DXR: A Semantic Source Code Browser

Joshua Cranmer

Problems of Large Codebases

- Multiple languages
- Generated code
- Firefox is really large:
  - 8.5M lines of text in 51K files (320 MiB)
  - 350K lines (15 MiB) of generated C/C++ code
  - 30K types, 40K macros, 100K functions

Existing Work

- Source code browsers exist
  - LXR, ctags, doxygen
- Macro support problematic
  - Hidden definitions
  - Confusing parses
- Unqualified simple names as UIDs
  - Firefox has 774 functions called `Init`
- Compilers already solve these problems

Architecture

- Three major parts:
  - Compiler plugins
  - Indexer
  - Web application
- Support for multiple languages
- Modular and extensible

Architecture – Compiler Plugin

- Uses a Clang compiler plugin
- Hook into preprocessor, diagnostics, AST
- For every useful feature:
  - Is this interesting?
  - If so, record information to per-file buffer
- Output filename uses hash of contents

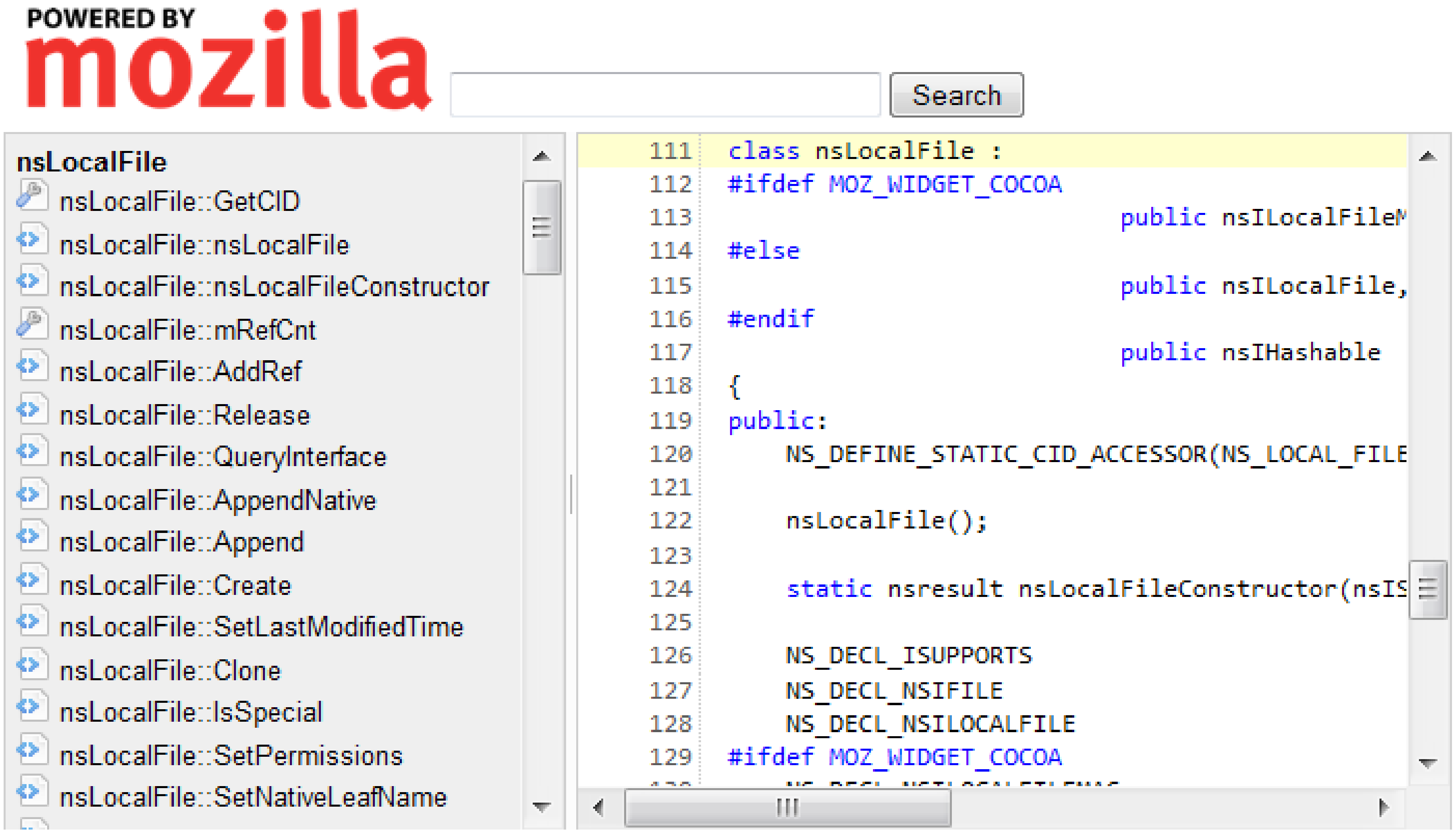
Architecture – Indexer

- Combines results from plugins
- Maps declarations to definitions
- Performs limited whole-program analysis
    - Creates complete type hierarchy
    - Creates full callgraph
- Resolves cross-language entities
- Outputs annotated source, database

Architecture – Web App

- SQLite database backend
- Statically-generated HTML files
- Information retrieval via JSON

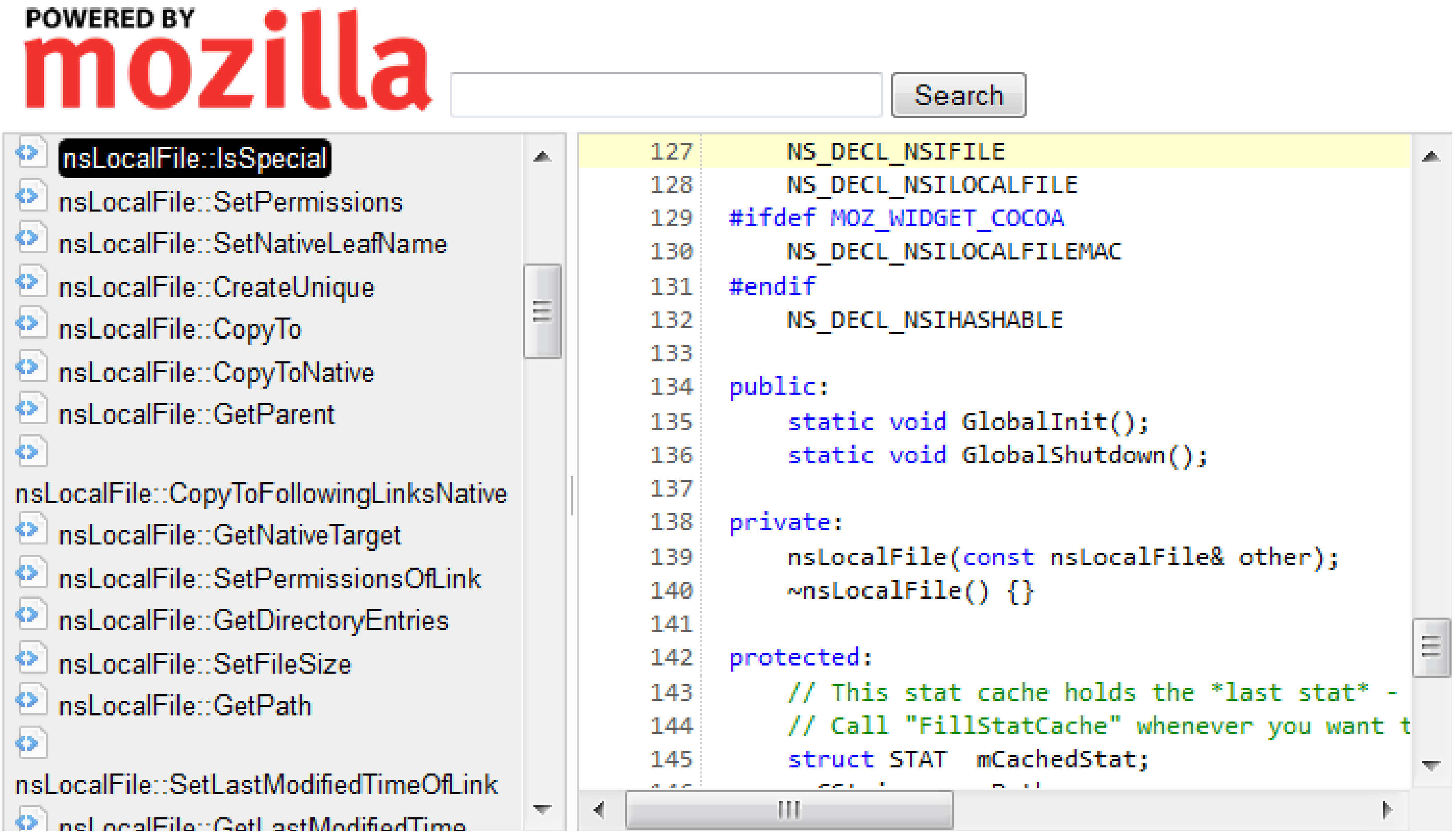
Demo: Sidebar



nsLocalFile
- nsLocalFile::GetCID
- nsLocalFile::nsLocalFile
- nsLocalFile::nsLocalFileConstructor
- nsLocalFile::mRefCnt
- nsLocalFile::AddRef
- nsLocalFile::Release
- nsLocalFile::QueryInterface
- nsLocalFile::AppendNative
- nsLocalFile::Append
- nsLocalFile::Create
- nsLocalFile::SetLastModifiedTime
- nsLocalFile::Clone
- nsLocalFile::IsSpecial
- nsLocalFile::SetPermissions
- nsLocalFile::SetNativeLeafName

```
class nsLocalFile :
#ifdef MOZ_WIDGET_COCOA
                           public nsILocalFileM
#else
                           public nsILocalFile,
#endif
                           public nsIHashable
{
public:
    NS_DEFINE_STATIC_CID_ACCESSOR(NS_LOCAL_FILE

    nsLocalFile();

    static nsresult nsLocalFileConstructor(nsIS

    NS_DECL_ISUPPORTS
    NS_DECL_NSIFILE
    NS_DECL_NSILOCALFILE
#ifdef MOZ_WIDGET_COCOA
```

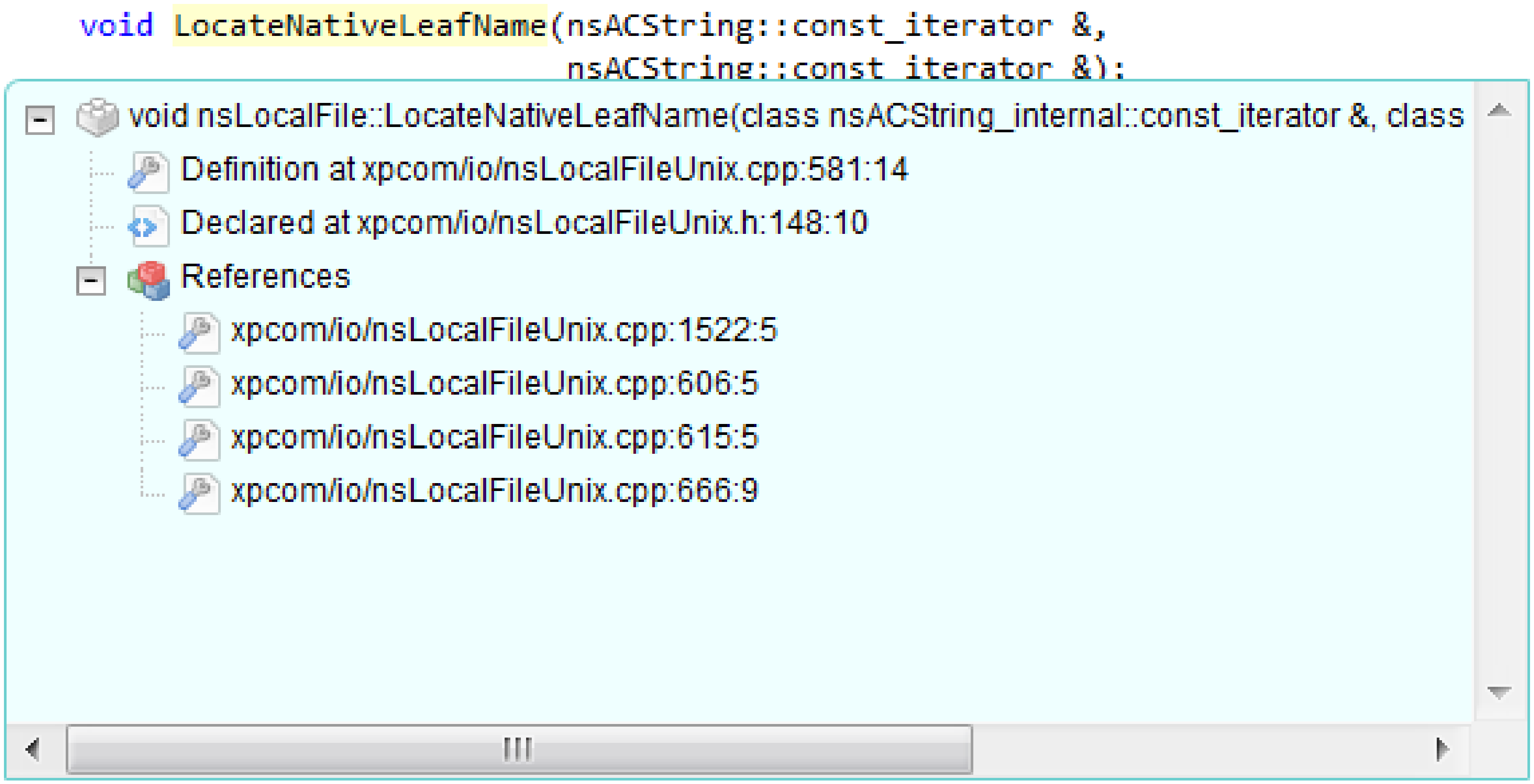
Demo: Sidebar



nsLocalFile::IsSpecial
nsLocalFile::SetPermissions
nsLocalFile::SetNativeLeafName
nsLocalFile::CreateUnique
nsLocalFile::CopyTo
nsLocalFile::CopyToNative
nsLocalFile::GetParent
nsLocalFile::CopyToFollowingLinksNative
nsLocalFile::GetNativeTarget
nsLocalFile::SetPermissionsOfLink
nsLocalFile::GetDirectoryEntries
nsLocalFile::SetFileSize
nsLocalFile::GetPath
nsLocalFile::SetLastModifiedTimeOfLink

```
    NS_DECL_NSIFILE
    NS_DECL_NSILOCALFILE
#ifdef MOZ_WIDGET_COCOA
    NS_DECL_NSILOCALFILEMAC
#endif
    NS_DECL_NSIHASHABLE

public:
    static void GlobalInit();
    static void GlobalShutdown();

private:
    nsLocalFile(const nsLocalFile& other);
    ~nsLocalFile() {}

protected:
    // This stat cache holds the *last stat* -
    // Call "FillStatCache" whenever you want t
    struct STAT  mCachedStat;
```

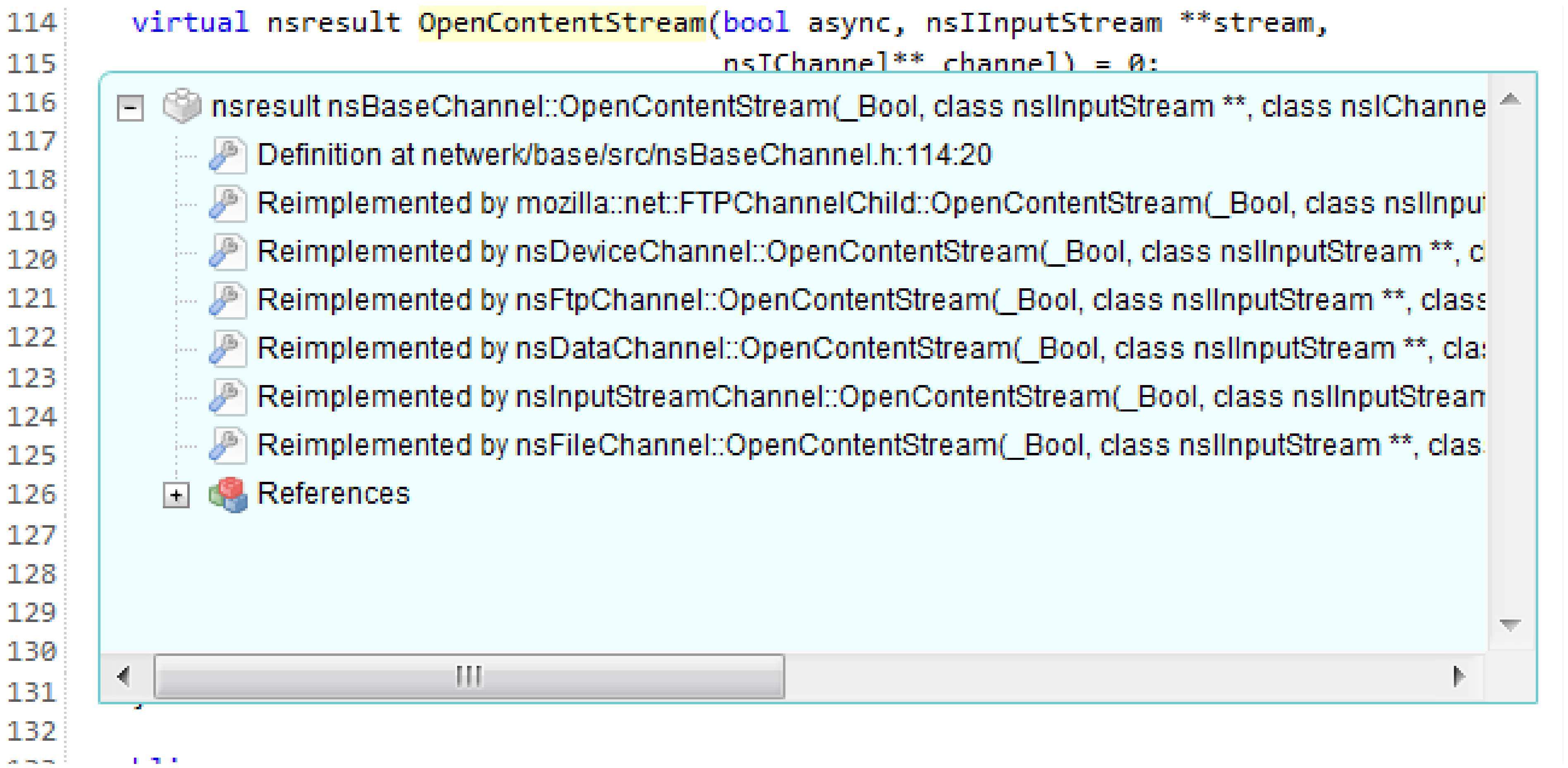
Demo: Infobox



```
void LocateNativeLeafName(nsACString::const_iterator &,
                          nsACString::const_iterator &);
```

- void nsLocalFile::LocateNativeLeafName(class nsACString_internal::const_iterator &, class
  - Definition at xpcom/io/nsLocalFileUnix.cpp:581:14
  - Declared at xpcom/io/nsLocalFileUnix.h:148:10
  - References
    - xpcom/io/nsLocalFileUnix.cpp:1522:5
    - xpcom/io/nsLocalFileUnix.cpp:606:5
    - xpcom/io/nsLocalFileUnix.cpp:615:5
    - xpcom/io/nsLocalFileUnix.cpp:666:9

Demo: Infobox

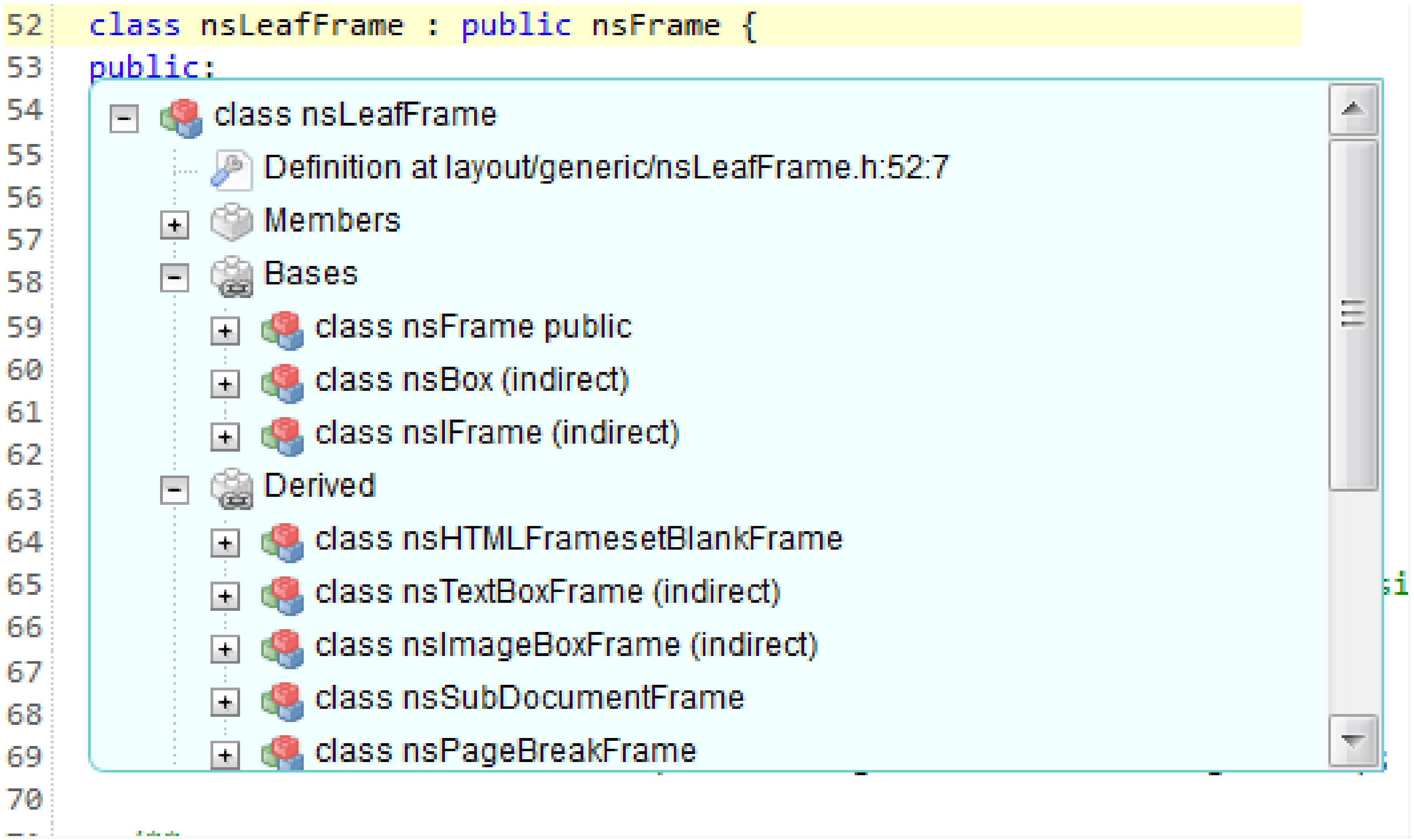


```
virtual nsresult OpenContentStream(bool async, nsIInputStream **stream,
                                   nsIChannel** channel) = 0;
```

nsresult nsBaseChannel::OpenContentStream(_Bool, class nsIInputStream **, class nsIChanne
- Definition at netwerk/base/src/nsBaseChannel.h:114:20
- Reimplemented by mozilla::net::FTPChannelChild::OpenContentStream(_Bool, class nsIInput
- Reimplemented by nsDeviceChannel::OpenContentStream(_Bool, class nsIInputStream **, c
- Reimplemented by nsFtpChannel::OpenContentStream(_Bool, class nsIInputStream **, class
- Reimplemented by nsDataChannel::OpenContentStream(_Bool, class nsIInputStream **, cla
- Reimplemented by nsInputStreamChannel::OpenContentStream(_Bool, class nsIInputStrean
- Reimplemented by nsFileChannel::OpenContentStream(_Bool, class nsIInputStream **, clas

References

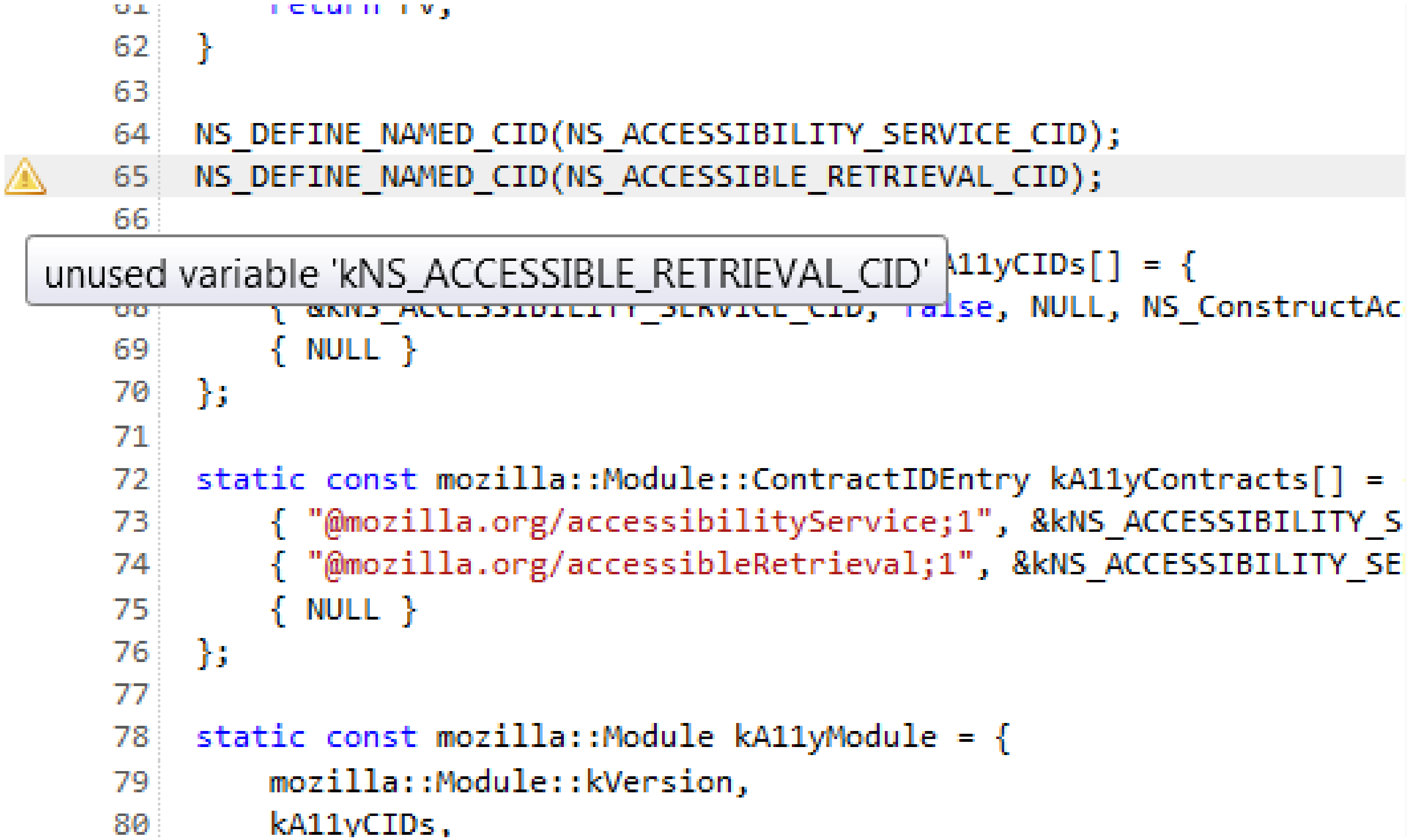
Demo: Infobox



```
class nsLeafFrame : public nsFrame {
public:
```

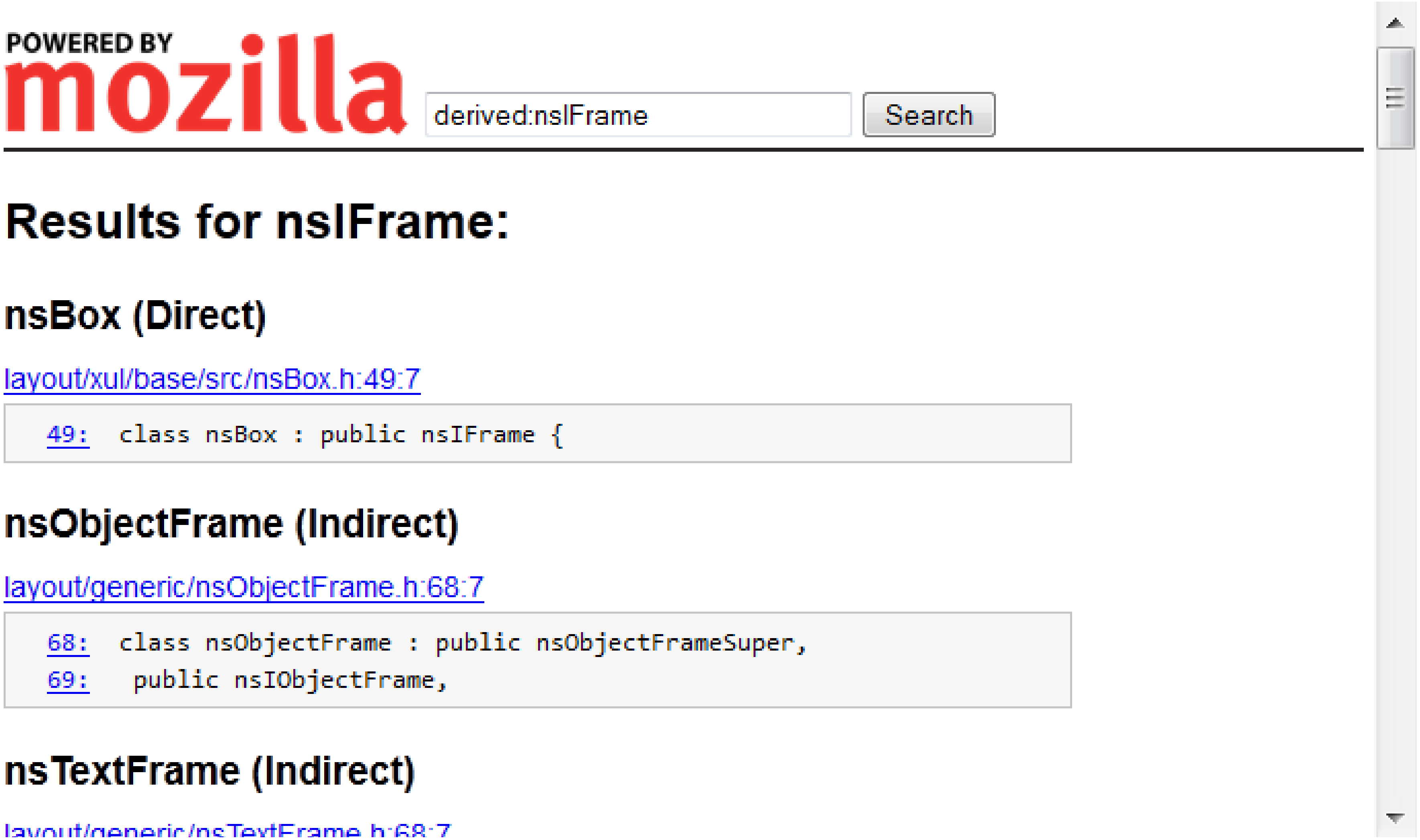
class nsLeafFrame
- Definition at layout/generic/nsLeafFrame.h:52:7
- Members
- Bases
  - class nsFrame public
  - class nsBox (indirect)
  - class nsIFrame (indirect)
- Derived
  - class nsHTMLFramesetBlankFrame
  - class nsTextBoxFrame (indirect)
  - class nsImageBoxFrame (indirect)
  - class nsSubDocumentFrame
  - class nsPageBreakFrame

Demo: Warnings



```
    return rv;
}

NS_DEFINE_NAMED_CID(NS_ACCESSIBILITY_SERVICE_CID);
NS_DEFINE_NAMED_CID(NS_ACCESSIBLE_RETRIEVAL_CID);

unused variable 'kNS_ACCESSIBLE_RETRIEVAL_CID'  A11yCIDs[] = {
    { &kNS_ACCESSIBILITY_SERVICE_CID, false, NULL, NS_ConstructAc
    { NULL }
};

static const mozilla::Module::ContractIDEntry kA11yContracts[] =
    { "@mozilla.org/accessibilityService;1", &kNS_ACCESSIBILITY_S
    { "@mozilla.org/accessibleRetrieval;1", &kNS_ACCESSIBILITY_SE
    { NULL }
};

static const mozilla::Module kA11yModule = {
    mozilla::Module::kVersion,
    kA11yCIDs,
```

Demo: Searching



Results for nsIFrame:

nsBox (Direct)

layout/xul/base/src/nsBox.h:49:7

```
49:  class nsBox : public nsIFrame {
```

nsObjectFrame (Indirect)

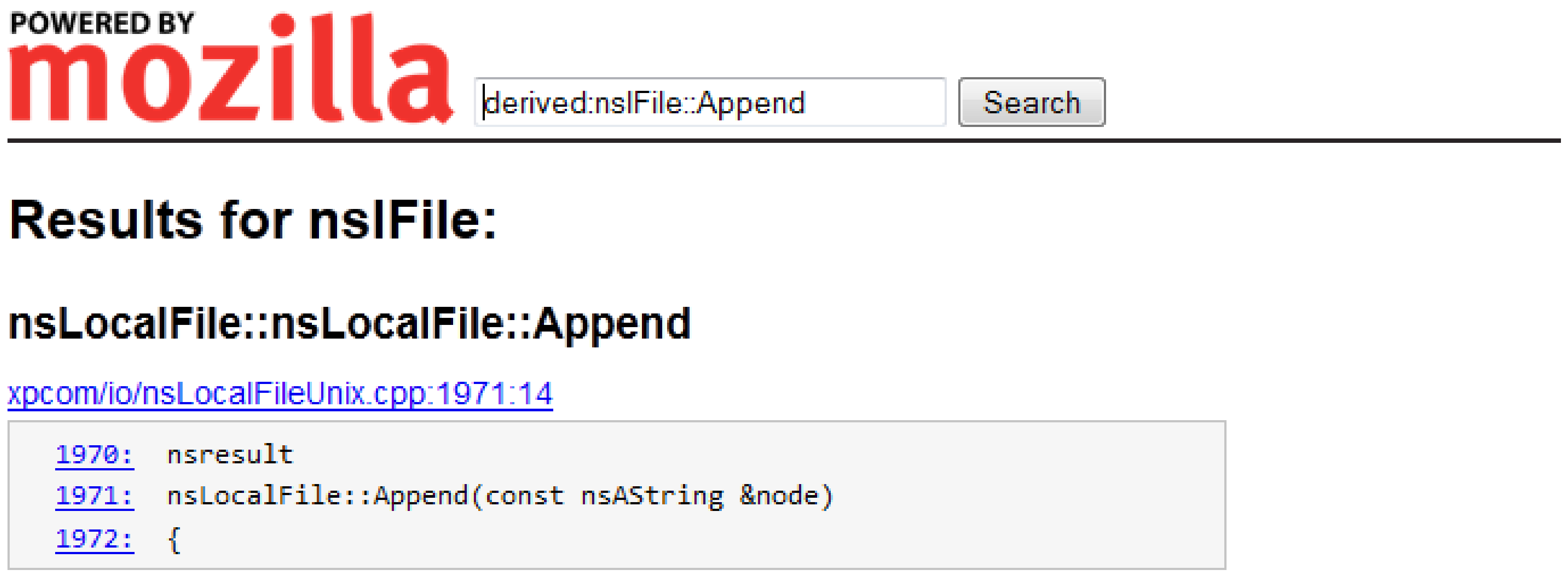
layout/generic/nsObjectFrame.h:68:7

```
68:  class nsObjectFrame : public nsObjectFrameSuper,
69:   public nsIObjectFrame,
```

nsTextFrame (Indirect)

layout/generic/nsTextFrame.h:68:7

Demo: Searching



derived:nsIFile::Append

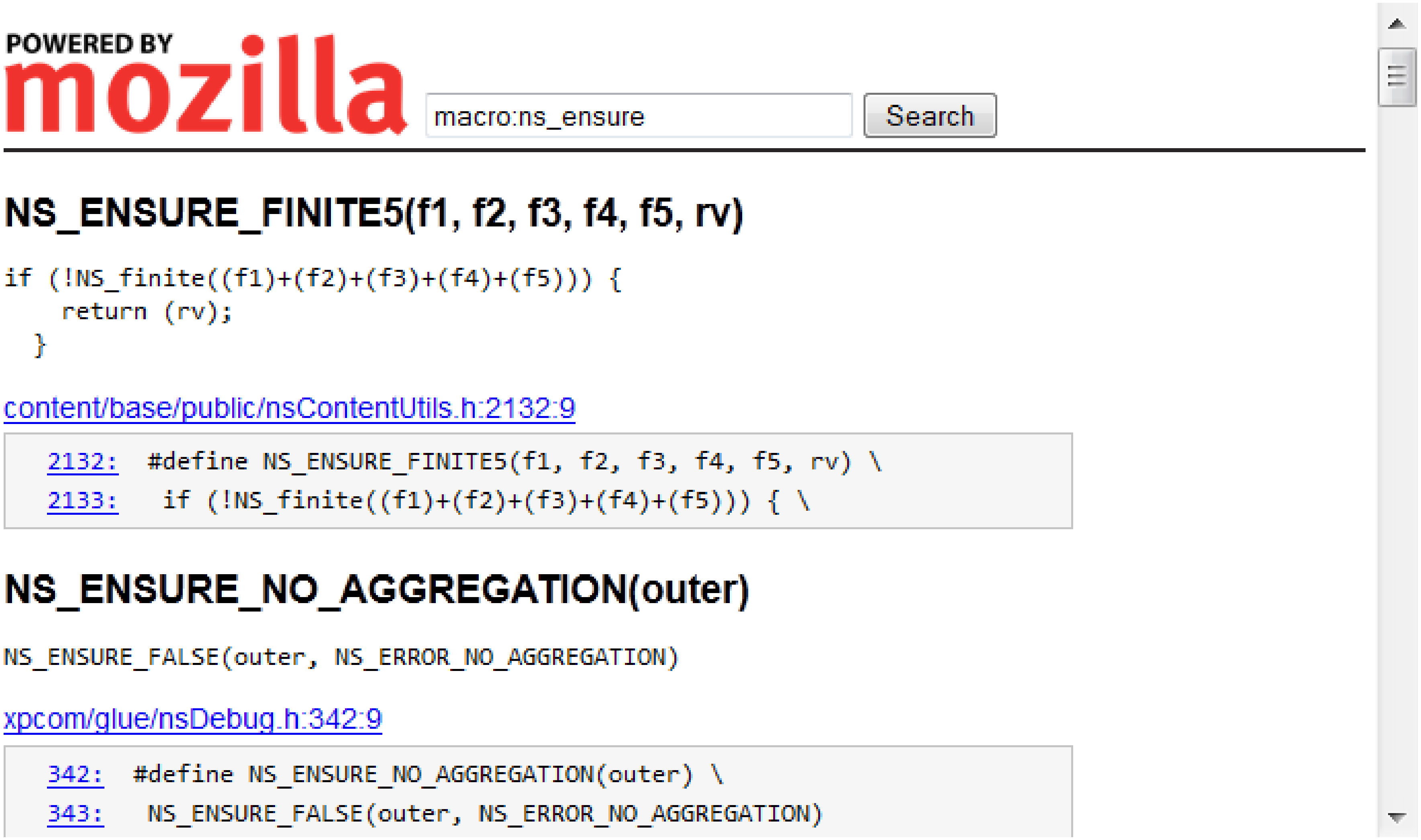
Results for nsIFile:

nsLocalFile::nsLocalFile::Append

xpcom/io/nsLocalFileUnix.cpp:1971:14

```
nsresult
nsLocalFile::Append(const nsAString &node)
{
```

Demo: Searching



NS_ENSURE_FINITE5(f1, f2, f3, f4, f5, rv)

```
if (!NS_finite((f1)+(f2)+(f3)+(f4)+(f5))) {
    return (rv);
  }
```

content/base/public/nsContentUtils.h:2132:9

```
#define NS_ENSURE_FINITE5(f1, f2, f3, f4, f5, rv) \
 if (!NS_finite((f1)+(f2)+(f3)+(f4)+(f5))) { \
```

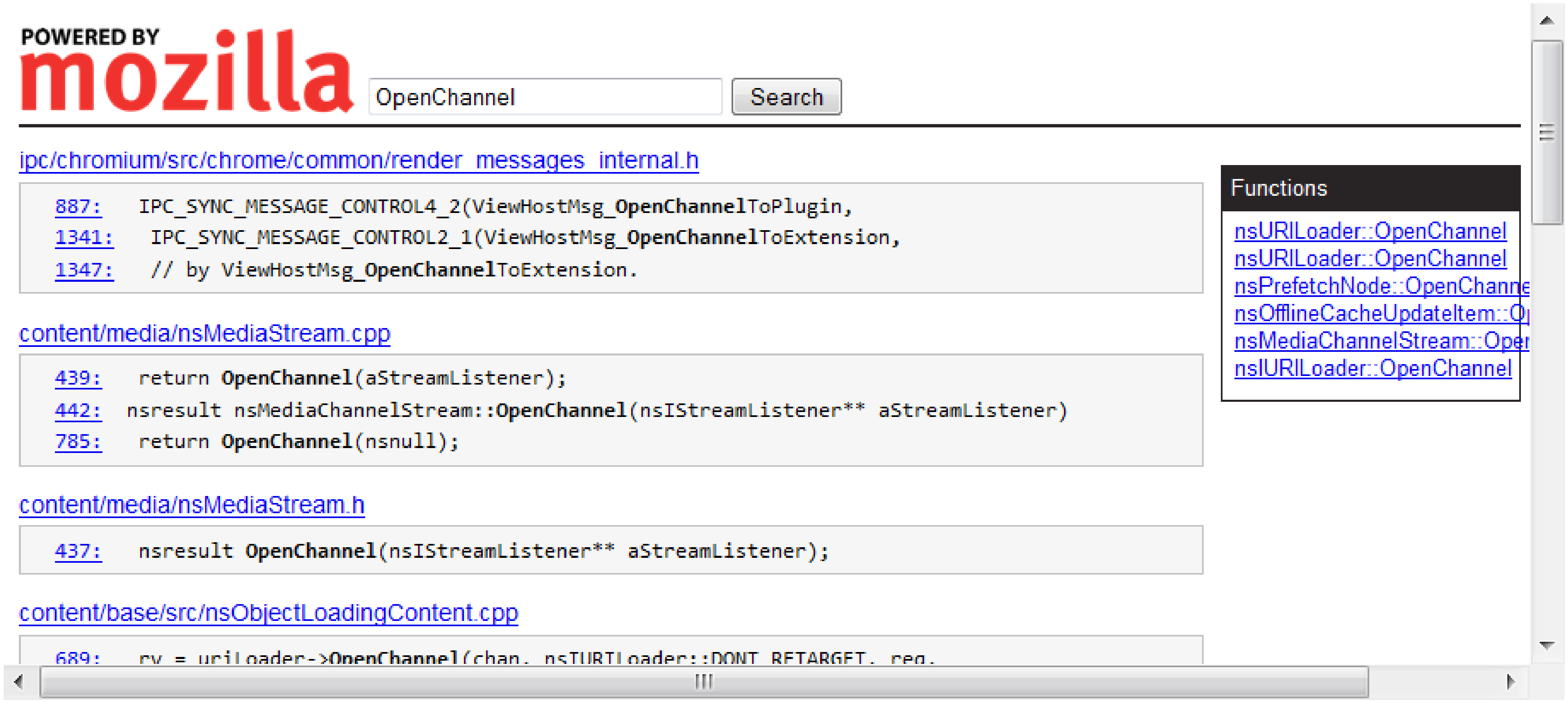
NS_ENSURE_NO_AGGREGATION(outer)

```
NS_ENSURE_FALSE(outer, NS_ERROR_NO_AGGREGATION)
```

xpcom/glue/nsDebug.h:342:9

```
#define NS_ENSURE_NO_AGGREGATION(outer) \
  NS_ENSURE_FALSE(outer, NS_ERROR_NO_AGGREGATION)
```

Demo: Searching



ipc/chromium/src/chrome/common/render_messages_internal.h

```
887:   IPC_SYNC_MESSAGE_CONTROL4_2(ViewHostMsg_OpenChannelToPlugin,
1341:   IPC_SYNC_MESSAGE_CONTROL2_1(ViewHostMsg_OpenChannelToExtension,
1347:   // by ViewHostMsg_OpenChannelToExtension.
```

content/media/nsMediaStream.cpp

```
439:   return OpenChannel(aStreamListener);
442: nsresult nsMediaChannelStream::OpenChannel(nsIStreamListener** aStreamListener)
785:   return OpenChannel(nsnull);
```

content/media/nsMediaStream.h

```
437:   nsresult OpenChannel(nsIStreamListener** aStreamListener);
```

content/base/src/nsObjectLoadingContent.cpp

```
689:   rv = uriLoader->OpenChannel(chan, nsIURILoader::DONT_RETARGET, req,
```

Functions

nsURILoader::OpenChannel
nsURILoader::OpenChannel
nsPrefetchNode::OpenChanne
nsOfflineCacheUpdateItem::O
nsMediaChannelStream::Ope
nsIURILoader::OpenChannel

Clang versus GCC

- Benefits of clang:
  - Easier to hook into preprocessor
  - More accurate location information
  - More accurate AST
- Benefits of gcc:
  - Easier to work with plugins
  - Cleaner data representation
  - Most programs compile with no problems

Future Work

- Support other languages (e.g., JavaScript)
- Support for multiple build configurations
- Integrate documentation
- Incremental reindexing

Wrap up

- DXR available at http://github.com/mozilla/dxr
- Live version at http://dxr.mozilla.org/mozilla

Any questions?