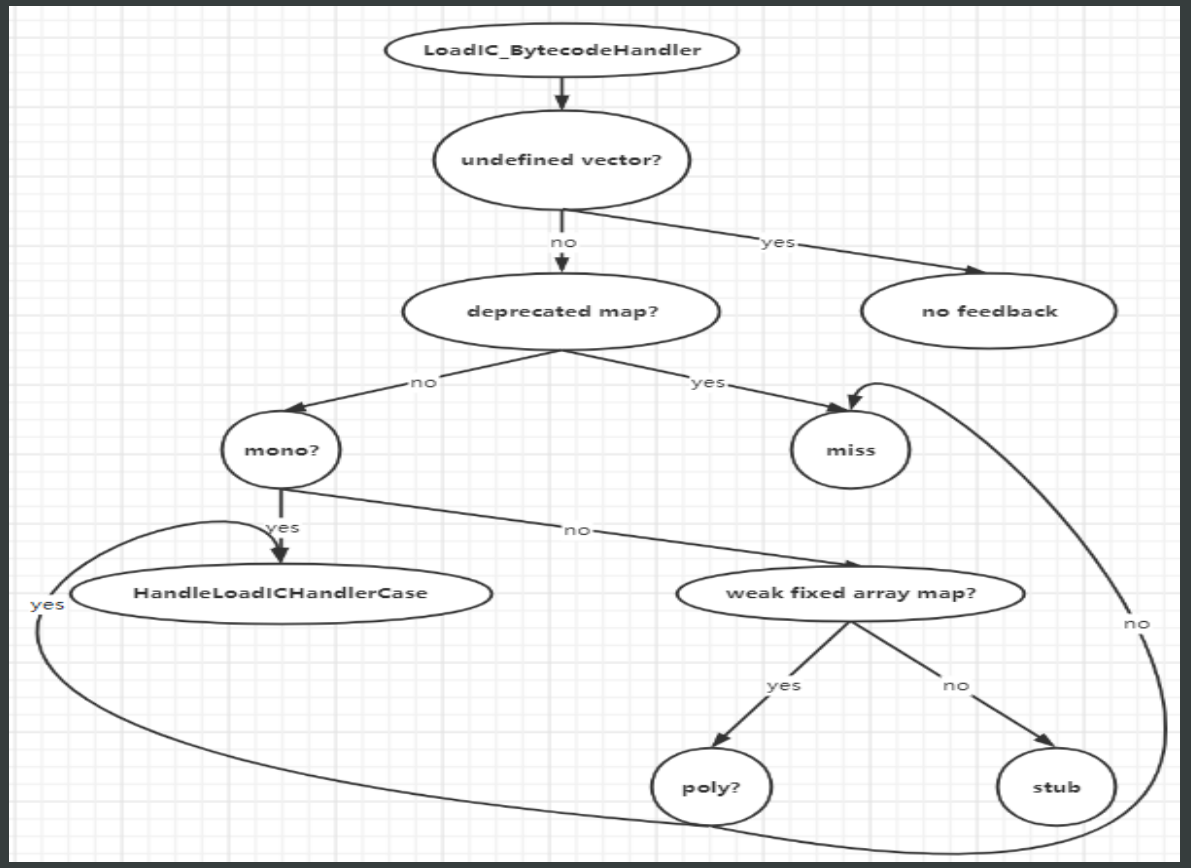
https://slides.com/ripsawridge/deck

Shape



https://www.youtube.com/watch?v=5nmpokoRaZI

LoadIC_BytecodeHandler



Ignition解释器

- 生成字节码
- 解释执行
- 收集反馈信息

Thank You!