//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

#pragma once

namespace Js

{

enum TypeId;

class DetachedStateBase

{

protected:

TypeId typeId;

bool hasBeenClaimed;

public:

DetachedStateBase(TypeId typeId)

: typeId(typeId),

hasBeenClaimed(false)

{

}

TypeId GetTypeId() { return typeId; }

bool HasBeenClaimed() { return hasBeenClaimed; }

void MarkAsClaimed() { hasBeenClaimed = true; }

void CleanUp()

{

if (!hasBeenClaimed)

{

DiscardState();

}

ClearSelfOnly();

}

virtual void ClearSelfOnly() = 0;

virtual void DiscardState() = 0;

virtual void Discard() = 0;

};

typedef enum ArrayBufferAllocationType

{

Heap = 0x0,

CoTask = 0x1,

MemAlloc = 0x02

} ArrayBufferAllocationType;

class ArrayBufferDetachedStateBase : public DetachedStateBase

{

public:

BYTE\* buffer;

uint32 bufferLength;

ArrayBufferAllocationType allocationType;

ArrayBufferDetachedStateBase(TypeId typeId, BYTE\* buffer, uint32 bufferLength, ArrayBufferAllocationType allocationType)

: DetachedStateBase(typeId),

buffer(buffer),

bufferLength(bufferLength),

allocationType(allocationType)

{}

};

}

//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

// This is the list of internal properties used in the Chakra engine.

// They become nameless compile time known PropertyRecords, stored as static

// fields on the InternalPropertyRecords class.

INTERNALPROPERTY(TypeOfPrototypObject) // Used to store the type of the prototype object in the prototype objects slots

INTERNALPROPERTY(NonExtensibleType) // Used to store shared non-extensible type in PathTypeHandler::propertySuccessors map.

INTERNALPROPERTY(SealedType) // Used to store shared sealed type in PathTypeHandler::propertySuccessors map.

INTERNALPROPERTY(FrozenType) // Used to store shared frozen type in PathTypeHandler::propertySuccessors map.

INTERNALPROPERTY(StackTrace) // Stack trace object for Error.stack generation

INTERNALPROPERTY(StackTraceCache) // Cache of Error.stack string

INTERNALPROPERTY(WeakMapKeyMap) // WeakMap data stored on WeakMap key objects

INTERNALPROPERTY(HiddenObject) // Used to store hidden data for JS library code (Intl as an example will use this)

INTERNALPROPERTY(RevocableProxy) // Internal slot for [[RevokableProxy]] for revocable proxy in ES6

INTERNALPROPERTY(MutationBp) // Used to store strong reference to the mutation breakpoint object

#undef INTERNALPROPERTY

//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

/\*

\* These are javascript library functions that might be inlined

\* by the JIT.

\*

\* Notes:

\* - the argc is the number of args to pass to InlineXXX call, e.g. 2 for Math.pow and 2 for String.CharAt.

\* - TODO: consider having dst/src1/src2 in separate columns rather than bitmask, this seems to be better for design but we won't be able to see 'all float' by single check.

\* - TODO: enable string inlines when string type spec is available

\*

\* target name argc flags EntryInfo

\*/

LIBRARY\_FUNCTION(Math, Abs, 1, BIF\_TypeSpecSrcAndDstToFloatOrInt , Math::EntryInfo::Abs)

LIBRARY\_FUNCTION(Math, Acos, 1, BIF\_TypeSpecUnaryToFloat , Math::EntryInfo::Acos)

LIBRARY\_FUNCTION(Math, Asin, 1, BIF\_TypeSpecUnaryToFloat , Math::EntryInfo::Asin)

LIBRARY\_FUNCTION(Math, Atan, 1, BIF\_TypeSpecUnaryToFloat , Math::EntryInfo::Atan)

LIBRARY\_FUNCTION(Math, Atan2, 2, BIF\_TypeSpecAllToFloat , Math::EntryInfo::Atan2)

LIBRARY\_FUNCTION(Math, Ceil, 1, BIF\_TypeSpecDstToInt | BIF\_TypeSpecSrc1ToFloat , Math::EntryInfo::Ceil)

LIBRARY\_FUNCTION(String, CodePointAt, 2, BIF\_TypeSpecSrc2ToInt | BIF\_UseSrc0 , JavascriptString::EntryInfo::CodePointAt)

LIBRARY\_FUNCTION(String, CharAt, 2, BIF\_UseSrc0 , JavascriptString::EntryInfo::CharAt )

LIBRARY\_FUNCTION(String, CharCodeAt, 2, BIF\_UseSrc0 , JavascriptString::EntryInfo::CharCodeAt )

LIBRARY\_FUNCTION(String, Concat, 15, BIF\_UseSrc0 | BIF\_VariableArgsNumber , JavascriptString::EntryInfo::Concat )

LIBRARY\_FUNCTION(String, FromCharCode, 1, BIF\_None , JavascriptString::EntryInfo::FromCharCode)

LIBRARY\_FUNCTION(String, FromCodePoint, 1, BIF\_None , JavascriptString::EntryInfo::FromCodePoint)

LIBRARY\_FUNCTION(String, IndexOf, 3, BIF\_UseSrc0 | BIF\_VariableArgsNumber , JavascriptString::EntryInfo::IndexOf)

LIBRARY\_FUNCTION(String, LastIndexOf, 3, BIF\_UseSrc0 | BIF\_VariableArgsNumber , JavascriptString::EntryInfo::LastIndexOf)

LIBRARY\_FUNCTION(String, Link, 2, BIF\_UseSrc0 , JavascriptString::EntryInfo::Link)

LIBRARY\_FUNCTION(String, LocaleCompare, 2, BIF\_UseSrc0 , JavascriptString::EntryInfo::LocaleCompare)

LIBRARY\_FUNCTION(String, Match, 2, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptString::EntryInfo::Match)

LIBRARY\_FUNCTION(String, Replace, 3, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptString::EntryInfo::Replace)

LIBRARY\_FUNCTION(String, Search, 2, BIF\_UseSrc0 , JavascriptString::EntryInfo::Search)

LIBRARY\_FUNCTION(String, Slice, 3, BIF\_UseSrc0 | BIF\_VariableArgsNumber , JavascriptString::EntryInfo::Slice )

LIBRARY\_FUNCTION(String, Split, 3, BIF\_UseSrc0 | BIF\_VariableArgsNumber | BIF\_IgnoreDst , JavascriptString::EntryInfo::Split)

LIBRARY\_FUNCTION(String, Substr, 3, BIF\_UseSrc0 | BIF\_VariableArgsNumber , JavascriptString::EntryInfo::Substr)

LIBRARY\_FUNCTION(String, Substring, 3, BIF\_UseSrc0 | BIF\_VariableArgsNumber , JavascriptString::EntryInfo::Substring)

LIBRARY\_FUNCTION(String, ToLocaleLowerCase, 1, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptString::EntryInfo::ToLocaleLowerCase)

LIBRARY\_FUNCTION(String, ToLocaleUpperCase, 1, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptString::EntryInfo::ToLocaleUpperCase)

LIBRARY\_FUNCTION(String, ToLowerCase, 1, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptString::EntryInfo::ToLowerCase)

LIBRARY\_FUNCTION(String, ToUpperCase, 1, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptString::EntryInfo::ToUpperCase)

LIBRARY\_FUNCTION(String, Trim, 1, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptString::EntryInfo::Trim)

LIBRARY\_FUNCTION(String, TrimLeft, 1, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptString::EntryInfo::TrimLeft)

LIBRARY\_FUNCTION(String, TrimRight, 1, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptString::EntryInfo::TrimRight)

LIBRARY\_FUNCTION(Math, Cos, 1, BIF\_TypeSpecUnaryToFloat , Math::EntryInfo::Cos)

LIBRARY\_FUNCTION(Math, Exp, 1, BIF\_TypeSpecUnaryToFloat , Math::EntryInfo::Exp)

LIBRARY\_FUNCTION(Math, Floor, 1, BIF\_TypeSpecDstToInt | BIF\_TypeSpecSrc1ToFloat , Math::EntryInfo::Floor)

LIBRARY\_FUNCTION(Math, Log, 1, BIF\_TypeSpecUnaryToFloat , Math::EntryInfo::Log)

LIBRARY\_FUNCTION(Math, Max, 2, BIF\_TypeSpecSrcAndDstToFloatOrInt , Math::EntryInfo::Max)

LIBRARY\_FUNCTION(Math, Min, 2, BIF\_TypeSpecSrcAndDstToFloatOrInt , Math::EntryInfo::Min)

LIBRARY\_FUNCTION(Math, Pow, 2, BIF\_TypeSpecAllToFloat , Math::EntryInfo::Pow)

LIBRARY\_FUNCTION(Math, Imul, 2, BIF\_TypeSpecAllToInt , Math::EntryInfo::Imul)

LIBRARY\_FUNCTION(Math, Clz32, 1, BIF\_TypeSpecAllToInt , Math::EntryInfo::Clz32)

LIBRARY\_FUNCTION(Array, Push, 2, BIF\_UseSrc0 | BIF\_IgnoreDst | BIF\_TypeSpecSrc1ToFloatOrInt, JavascriptArray::EntryInfo::Push)

LIBRARY\_FUNCTION(Array, Pop, 1, BIF\_UseSrc0 | BIF\_TypeSpecDstToFloatOrInt , JavascriptArray::EntryInfo::Pop)

LIBRARY\_FUNCTION(Math, Random, 0, BIF\_TypeSpecDstToFloat , Math::EntryInfo::Random)

LIBRARY\_FUNCTION(Math, Round, 1, BIF\_TypeSpecDstToInt | BIF\_TypeSpecSrc1ToFloat , Math::EntryInfo::Round)

LIBRARY\_FUNCTION(Math, Sin, 1, BIF\_TypeSpecUnaryToFloat , Math::EntryInfo::Sin)

LIBRARY\_FUNCTION(Math, Sqrt, 1, BIF\_TypeSpecUnaryToFloat , Math::EntryInfo::Sqrt)

LIBRARY\_FUNCTION(Math, Tan, 1, BIF\_TypeSpecUnaryToFloat , Math::EntryInfo::Tan)

LIBRARY\_FUNCTION(Array, Concat, 15, BIF\_UseSrc0 | BIF\_VariableArgsNumber , JavascriptArray::EntryInfo::Concat)

LIBRARY\_FUNCTION(Array, IndexOf, 2, BIF\_UseSrc0 , JavascriptArray::EntryInfo::IndexOf)

LIBRARY\_FUNCTION(Array, Includes, 2, BIF\_UseSrc0 , JavascriptArray::EntryInfo::Includes)

LIBRARY\_FUNCTION(Array, IsArray, 1, BIF\_VariableArgsNumber , JavascriptArray::EntryInfo::IsArray)

LIBRARY\_FUNCTION(Array, Join, 2, BIF\_UseSrc0 | BIF\_VariableArgsNumber , JavascriptArray::EntryInfo::Join)

LIBRARY\_FUNCTION(Array, LastIndexOf, 3, BIF\_UseSrc0 | BIF\_VariableArgsNumber , JavascriptArray::EntryInfo::LastIndexOf)

LIBRARY\_FUNCTION(Array, Reverse, 1, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptArray::EntryInfo::Reverse)

LIBRARY\_FUNCTION(Array, Shift, 1, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptArray::EntryInfo::Shift)

LIBRARY\_FUNCTION(Array, Slice, 3, BIF\_UseSrc0 | BIF\_VariableArgsNumber , JavascriptArray::EntryInfo::Slice)

LIBRARY\_FUNCTION(Array, Splice, 15, BIF\_UseSrc0 | BIF\_VariableArgsNumber | BIF\_IgnoreDst , JavascriptArray::EntryInfo::Splice)

LIBRARY\_FUNCTION(Array, Unshift, 15, BIF\_UseSrc0 | BIF\_VariableArgsNumber | BIF\_IgnoreDst , JavascriptArray::EntryInfo::Unshift)

LIBRARY\_FUNCTION(Function, Apply, 3, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptFunction::EntryInfo::Apply)

LIBRARY\_FUNCTION(Function, Call, 15, BIF\_UseSrc0 | BIF\_IgnoreDst | BIF\_VariableArgsNumber , JavascriptFunction::EntryInfo::Call)

LIBRARY\_FUNCTION(GlobalObject, ParseInt, 1, BIF\_IgnoreDst , GlobalObject::EntryInfo::ParseInt)

LIBRARY\_FUNCTION(RegExp, Exec, 2, BIF\_UseSrc0 | BIF\_IgnoreDst , JavascriptRegExp::EntryInfo::Exec)

LIBRARY\_FUNCTION(Math, Fround, 1, BIF\_TypeSpecUnaryToFloat , Math::EntryInfo::Fround)

// Note: 1st column is currently used only for debug tracing.

// SIMD\_JS

#if ENABLE\_NATIVE\_CODEGEN

LIBRARY\_FUNCTION(SIMD\_Float32x4, Float32x4, 4, BIF\_IgnoreDst , SIMDFloat32x4Lib::EntryInfo::Float32x4)

LIBRARY\_FUNCTION(SIMD\_Float32x4, Add, 2, BIF\_IgnoreDst , SIMDFloat32x4Lib::EntryInfo::Add)

LIBRARY\_FUNCTION(SIMD\_Int32x4, Int32x4, 4, BIF\_IgnoreDst , SIMDInt32x4Lib::EntryInfo::Int32x4)

LIBRARY\_FUNCTION(SIMD\_Int32x4, Add, 2, BIF\_IgnoreDst , SIMDInt32x4Lib::EntryInfo::Add)

#endif

//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

#pragma once

#include "Common.h"

//========================

// Parser includes

//========================

#include "ParserCommon.h"

#include "ParseFlags.h"

#include "rterror.h"

// Parser forward decl

class FuncInfo;

class Scope;

class Symbol;

struct Ident;

typedef Ident \*IdentPtr;

enum SymbolType : byte;

// Regex forward decl

namespace UnifiedRegex

{

struct RegexPattern;

template <typename T> class StandardChars; // Used by ThreadContext.h

struct TrigramAlphabet;

struct RegexStacks;

#if ENABLE\_REGEX\_CONFIG\_OPTIONS

class DebugWriter;

struct RegexStats;

class RegexStatsDatabase;

#endif

};

//========================

#include "RuntimeCommon.h"

#include <intsafe.h>

#if !defined(UNREFERENCED\_PARAMETER)

#define UNREFERENCED\_PARAMETER(x) (x)

#endif

class SRCINFO;

class Lowerer;

class LowererMD;

class LowererMDArch;

class ByteCodeGenerator;

interface IActiveScriptDataCache;

class ActiveScriptProfilerHeapEnum;

namespace Js

{

//

// Forward declarations

//

class CharClassifier;

typedef int32 MessageId;

/\* enum \*/ struct PropertyIds;

class DebugDocument;

struct Utf8SourceInfo;

struct CallInfo;

struct InlineeCallInfo;

struct InlineCache;

struct PolymorphicInlineCache;

struct Arguments;

class StringDictionaryWrapper;

struct ByteCodeDumper;

struct ByteCodeReader;

struct ByteCodeWriter;

class JavascriptConversion;

class JavascriptDate;

class JavascriptVariantDate;

class DateImplementation;

class BufferString;

class BufferStringBuilder;

class ConcatString;

class CompoundString;

class JavascriptBoolean;

class JavascriptBooleanObject;

class JavascriptSymbol;

class JavascriptSymbolObject;

class JavascriptProxy;

class JavascriptReflect;

class JavascriptEnumeratorIterator;

class JavascriptArrayIterator;

enum class JavascriptArrayIteratorKind;

class JavascriptMapIterator;

enum class JavascriptMapIteratorKind;

class JavascriptSetIterator;

enum class JavascriptSetIteratorKind;

class JavascriptStringIterator;

class JavascriptPromise;

class JavascriptPromiseCapability;

class JavascriptPromiseReaction;

class JavascriptPromiseAsyncSpawnExecutorFunction;

class JavascriptPromiseAsyncSpawnStepArgumentExecutorFunction;

class JavascriptPromiseCapabilitiesExecutorFunction;

class JavascriptPromiseResolveOrRejectFunction;

class JavascriptPromiseReactionTaskFunction;

class JavascriptPromiseResolveThenableTaskFunction;

class JavascriptPromiseAllResolveElementFunction;

struct JavascriptPromiseAllResolveElementFunctionRemainingElementsWrapper;

struct JavascriptPromiseResolveOrRejectFunctionAlreadyResolvedWrapper;

class JavascriptGenerator;

class LiteralString;

class ArenaLiteralString;

class JavascriptStringObject;

struct PropertyDescriptor;

class Type;

class DynamicType;

class ScriptFunctionType;

class DynamicTypeHandler;

class DeferredTypeHandlerBase;

template <bool IsPrototype> class NullTypeHandler;

template<size\_t size> class SimpleTypeHandler;

class PathTypeHandler;

class IndexPropertyDescriptor;

class DynamicObject;

class ArrayObject;

class WithScopeObject;

class SpreadArgument;

class JavascriptString;

class StringCopyInfo;

class StringCopyInfoStack;

class ObjectPrototypeObject;

class PropertyString;

class ArgumentsObject;

class HeapArgumentsObject;

class ActivationObject;

class JavascriptNumber;

class JavascriptNumberObject;

class ES5ArgumentsObjectEnumerator;

class ScriptContextProfiler;

struct RestrictedErrorStrings;

class JavascriptError;

class NullEnumerator;

//SIMD\_JS

// SIMD

class SIMDFloat32x4Lib;

class JavascriptSIMDFloat32x4;

class SIMDFloat64x2Lib;

class JavascriptSIMDFloat64x2;

class SIMDInt32x4Lib;

class JavascriptSIMDInt32x4;

class SIMDInt8x16Lib;

class JavascriptSIMDInt8x16;

class RecyclableObject;

class JavascriptRegExp;

class JavascriptRegularExpressionResult;

template<typename T> class SparseArraySegment;

enum class DynamicObjectFlags : uint16;

class JavascriptArray;

class JavascriptNativeIntArray;

#if ENABLE\_COPYONACCESS\_ARRAY

class JavascriptCopyOnAccessNativeIntArray;

#endif

class JavascriptNativeFloatArray;

class ES5Array;

class JavascriptFunction;

class ScriptFunction;

class ScriptFunctionWithInlineCache;

class StackScriptFunction;

class GeneratorVirtualScriptFunction;

class JavascriptGeneratorFunction;

class AsmJsScriptFunction;

class JavascriptRegExpConstructor;

class JavascriptRegExpEnumerator;

class BoundFunction;

class JavascriptMap;

class JavascriptSet;

class JavascriptWeakMap;

class JavascriptWeakSet;

class DynamicObject;

class HostObjectBase;

class RootObjectBase;

class ModuleRoot;

class GlobalObject;

class Math;

class JavascriptOperators;

class JavascriptLibrary;

class JavascriptEncodeURI;

class JavascriptEncodeURIComponent;

class JavascriptDecodeURI;

class JavascriptDecodeURIComponent;

class DataView;

struct ConstructorCache;

enum class OpCode : ushort;

enum class OpCodeAsmJs : ushort;

/\* enum \*/ struct OpLayoutType;

/\* enum \*/ struct OpLayoutTypeAsmJs;

class ExceptionBase;

class OutOfMemoryException;

class ScriptDebug;

class ScriptContext;

struct NativeModule;

template <class T> class RcRef;

class TaggedInt;

class TaggedNumber;

struct InterpreterStackFrame;

struct ScriptEntryExitRecord;

class JavascriptStackWalker;

struct AsmJsCallStackLayout;

class JavascriptCallStackLayout;

class Throw;

struct Tick;

struct TickDelta;

class ByteBlock;

class FunctionInfo;

class FunctionBody;

class ParseableFunctionInfo;

struct StatementLocation;

class EntryPointInfo;

struct LoopHeader;

class InternalString;

/\* enum \*/ struct JavascriptHint;

/\* enum \*/ struct BuiltinFunction;

class EnterScriptObject;

class PropertyRecord;

struct IsInstInlineCache;

class EntryPointInfo;

class PolymorphicInlineCacheInfo;

class PropertyGuard;

// asm.js

namespace ArrayBufferView

{

enum ViewType;

}

struct EmitExpressionInfo;

struct AsmJsModuleMemory;

namespace AsmJsLookupSource

{

enum Source;

}

struct AsmJsByteCodeWriter;

class AsmJsArrayView;

class AsmJsType;

class AsmJsRetType;

class AsmJsVarType;

class AsmJsSymbol;

class AsmJsVarBase;

class AsmJsVar;

class AsmJsConstantImport;

class AsmJsArgument;

class AsmJsFunc;

class AsmJsFunctionDeclaration;

class AsmJsFunctionInfo;

class AsmJsModuleInfo;

class AsmJsGlobals;

class AsmJsImportFunction;

class AsmJsTypedArrayFunction;

class AsmJsMathFunction;

class AsmJsMathConst;

#ifdef ASMJS\_PLAT

class AsmJsCodeGenerator;

class AsmJsEncoder;

#endif

struct MathBuiltin;

struct ExclusiveContext;

class AsmJsModuleCompiler;

class AsmJSCompiler;

class AsmJSByteCodeGenerator;

enum AsmJSMathBuiltinFunction;

//////////////////////////////////////////////////////////////////////////

typedef JsUtil::WeakReferenceDictionary<PropertyId, PropertyString, PowerOf2SizePolicy> PropertyStringCacheMap;

extern const FrameDisplay NullFrameDisplay;

extern const FrameDisplay StrictNullFrameDisplay;

enum ImplicitCallFlags : BYTE

{

ImplicitCall\_HasNoInfo = 0x00,

ImplicitCall\_None = 0x01,

ImplicitCall\_ToPrimitive = 0x02 | ImplicitCall\_None,

ImplicitCall\_Accessor = 0x04 | ImplicitCall\_None,

ImplicitCall\_NonProfiledAccessor = 0x08 | ImplicitCall\_None,

ImplicitCall\_External = 0x10 | ImplicitCall\_None,

ImplicitCall\_Exception = 0x20 | ImplicitCall\_None,

ImplicitCall\_NoOpSet = 0x40 | ImplicitCall\_None,

ImplicitCall\_All = 0x7F,

// Implicit call that is not caused by operations for the instruction (e.g. QC and GC dispose)

// where we left script and enter script again. (Also see BEGIN\_LEAVE\_SCRIPT\_INTERNAL)

// This doesn't count as an implicit call on the recorded profile, but if it happens on JIT'ed code

// it will still cause a bailout. Should happen very rarely.

ImplicitCall\_AsyncHostOperation = 0x80

};

}

#include "DataStructures\EvalMapString.h"

bool IsMathLibraryId(Js::PropertyId propertyId);

#include "ByteCode\PropertyIdArray.h"

#include "ByteCode\AuxArray.h"

#include "ByteCode\VarArrayVarCount.h"

// module id

const Js::ModuleID kmodGlobal = 0;

class SourceContextInfo;

#include "activdbg100.h"

#ifndef NTDDI\_WIN10

// These are only defined for the Win10 SDK and above

// Consider: Refactor to avoid needing these?

typedef

enum tagDEBUG\_EVENT\_INFO\_TYPE

{

DEIT\_GENERAL = 0,

DEIT\_ASMJS\_IN\_DEBUGGING = (DEIT\_GENERAL + 1),

DEIT\_ASMJS\_SUCCEEDED = (DEIT\_ASMJS\_IN\_DEBUGGING + 1),

DEIT\_ASMJS\_FAILED = (DEIT\_ASMJS\_SUCCEEDED + 1)

} DEBUG\_EVENT\_INFO\_TYPE;

#define SDO\_ENABLE\_LIBRARY\_STACK\_FRAME ((SCRIPT\_DEBUGGER\_OPTIONS)0x8)

#endif

#include "Base\SourceHolder.h"

#include "Base\Utf8SourceInfo.h"

#include "Base\PropertyRecord.h"

#include "Base\DelayLoadLibrary.h"

#include "Base\CallInfo.h"

#include "Language\ExecutionMode.h"

#include "BackEndAPI.h"

#include "DetachedStateBase.h"

#include "Base\Constants.h"

#include "ByteCode\OpLayoutsCommon.h"

#include "ByteCode\OpLayouts.h"

#include "ByteCode\OpLayoutsAsmJs.h"

#include "ByteCode\OpCodeUtil.h"

#include "Language\Arguments.h"

#include "Types\TypeId.h"

#include "Types\RecyclableObject.h"

#include "Base\ExpirableObject.h"

#include "Types\Type.h"

#include "Types\StaticType.h"

#include "Base\CrossSite.h"

#include "Base\CrossSiteObject.h"

#include "Base\CrossSiteEnumerator.h"

#include "Types\JavascriptEnumerator.h"

#include "Types\DynamicObject.h"

#include "Types\ArrayObject.h"

#include "Types\TypePath.h"

#include "Types\TypeHandler.h"

#include "Types\SimplePropertyDescriptor.h"

#include "Types\DynamicType.h"

#include "Language\StackTraceArguments.h"

#include "Types\PropertyDescriptor.h"

#include "Types\ActivationObject.h"

#include "Base\TempArenaAllocatorObject.h"

#include "Language\ValueType.h"

#include "Language\DynamicProfileInfo.h"

#include "Debug\SourceContextInfo.h"

#include "Language\InlineCache.h"

#include "Language\InlineCachePointerArray.h"

#include "Base\FunctionInfo.h"

#include "Base\FunctionBody.h"

#include "Language\JavascriptExceptionContext.h"

#include "Language\JavascriptExceptionObject.h"

#include "Base\PerfHint.h"

#include "ByteCode\ByteBlock.h"

#include "Library\JavascriptBuiltInFunctions.h"

#include "Library\JavascriptString.h"

#include "Library\StringCopyInfo.h"

#include "Library\JavascriptNumber.h"

#include "Library\JavascriptFunction.h"

#include "Library\RuntimeFunction.h"

#include "Library\JavascriptExternalFunction.h"

#include "Base\CharStringCache.h"

#include "Library\JavascriptObject.h"

#include "Library\BuiltInFlags.h"

#include "Library\ExternalLibraryBase.h"

#include "Library\JavascriptLibraryBase.h"

#include "Library\JavascriptLibrary.h"

#include "Language\JavascriptExceptionOperators.h"

#include "Language\JavascriptOperators.h"

#include "Library\MathLibrary.h"

#include "Base\HiResTimer.h"

#include "Base\WindowsGlobalizationAdapter.h"

#include "Base\WindowsFoundationAdapter.h"

#include "Base\Debug.h"

#ifdef \_M\_X64

#include "Language\amd64\stackframe.h"

#endif

#include "Base\Entropy.h"

#ifdef ENABLE\_BASIC\_TELEMETRY

#include "DirectCall.h"

#include "LanguageTelemetry.h"

#else

#define CHAKRATEL\_LANGSTATS\_INC\_BUILTINCOUNT(builtin)

#define CHAKRATEL\_LANGSTATS\_INC\_LANGFEATURECOUNT(feature, m\_scriptContext)

#endif

#include "Base\ThreadContext.h"

#include "Base\StackProber.h"

#include "Language\EvalMapRecord.h"

#include "Base\RegexPatternMruMap.h"

#include "Language\JavascriptConversion.h"

#include "Base\ScriptContextOptimizationOverrideInfo.h"

#include "Base\scriptContextbase.h"

#include "Base\ScriptContext.h"

#include "Base\LeaveScriptObject.h"

#include "Base\PropertyRecord.h"

#include "ByteCode\ByteCodeReader.h"

#include "Language\TaggedInt.h"

#include "Library\RootObjectBase.h"

#include "Library\GlobalObject.h"

#include "Library\LiteralString.h"

#include "Library\ConcatString.h"

#include "Library\CompoundString.h"

#include "Library\PropertyString.h"

#include "Library\JavascriptTypedNumber.h"

#include "Library\SparseArraySegment.h"

#include "Library\JavascriptError.h"

#include "Library\JavascriptArray.h"

#include "Library\ArrayBuffer.h"

#include "Library\TypedArray.h"

#include "Library\JavascriptBoolean.h"

#include "Types\ScriptFunctionType.h"

#include "Library\ScriptFunction.h"

//

// .inl files

//

#include "commoninl.h"

#include "Language\JavascriptConversion.inl"

#include "Types\RecyclableObject.inl"

#include "Types\DynamicObject.inl"

#include "Library\JavascriptBoolean.inl"

#include "Library\JavascriptArray.inl"

#include "Library\SparseArraySegment.inl"

#include "Library\JavascriptNumber.inl"

#include "Library\JavascriptLibrary.inl"

#include "Language\InlineCache.inl"

#include "Language\InlineCachePointerArray.inl"

#include "Language\JavascriptOperators.inl"

#include "Language\TaggedInt.inl"

#ifndef USED\_IN\_STATIC\_LIB

#ifdef ENABLE\_INTL\_OBJECT

//The "helper" methods below are to resolve external symbol references to our delay-loaded libraries.

inline HRESULT WindowsCreateString(\_In\_reads\_opt\_(length) const WCHAR \* sourceString, UINT32 length, \_Outptr\_result\_maybenull\_ \_Result\_nullonfailure\_ HSTRING \* string)

{

return ThreadContext::GetContextForCurrentThread()->GetWindowsGlobalizationLibrary()->WindowsCreateString(sourceString, length, string);

}

inline HRESULT WindowsCreateStringReference(\_In\_reads\_opt\_(length + 1) const WCHAR \* sourceString, UINT32 length, \_Out\_ HSTRING\_HEADER \* header, \_Outptr\_result\_maybenull\_ \_Result\_nullonfailure\_ HSTRING \* string)

{

return ThreadContext::GetContextForCurrentThread()->GetWindowsGlobalizationLibrary()->WindowsCreateStringReference(sourceString, length, header, string);

}

inline HRESULT WindowsDeleteString(\_In\_opt\_ HSTRING string)

{

return ThreadContext::GetContextForCurrentThread()->GetWindowsGlobalizationLibrary()->WindowsDeleteString(string);

}

inline PCWSTR WindowsGetStringRawBuffer(\_In\_opt\_ HSTRING string, \_Out\_opt\_ UINT32 \* length)

{

return ThreadContext::GetContextForCurrentThread()->GetWindowsGlobalizationLibrary()->WindowsGetStringRawBuffer(string, length);

}

inline HRESULT WindowsCompareStringOrdinal(\_In\_opt\_ HSTRING string1, \_In\_opt\_ HSTRING string2, \_Out\_ INT32 \* result)

{

return ThreadContext::GetContextForCurrentThread()->GetWindowsGlobalizationLibrary()->WindowsCompareStringOrdinal(string1, string2, result);

}

inline HRESULT WindowsDuplicateString(\_In\_opt\_ HSTRING original, \_Outptr\_result\_maybenull\_ \_Result\_nullonfailure\_ HSTRING \*newString)

{

return ThreadContext::GetContextForCurrentThread()->GetWindowsGlobalizationLibrary()->WindowsDuplicateString(original, newString);

}

#endif

#endif

//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

#pragma once

// Runtime.h has both definitions and #include other runtime files.

// Definitions here are extracted definitions (not #include's) from Runtime.h that core Runtime .h's can be used without #including full Runtime.h

extern int TotalNumberOfBuiltInProperties;

namespace Js

{

// Forwards

class RecyclableObject;

struct CallInfo;

class PropertyRecord;

class JavascriptString;

struct FrameDisplay;

class TypedArrayBase;

#if \_M\_IX86

#define unaligned

#elif \_M\_X64 || \_M\_ARM || \_M\_ARM64

#define unaligned \_\_unaligned

#else

#error Must define alignment capabilities for processor

#endif

typedef uint32 RegSlot;

typedef uint16 ArgSlot;

typedef uint16 PropertyIndex;

typedef int32 BigPropertyIndex;

typedef unsigned char PropertyAttributes;

typedef uint32 SourceId;

typedef uint16 ProfileId;

typedef uint32 InlineCacheIndex;

// Inline cache flags when property of the object is not writable

enum InlineCacheFlags : char {

InlineCacheNoFlags = 0x0,

InlineCacheGetterFlag = 0x1,

InlineCacheSetterFlag = 0x2,

};

#define PropertyNone 0x00

#define PropertyEnumerable 0x01

#define PropertyConfigurable 0x02

#define PropertyWritable 0x04

#define PropertyDeleted 0x08

#define PropertyLetConstGlobal 0x10

#define PropertyDeclaredGlobal 0x20

#define PropertyLet 0x40

#define PropertyConst 0x80

// No more flags will fit unless PropertyAttributes is bumped up to a short instead of char

#define PropertyBuiltInMethodDefaults (PropertyConfigurable|PropertyWritable)

#define PropertyDynamicTypeDefaults (PropertyConfigurable|PropertyWritable|PropertyEnumerable)

#define PropertyLetDefaults (PropertyEnumerable|PropertyConfigurable|PropertyWritable|PropertyLet)

#define PropertyConstDefaults (PropertyEnumerable|PropertyConfigurable|PropertyConst)

#define PropertyDeletedDefaults (PropertyDeleted | PropertyWritable | PropertyConfigurable)

#define PropertyNoRedecl (PropertyLet | PropertyConst)

#define PropertyClassMemberDefaults (PropertyConfigurable|PropertyWritable)

BEGIN\_ENUM\_UINT(InternalPropertyIds)

#define INTERNALPROPERTY(n) n,

#include "InternalPropertyList.h"

Count,

END\_ENUM\_UINT()

inline BOOL IsInternalPropertyId(PropertyId propertyId)

{

return propertyId < InternalPropertyIds::Count;

}

BEGIN\_ENUM\_UINT(PropertyIds)

\_none = InternalPropertyIds::Count,

#define ENTRY\_INTERNAL\_SYMBOL(n) n,

#define ENTRY\_SYMBOL(n, d) n,

#define ENTRY(n) n,

#define ENTRY2(n, s) n,

#include "Base\JnDirectFields.h"

\_countJSOnlyProperty,

END\_ENUM\_UINT()

inline BOOL IsBuiltInPropertyId(PropertyId propertyId)

{

return propertyId < TotalNumberOfBuiltInProperties;

}

#define PropertyTypesNone 0x00

#define PropertyTypesReserved 0x01 // This bit is always to prevent the DWORD in DynamicTypeHandler looking like a pointer.

#define PropertyTypesWritableDataOnly 0x10 // Indicates that a type handler has only writable data properties

// (no accessors or non-writable properties)

#define PropertyTypesWritableDataOnlyDetection 0x20 // Set on each call to DynamicTypeHandler::SetHasOnlyWritableDataProperties.

#define PropertyTypesInlineSlotCapacityLocked 0x40 // Indicates that the inline slot capacity has been shrunk already and shouldn't be touched again.

#define PropertyTypesAll 0x70

typedef unsigned char PropertyTypes; // Holds flags that represent general information about the types of properties

// handled by a type handler.

BEGIN\_ENUM\_UINT(JavascriptHint)

None, // no hint. use the default for that object

HintString = 0x00000001, // 'string' hint in ToPrimitiveValue()

HintNumber = 0x00000002, // 'number' hint

END\_ENUM\_UINT()

enum DescriptorFlags

{

None = 0x0, // No data/accessor descriptor

Accessor = 0x1, // An accessor descriptor is present

Data = 0x2, // A data descriptor is present

Writable = 0x4, // Data descriptor is writable

Const = 0x8, // Data is const, meaning we throw on attempt to write to it

Proxy = 0x10, // data returned from proxy.

WritableData = Data | Writable // Data descriptor is writable

};

BEGIN\_ENUM\_BYTE(BuiltinFunction)

#define LIBRARY\_FUNCTION(obj, name, argc, flags, entry) obj##\_##name,

#include "LibraryFunction.h"

#undef LIBRARY\_FUNCTION

Count,

None,

END\_ENUM\_BYTE()

typedef void \* Var;

typedef WriteBarrierPtr<void> WriteBarrierVar;

typedef Var(\_\_cdecl \*JavascriptMethod)(RecyclableObject\*, CallInfo, ...);

typedef Var(\*ExternalMethod)(RecyclableObject\*, CallInfo, Var\*);

const uintptr AtomTag\_Object = 0x0;

#if INT32VAR

// The 49th bit is set in this representation

const int32 VarTag\_Shift = 48;

const uintptr AtomTag\_IntPtr = (((uintptr)0x1i64) << VarTag\_Shift);

const int32 AtomTag\_Int32 = 0x0; // lower 32-bits of a tagged integer

const uintptr AtomTag = 0x1;

const int32 AtomTag\_Multiply = 1;

const int32 AtomTag\_Pair = 0x00010001; // Pair of tags

#else

const uintptr AtomTag\_IntPtr = 0x1;

const int32 AtomTag\_Int32 = 0x1; // lower 32-bits of a tagged integer

const uintptr AtomTag = 0x1;

const int32 VarTag\_Shift = 1;

const int32 AtomTag\_Multiply = 1 << VarTag\_Shift;

#endif

#if FLOATVAR

const uint64 FloatTag\_Value = 0xFFFCull << 48;

#endif

template <bool IsPrototypeTemplate> class NullTypeHandler;

template <typename TPropertyIndex, typename TMapKey, bool IsNotExtensibleSupported> class SimpleDictionaryTypeHandlerBase;

template <typename TPropertyIndex, typename TMapKey, bool IsNotExtensibleSupported> class SimpleDictionaryUnorderedTypeHandler;

template <typename TPropertyIndex> class DictionaryTypeHandlerBase;

template <typename TPropertyIndex> class ES5ArrayTypeHandlerBase;

typedef NullTypeHandler<false> NonProtoNullTypeHandler;

typedef NullTypeHandler<true> ProtoNullTypeHandler;

typedef SimpleDictionaryTypeHandlerBase<PropertyIndex, const PropertyRecord\*, false> SimpleDictionaryTypeHandler;

typedef SimpleDictionaryTypeHandlerBase<PropertyIndex, const PropertyRecord\*, true> SimpleDictionaryTypeHandlerNotExtensible;

typedef SimpleDictionaryTypeHandlerBase<BigPropertyIndex, const PropertyRecord\*, false> BigSimpleDictionaryTypeHandler;

typedef SimpleDictionaryTypeHandlerBase<BigPropertyIndex, const PropertyRecord\*, true> BigSimpleDictionaryTypeHandlerNotExtensible;

typedef SimpleDictionaryUnorderedTypeHandler<PropertyIndex, const PropertyRecord\*, false> SimpleDictionaryUnorderedPropertyRecordKeyedTypeHandler;

typedef SimpleDictionaryUnorderedTypeHandler<PropertyIndex, const PropertyRecord\*, true> SimpleDictionaryUnorderedPropertyRecordKeyedTypeHandlerNotExtensible;

typedef SimpleDictionaryUnorderedTypeHandler<BigPropertyIndex, const PropertyRecord\*, false> BigSimpleDictionaryUnorderedPropertyRecordKeyedTypeHandler;

typedef SimpleDictionaryUnorderedTypeHandler<BigPropertyIndex, const PropertyRecord\*, true> BigSimpleDictionaryUnorderedPropertyRecordKeyedTypeHandlerNotExtensible;

typedef SimpleDictionaryUnorderedTypeHandler<PropertyIndex, JavascriptString\*, false> SimpleDictionaryUnorderedStringKeyedTypeHandler;

typedef SimpleDictionaryUnorderedTypeHandler<PropertyIndex, JavascriptString\*, true> SimpleDictionaryUnorderedStringKeyedTypeHandlerNotExtensible;

typedef SimpleDictionaryUnorderedTypeHandler<BigPropertyIndex, JavascriptString\*, false> BigSimpleDictionaryUnorderedStringKeyedTypeHandler;

typedef SimpleDictionaryUnorderedTypeHandler<BigPropertyIndex, JavascriptString\*, true> BigSimpleDictionaryUnorderedStringKeyedTypeHandlerNotExtensible;

typedef DictionaryTypeHandlerBase<PropertyIndex> DictionaryTypeHandler;

typedef DictionaryTypeHandlerBase<BigPropertyIndex> BigDictionaryTypeHandler;

typedef ES5ArrayTypeHandlerBase<PropertyIndex> ES5ArrayTypeHandler;

typedef ES5ArrayTypeHandlerBase<BigPropertyIndex> BigES5ArrayTypeHandler;

template <int N> class ConcatStringN;

typedef ConcatStringN<2> ConcatStringN2;

typedef ConcatStringN<4> ConcatStringN4;

typedef ConcatStringN<6> ConcatStringN6;

typedef ConcatStringN<7> ConcatStringN7;

template <wchar\_t L, wchar\_t R> class ConcatStringWrapping;

typedef ConcatStringWrapping<L'[', L']'> ConcatStringWrappingSB;

typedef ConcatStringWrapping<L'{', L'}'> ConcatStringWrappingB;

typedef ConcatStringWrapping<L'"', L'"'> ConcatStringWrappingQ;

} // namespace Js.

namespace JSON

{

class JSONParser;

}

//

// Below was moved from ByteCodeGenerator.h to share with jscript9diag.

//

#define REGSLOT\_TO\_VARREG(r) (r)

// To map between real reg number and const reg number, add 2 and negate.

// This way, 0xFFFF (no register) maps to itself, and 0xFFFF is never a valid number.

#define REGSLOT\_TO\_CONSTREG(r) ((Js::RegSlot)(0 - (r + 2)))

#define CONSTREG\_TO\_REGSLOT(r) ((Js::RegSlot)(0 - (r + 2)))

//

// Shared string literals

//

#define JS\_DISPLAY\_STRING\_NAN L"NaN"

#define JS\_DISPLAY\_STRING\_DATE L"Date"

#define JS\_DISPLAY\_STRING\_INVALID\_DATE L"Invalid Date"

#define JS\_DISPLAY\_STRING\_FUNCTION\_ANONYMOUS L"\012function() {\012 [native code]\012}\012"

#define JS\_DISPLAY\_STRING\_FUNCTION\_HEADER L"function "

#define JS\_DISPLAY\_STRING\_FUNCTION\_BODY L"() { [native code] }"

#define JS\_DIAG\_TYPE\_JavascriptRegExp L"Object, (Regular Expression)"

#define JS\_DIAG\_VALUE\_JavascriptRegExpConstructor L"{...}"

#define JS\_DIAG\_TYPE\_JavascriptRegExpConstructor L"Object, (RegExp constructor)"

#define JS\_DEFAULT\_CTOR\_DISPLAY\_STRING L"constructor() {}"

#define JS\_DEFAULT\_EXTENDS\_CTOR\_DISPLAY\_STRING L"constructor(...args) { super(...args); }"

#include "Language\SIMDUtils.h"

//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

#ifdef DEFINE\_ALL\_FIELDS

#define DEFINE\_FUNCTION\_PROXY\_FIELDS 1

#define DEFINE\_PARSEABLE\_FUNCTION\_INFO\_FIELDS 1

#define DEFINE\_FUNCTION\_BODY\_FIELDS 1

#endif

// Default declaration for FunctionBody.h

#ifndef DECLARE\_SERIALIZABLE\_FIELD

#define DECLARE\_SERIALIZABLE\_FIELD(type, name, serializableType) type name

#endif

#ifndef DECLARE\_MANUAL\_SERIALIZABLE\_FIELD

#define DECLARE\_MANUAL\_SERIALIZABLE\_FIELD(type, name, serializableType, serializeHere) type name

#endif

#ifdef CURRENT\_ACCESS\_MODIFIER

#define PROTECTED\_FIELDS protected:

#define PRIVATE\_FIELDS private:

#define PUBLIC\_FIELDS public:

#else

#define CURRENT\_ACCESS\_MODIFIER

#define PROTECTED\_FIELDS

#define PRIVATE\_FIELDS

#define PUBLIC\_FIELDS

#endif

#if DEFINE\_FUNCTION\_PROXY\_FIELDS

PROTECTED\_FIELDS

DECLARE\_MANUAL\_SERIALIZABLE\_FIELD(uint, m\_nestedCount, UInt32, false);

CURRENT\_ACCESS\_MODIFIER

#endif

#if DEFINE\_PARSEABLE\_FUNCTION\_INFO\_FIELDS

PROTECTED\_FIELDS

DECLARE\_SERIALIZABLE\_FIELD(ulong, m\_grfscr, ULong); // For values, see fscr\* values in scrutil.h.

DECLARE\_SERIALIZABLE\_FIELD(ArgSlot, m\_inParamCount, ArgSlot); // Count of 'in' parameters to method

DECLARE\_SERIALIZABLE\_FIELD(ArgSlot, m\_reportedInParamCount, ArgSlot); // Count of 'in' parameters to method excluding default and rest

DECLARE\_SERIALIZABLE\_FIELD(charcount\_t, m\_cchStartOffset, CharCount); // offset in characters from the start of the document.

DECLARE\_SERIALIZABLE\_FIELD(charcount\_t, m\_cchLength, CharCount); // length of the function in code points (not bytes)

DECLARE\_SERIALIZABLE\_FIELD(uint, m\_cbLength, UInt32); // length of the function in bytes

PUBLIC\_FIELDS

DECLARE\_SERIALIZABLE\_FIELD(UINT, scopeSlotArraySize, UInt32);

CURRENT\_ACCESS\_MODIFIER

#endif

#if DEFINE\_FUNCTION\_BODY\_FIELDS

PUBLIC\_FIELDS

DECLARE\_SERIALIZABLE\_FIELD(RegSlot, m\_varCount, RegSlot); // Count of non-constant locals

DECLARE\_SERIALIZABLE\_FIELD(RegSlot, m\_constCount, RegSlot); // Count of enregistered constants

DECLARE\_SERIALIZABLE\_FIELD(RegSlot, m\_firstTmpReg, RegSlot);

DECLARE\_SERIALIZABLE\_FIELD(RegSlot, m\_outParamMaxDepth, RegSlot); // Count of call depth in a nested expression

DECLARE\_SERIALIZABLE\_FIELD(uint, m\_byteCodeCount, RegSlot);

DECLARE\_SERIALIZABLE\_FIELD(uint, m\_byteCodeWithoutLDACount, RegSlot);

DECLARE\_SERIALIZABLE\_FIELD(uint, m\_byteCodeInLoopCount, UInt32);

DECLARE\_SERIALIZABLE\_FIELD(uint16, m\_envDepth, UInt16);

DECLARE\_SERIALIZABLE\_FIELD(uint16, m\_argUsedForBranch, UInt16);

PRIVATE\_FIELDS

DECLARE\_SERIALIZABLE\_FIELD(ProfileId, profiledLdElemCount, UInt16);

DECLARE\_SERIALIZABLE\_FIELD(ProfileId, profiledStElemCount, UInt16);

DECLARE\_SERIALIZABLE\_FIELD(ProfileId, profiledCallSiteCount, UInt16);

DECLARE\_SERIALIZABLE\_FIELD(ProfileId, profiledArrayCallSiteCount, UInt16);

DECLARE\_SERIALIZABLE\_FIELD(ProfileId, profiledDivOrRemCount, UInt16);

DECLARE\_SERIALIZABLE\_FIELD(ProfileId, profiledSwitchCount, UInt16);

DECLARE\_SERIALIZABLE\_FIELD(ProfileId, profiledReturnTypeCount, UInt16);

DECLARE\_SERIALIZABLE\_FIELD(ProfileId, profiledSlotCount, UInt16);

DECLARE\_SERIALIZABLE\_FIELD(uint, loopCount, RegSlot);

DECLARE\_SERIALIZABLE\_FIELD(FunctionBodyFlags, flags, FunctionBodyFlags);

DECLARE\_SERIALIZABLE\_FIELD(bool, m\_hasFinally, Bool);

DECLARE\_SERIALIZABLE\_FIELD(bool, hasScopeObject, Bool);

DECLARE\_SERIALIZABLE\_FIELD(bool, hasCachedScopePropIds, Bool);

DECLARE\_SERIALIZABLE\_FIELD(uint, inlineCacheCount, UInt32);

DECLARE\_SERIALIZABLE\_FIELD(uint, rootObjectLoadInlineCacheStart, UInt32);

DECLARE\_SERIALIZABLE\_FIELD(uint, rootObjectLoadMethodInlineCacheStart, UInt32);

DECLARE\_SERIALIZABLE\_FIELD(uint, rootObjectStoreInlineCacheStart, UInt32);

DECLARE\_SERIALIZABLE\_FIELD(uint, isInstInlineCacheCount, UInt32);

DECLARE\_SERIALIZABLE\_FIELD(uint, referencedPropertyIdCount, UInt32);

DECLARE\_SERIALIZABLE\_FIELD(uint, objLiteralCount, UInt32);

DECLARE\_SERIALIZABLE\_FIELD(uint, literalRegexCount, UInt32);

DECLARE\_SERIALIZABLE\_FIELD(uint, innerScopeCount, UInt32);

DECLARE\_SERIALIZABLE\_FIELD(RegSlot, localClosureRegister, RegSlot);

DECLARE\_SERIALIZABLE\_FIELD(RegSlot, localFrameDisplayRegister, RegSlot);

DECLARE\_SERIALIZABLE\_FIELD(RegSlot, envRegister, RegSlot);

DECLARE\_SERIALIZABLE\_FIELD(RegSlot, thisRegisterForEventHandler, RegSlot);

DECLARE\_SERIALIZABLE\_FIELD(RegSlot, firstInnerScopeRegister, RegSlot);

DECLARE\_SERIALIZABLE\_FIELD(RegSlot, funcExprScopeRegister, RegSlot);

CURRENT\_ACCESS\_MODIFIER

#endif

#undef DEFINE\_ALL\_FIELDS

#undef DEFINE\_FUNCTION\_PROXY\_FIELDS

#undef DEFINE\_PARSEABLE\_FUNCTION\_INFO\_FIELDS

#undef DEFINE\_FUNCTION\_BODY\_FIELDS

#undef CURRENT\_ACCESS\_MODIFIER

#undef DECLARE\_MANUAL\_SERIALIZABLE\_FIELD

#undef DECLARE\_SERIALIZABLE\_FIELD

#undef PROTECTED\_FIELDS

#undef PRIVATE\_FIELDS

#undef PUBLIC\_FIELDS