ANATOMY OF A BROWSER

Embedded Mobile Lizards

David Llewellyn-Jones
3rd December 2024

THE TRUTH

> *“The truth, the whole truth, and nothing but the truth.”*
>
> *King’s Bench Division, 1927*

THE TRUTH ABOUT BROWSERS

> *Subjective, incomplete, and easily overgeneralised.*
>
> *David, just now*

GECKO

1. Mozilla's rendering engine used in Firefox
2. Released in 2000, Netscape 6
3. Alternative to Blink, WebKit, Netsurf

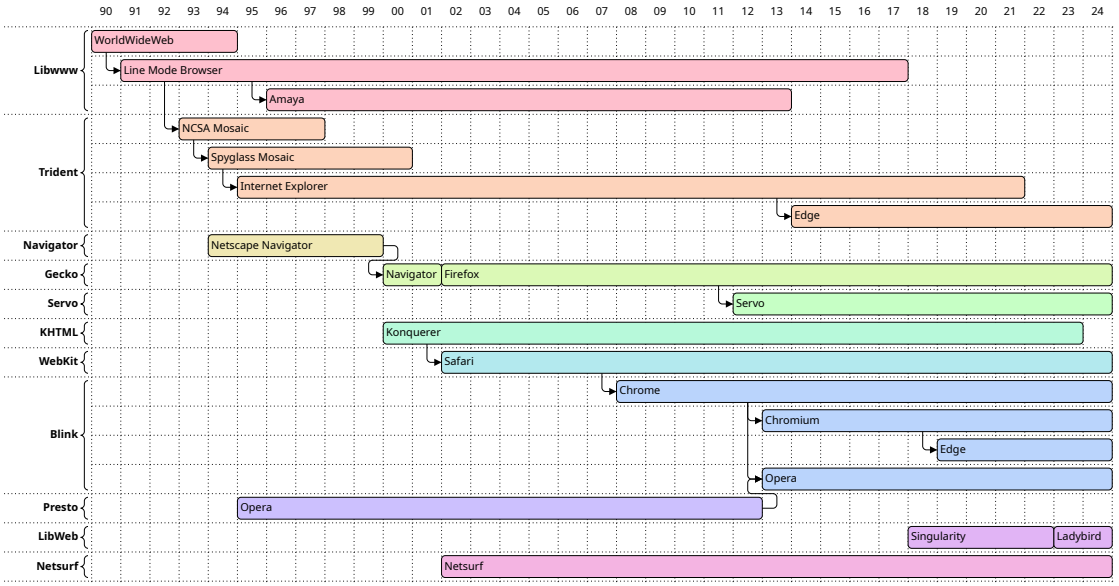
EMBEDDED BROWSERS

> *For many of us, a browser is an application... You click an icon on your graphical operating system, navigate somewhere with a URL bar, search, and so on... In contrast, an* ***embedded browser*** *is contained within another application or is built for a specific purpose and runs in an embedded system, and the application controlling the embedded browser does not provide all the typical features of browsers running in desktops.*
>
> *WPE, October 2024*

EMBEDDED BROWSERS

What does “embedded” mean?

1. Smaller resource footprint devices
2. Embeddable in other programs
3. Minimal user interface
4. All of the above


90 91 92 93 94 95 96 97 98 99 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Libwww
WorldWideWeb
Line Mode Browser
Amaya
Trident
NCSA Mosaic
Spyglass Mosaic
Internet Explorer
Edge
Navigator
Netscape Navigator
Gecko
Navigator
Firefox
Servo
Servo
KHTML
Konquerer
WebKit
Safari
Blink
Chrome
Chromium
Edge
Opera
Presto
Opera
LibWeb
Singularity
Ladybird
Netsurf
Netsurf

EMBEDDING GECKO

> *Gecko has limited embedding capability that is not well-documented, not well-maintained and not heavily invested in... We have at various points in history had embedding APIs/capabilities, but we have either dropped them (gtkmozembed) or let them bit-rot (IPCLite).*
>
> *Chris Lord, February 2016*

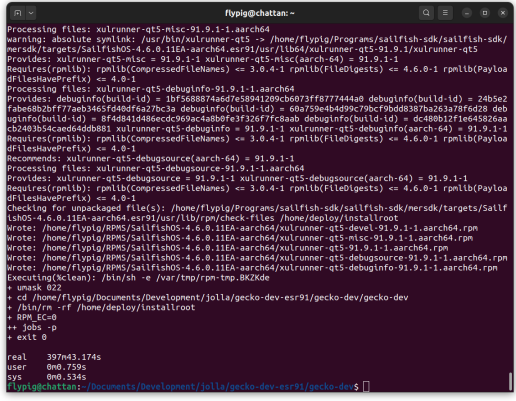
CONTEXT

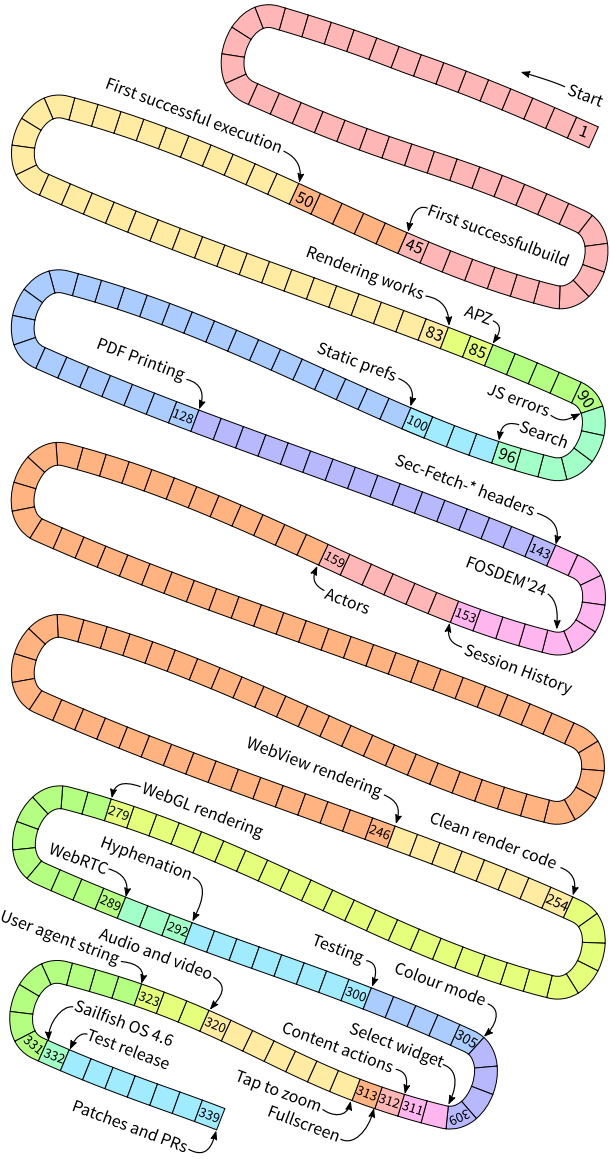
1. Work upgrading the browser on Sailfish OS
2. ESR 78.15.0 to ESR 91.9.1
3. Work conducted over 13 months: August 2023 - September 2024

https://
www.flypig.co.uk
/gecko

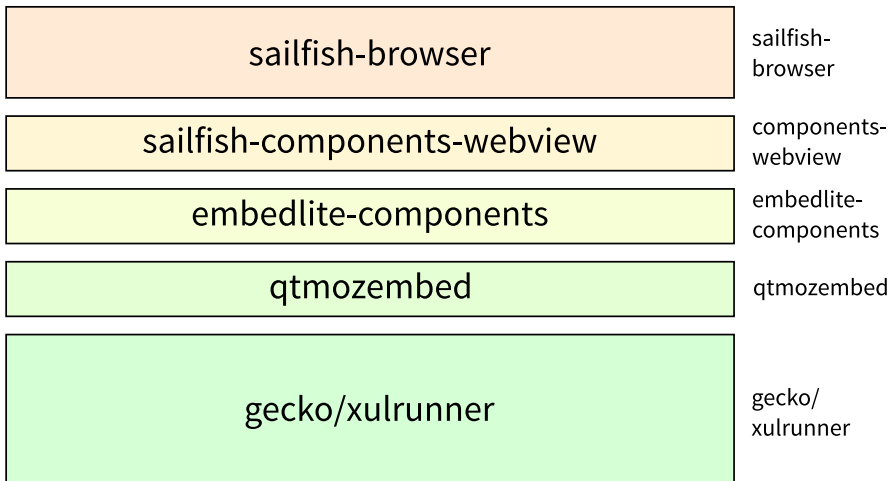
WORKING WITH THE BROWSER

1. Long build times
2. Complex build process
3. Random upstream changes (e.g. converting quote characters)
4. Removal of the entire EmbedLite rendering pipeline

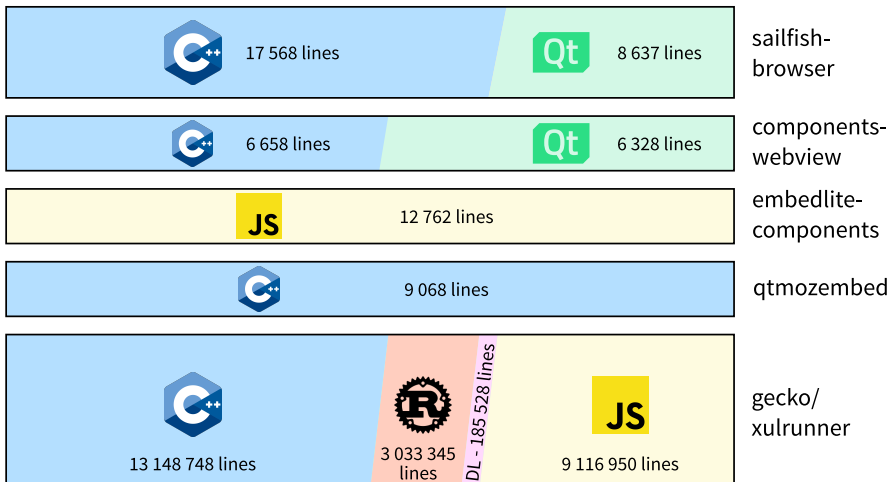



Start
1
First successful execution
50
First successfulbuild
45
Rendering works
83
APZ
85
JS errors
90
Search
96
Static prefs
100
PDF Printing
128
Sec-Fetch-* headers
143
FOSDEM'24
153
Session History
Actors
159
WebView rendering
246
Clean render code
254
WebGL rendering
279
Hyphenation
292
WebRTC
289
Testing
300
Colour mode
305
Audio and video
320
User agent string
323
Select widget
309
Content actions
311
Tap to zoom
312
Fullscreen
313
Sailfish OS 4.6
332
Test release
333
Patches and PRs
339

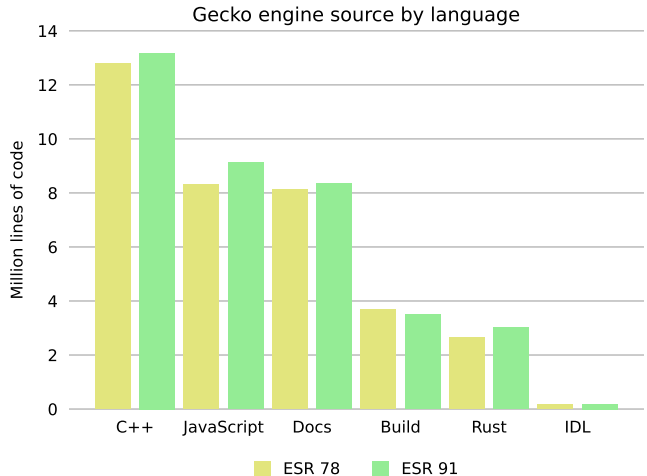
SAILFISH BROWSER



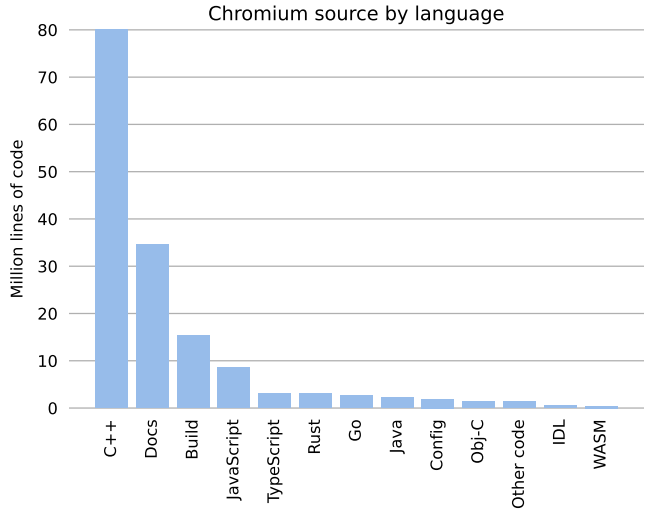
SAILFISH BROWSER



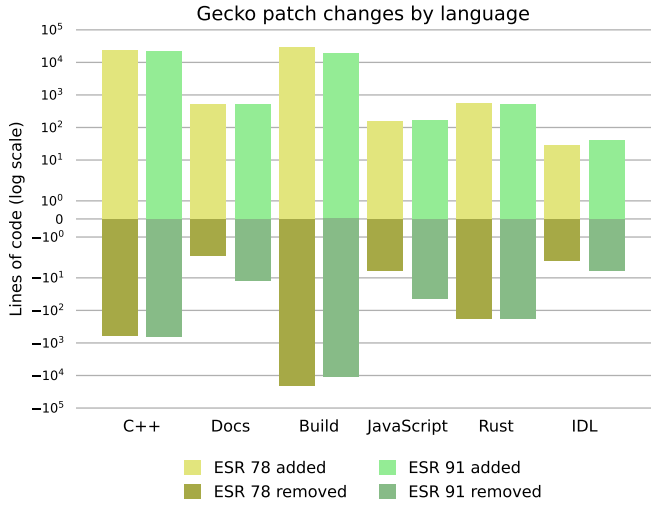
GECKO CODE DISTRIBUTION



CHROMIUM CODE DISTRIBUTION

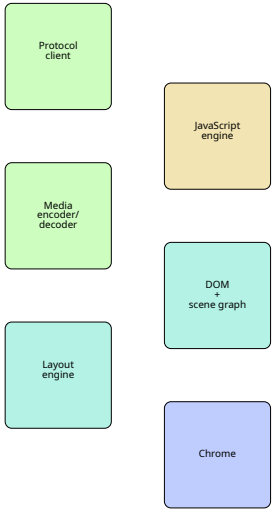


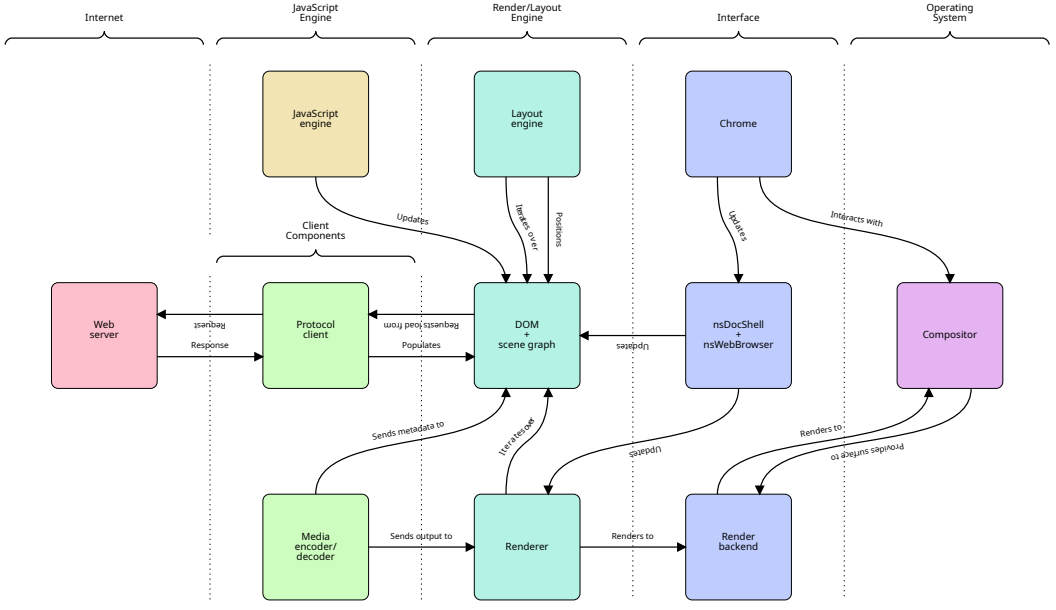
GECKO PATCHES



BASIC BROWSER STRUCTURE

1. Protocol client (HTTP/S, file, FTP,...)
2. JavaScript engine
3. Media decoder (JPG, PNG, audio, video)
4. DOM - Document Object Model
5. Layout/rendering engine (HTML, CSS, SVG)
6. Chrome




Internet
JavaScript Engine
Render/Layout Engine
Interface
Operating System
JavaScript engine
Layout engine
Chrome
Updates
Iterates over
Positions
Updates
Interacts with
Client Components
Web server
Protocol client
DOM + scene graph
nsDocShell + nsWebBrowser
Compositor
Request
Response
Requests load from
Populates
Updates
Sends metadata to
Iterates over
Updates
Renders to
Provides surface to
Media encoder/ decoder
Sends output to
Renderer
Renders to
Render backend

EMBEDDED BROWSER SURFACE

1. Rendering canvas
2. Event loop and user input (taps, keyboard, APZ)
3. Browser actions
4. JavaScript privileged execution
5. JavaScript in-DOM execution
6. Message sending/receiving

EMBEDDING APIS

Most browser embedding frameworks offer a similar API

1. Settings controller
2. Separate view widgets
3. Web controls (load, forwards, back)
4. Javascript execution
5. Message passing interface

Brief look at Chromium Embedded Framework, Qt WebEngine, Gecko WebView

```
class CefBrowserHost : public CefBaseRefCounted {
 public:
  static bool CreateBrowser(const CefWindowInfo& windowInfo,
                CefRefPtr<CefClient> client,
                const CefString& url,
                const CefBrowserSettings& settings,
                CefRefPtr<CefDictionaryValue> extra_info,
                CefRefPtr<CefRequestContext> request_context);
  CefRefPtr<CefBrowser> GetBrowser();
  void CloseBrowser(bool force_close);
  ...
  void StartDownload(const CefString& url);
  ...
};
```

```
class CefBrowser : public CefBaseRefCounted {
 public:
  bool CanGoBack();
  void GoBack();
  bool CanGoForward();
  void GoForward();
  bool IsLoading();
  void Reload();
  void StopLoad();
  bool HasDocument();
  CefRefPtr<CefFrame> GetMainFrame();
  ...
};
```

```
class CefFrame : public CefBaseRefCounted {
 public:
  CefRefPtr<CefBrowser> GetBrowser();
  void LoadURL(const CefString& url);
  CefString GetURL();
  CefString GetName();
  void Cut();
  void Copy();
  void Paste();
  void ExecuteJavaScript(const CefString& code,
                         const CefString& script_url,
                         int start_line);
  void SendProcessMessage(CefProcessId target_process,
                      CefRefPtr<CefProcessMessage> message);
  ...
};
```

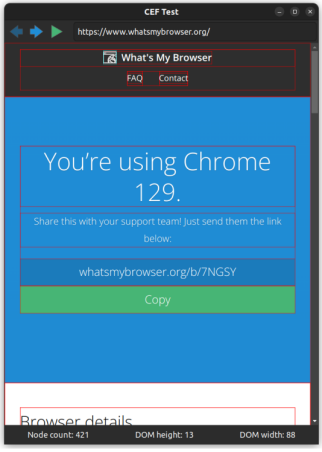
```
class QWebEnginePage : public QObject
{
    Q_PROPERTY(QUrl requestedUrl...)
    Q_PROPERTY(qreal zoomFactor...)
    Q_PROPERTY(QString title..)
    Q_PROPERTY(QUrl url READ...)
    Q_PROPERTY(bool loading...)
    ...

public:
    explicit QWebEnginePage(QObject *parent);

    virtual void triggerAction(WebAction action,
          bool checked);

    void findText(const QString &subString,
          FindFlags options, const std::function<void(const
          QWebEngineFindTextResult &)> &resultCallback);
    void load(const QUrl &url);
    void download(const QUrl &url, const QString &filename);
    void runJavaScript(const QString &scriptSource,
          const std::function<void(const QVariant &)> &
          resultCallback);
    void fullScreenRequested(QWebEngineFullScreenRequest
          fullScreenRequest);
    ...
};
```

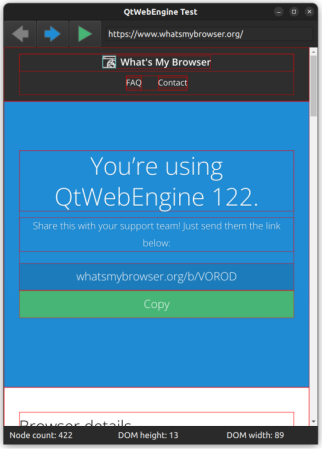
```
class QWebEngineView : public QWidget
{
    Q_PROPERTY(QString title...)
    Q_PROPERTY(QUrl url...)
    Q_PROPERTY(QString selectedText...)
    Q_PROPERTY(bool hasSelection...)
    Q_PROPERTY(qreal zoomFactor...)
    ...

public:
    explicit QWebEngineView(QWidget *parent);
    QWebEnginePage *page() const;
    void setPage(QWebEnginePage *page);

    void load(const QUrl &url);
    void findText(const QString &subString,
          FindFlags options, const std::function<void(const
          QWebEngineFindTextResult &)> &resultCallback);
    QWebEngineSettings *settings() const;
    void printToPdf(const QString &filePath, const
          QPageLayout &layout, const QPageRanges &ranges);
    ...

public slots:
    void stop();
    void back();
    void forward();
    void reload();
    ...
};
```

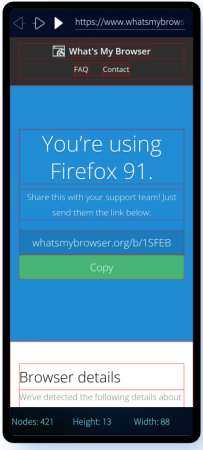
```
class WebEngine
{
public:
    static WebEngine *instance();
    ...

public slots:
    void addObserver(const QString &aTopic);
    void removeObserver(const QString &aTopic);
    void notifyObservers(const QString &topic, const QString
          &value);
    void addComponentManifest(const QString &manifestPath);
    void setProfile(const QString &);
    ...

public signals:
    void recvObserve(const QString message, const QVariant
          data);
    ...
};
```

```
class WebEngineSettings
{
    Q_PROPERTY(bool autoLoadImages...)
    Q_PROPERTY(bool javascriptEnabled...)
    Q_PROPERTY(bool popupEnabled...)
    Q_PROPERTY(CookieBehavior cookieBehavior...)
    ...

public:
    static WebEngineSettings *instance();
    ...
};
```

```
class QuickMozView : public QQuickItem
{
    Q_PROPERTY(QUrl url...)
    Q_PROPERTY(QString title...)
    Q_PROPERTY(bool canGoBack...)
    Q_PROPERTY(bool canGoForward...)
    Q_PROPERTY(int loadProgress...)
    Q_PROPERTY(QString httpUserAgent...)
    ...

public:
    void runJavaScript(const QString &script, const QJSValue
          &callback, const QJSValue &errorCallback);
    void load(const QString&, const bool& fromExternal);
    void reload();
    void stop();
    void goBack();
    void goForward();
    void loadHtml(const QString &html, const QUrl &baseUrl);
    ...
};
```


CEF Test
https://www.whatsmybrowser.org/
What's My Browser
FAQ
Contact
You're using Chrome 129.
Share this with your support team! Just send them the link below:
whatsmybrowser.org/b/7NGSY
Copy
Browser details
Node count: 421
DOM height: 13
DOM width: 88


QtWebEngine Test
https://www.whatsmybrowser.org/
What's My Browser
FAQ
Contact
You're using QtWebEngine 122.
Share this with your support team! Just send them the link below:
whatsmybrowser.org/b/VOROD
Copy
Node count: 422
DOM height: 13
DOM width: 89


https://www.whatsmybrow
What's My Browser
FAQ
Contact
You're using Firefox 91.
Share this with your support team! Just send them the link below:
whatsmybrowser.org/b/15FEB
Copy
Browser details
We've detected the following details about
Nodes: 421
Height: 13
Width: 88

```
function collect_node_stats(global_ctx, local_ctx, node) {
    // Update the context
    local_ctx.depth += 1;
    local_ctx.breadth.push(node.childNodes.length);
    global_ctx.nodes += 1;
    global_ctx.maxdepth = Math.max(
        local_ctx.depth, global_ctx.maxdepth);

    // Recurse into child nodes
    for (child of node.childNodes) {
        child_ctx = structuredClone(local_ctx);
        child_ctx.breadth = local_ctx.breadth.slice(
            0, local_ctx.depth + 1);
        child_ctx = collect_node_stats(
            global_ctx, child_ctx, child);

        // Recalculate the child breadths
        for (let i = local_ctx.depth + 1;
            i < child_ctx.breadth.length; ++i) {
            local_ctx.breadth[i] = (local_ctx.breadth[i]||0)
                + child_ctx.breadth[i];
        }
    }

    // Paint the DOM red
    if (node.style) {
        node.style.boxShadow = "inset 0px 0px 1px 0.5px red";
    }

    // Move back up the tree
    local_ctx.depth -= 1;
    return local_ctx;
}
```

```
function node_stats() {
    // Data available to all nodes
    let global_ctx = {
        "nodes": 0,
        "maxdepth": 0,
        "maxbreadth": 0
    }

    // Data local to the node and shared with the parent
    let local_ctx = {
        "depth": 0,
        "breadth": [1]
    }

    // Off we go
    local_ctx = collect_node_stats(
        global_ctx, local_ctx, document);
    global_ctx.maxbreadth = Math.max.apply(
        null, local_ctx.breadth);
    return global_ctx;
}

// Return the results (only strings allowed)
JSON.stringify(node_stats())
```

INTERFACE DEFINITION LANGUAGE

1. Interaction between native code and JavaScript
2. Requires type conversion and memory management
3. IDL - generate native/JS-compatible interfaces
4. Make native calls from JavaScript
5. Make JavaScript calls from native

IDL Type	Javascript	C++ in	Rust in
`boolean`	boolean	`bool`	`bool`
`char`	string	`char`	`c_char`
`double`	number	`double`	`f64`
`float`	number	`float`	`f32`
`long`	number	`int32_t`	`i32`
`long long`	number	`int64_t`	`i64`
`short`	number	`uint16_t`	`u16`
`string`	string	`const char*`	`*const c_char`
`wchar`	string	`char16_t`	`i16`
`wstring`	string	`const char16_t*`	`*const i16`
`Array<T>`	array	`const nsTArray<T>&`	`*const ThinVec<T>`

```
/**
 * This interface allows creating various prompts that have a specific parent.
 */
[scriptable, uuid(2803541c-c96a-4ff1-bd7c-9cb566d46aeb)]
interface nsIPromptFactory : nsISupports
{
  /**
   * Returns an object implementing the specified interface that creates
   * prompts parented to aParent.
   */
  void getPrompt(in mozIDOMWindowProxy aParent, in nsIIDRef iid,
                 [iid_is(iid),retval] out nsQIResult result);
};
```

```
class NS_NO_VTABLE nsIPromptFactory : public nsISupports {
 public:

  NS_DECLARE_STATIC_IID_ACCESSOR(NS_IPROMPTFACTORY_IID)

  /* Used by ToJSValue to check which scriptable interface is implemented. */
  using ScriptableInterfaceType = nsIPromptFactory;

  /* void getPrompt (in mozIDOMWindowProxy aParent, in nsIIDRef iid, [iid_is (iid), retval] out nsQIResult result); */
  JS_HAZ_CAN_RUN_SCRIPT NS_IMETHOD GetPrompt(mozIDOMWindowProxy *aParent, const nsIID & iid, void * * result) = 0;

};
```

```
function LoginManagerPromptFactory() {
  Services.obs.addObserver(this,
    "quit-application-granted", true);
  ...
}

LoginManagerPromptFactory.prototype = {
  classID : Components.ID("{72de694e-6c88-11e2-a4ee-...}"),
  QueryInterface: ChromeUtils.generateQI([
    Ci.nsIPromptFactory,
    Ci.nsIObserver,
    Ci.nsISupportsWeakReference,
  ]),

  observe : function (subject, topic, data) {
    if (topic == "quit-application-granted") {
      this._cancelPendingPrompts();
      ...
    }
  },

  getPrompt : function (aWindow, aIID) {
    var prompt = new LoginManagerPrompter()
                     .QueryInterface(aIID);
    prompt.init(aWindow, this);
    return prompt;
  },
  ...
}; // end of LoginManagerPromptFactory implementation
```

```
nsresult nsPromptFactory::Init() {
  AddObserver(this, "quit-application-granted", true);
  ...
}

NS_IMETHODIMP
nsPromptFactory::Observe(nsISupports* subject,
                         const char* topic, const char16_t* data) {
  if (!strcmp(topic, "quit-application-granted")) {
    this.CancelPendingPrompts();
    ...
  }

  return NS_OK;
}

NS_IMETHODIMP
nsPromptFactory::GetPrompt(mozIDOMWindowProxy *aParent,
                           const nsIID & iid, void * * result) {
  nsCOMPtr<nsISupports> login = new LoginManagerPrompter();
  nsCOMPtr<nsILoginManagerPrompter> prompt =
           login.QueryInterface(iid, getter_AddRefs(prompt));
  if (prompt) {
    prompt.Init(aParent, this);
  }

  *result = prompt.forget().take();
  return NS_OK;
}
```

```
getPrompt(domWin, iid) {
  if (iid.equals(Ci.nsIAuthPrompt2) || iid.equals(Ci.nsIAuthPrompt)) {
    try {
      let pwmgr = Cc[
        "@mozilla.org/passwordmanager/authpromptfactory;1"
      ].getService(Ci.nsIPromptFactory);
      return pwmgr.getPrompt(domWin, iid);
    } catch (e) {
      Cu.reportError(
        "nsPrompter: Delegation to password manager failed: " + e
      );
    }
  }
},
```

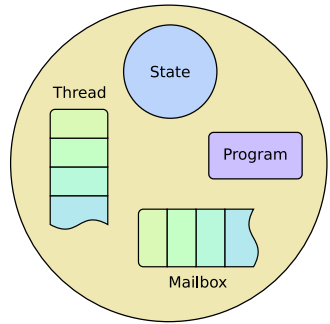
```
NS_IMETHODIMP
nsWindowWatcher::GetNewPrompter(mozIDOMWindowProxy* aParent,
                                nsIPrompt** aResult) {
  // This is for backwards compat only. Callers should just use the prompt
  // service directly.
  nsresult rv;
  nsCOMPtr<nsIPromptFactory> factory =
      do_GetService("@mozilla.org/prompter;1", &rv);
  NS_ENSURE_SUCCESS(rv, rv);
  return factory->GetPrompt(aParent, NS_GET_IID(nsIPrompt),
                            reinterpret_cast<void**>(aResult));
}
```

```
#define JSVAL_TAG_SHIFT 47

class alignas(8) Value {
 private:
  uint64_t asBits_;

  static uint64_t bitsFromDouble(double d) {
    d = CanonicalizeNaN(d);
    return mozilla::BitwiseCast<uint64_t>(d);
  }

 public:
  static constexpr uint64_t bitsFromTagAndPayload(
    JSValueTag tag, uint64_t payload) {
    return (uint64_t(tag) << JSVAL_TAG_SHIFT) | payload;
  }
  ...

  /*** Value type queries ***/
  bool isUndefined() const {
    return asBits_ == JSVAL_SHIFTED_TAG_UNDEFINED;
  }

  bool isInt32() const {
    return toTag() == JSVAL_TAG_INT32;
  }

  bool isDouble() const {
    return ValueIsDouble(asBits_);
  }

  bool isString() const {
    return toTag() == JSVAL_TAG_STRING;
  }
  ...
```

```
  /*** Mutators ***/
  void setUndefined() {
    asBits_ = bitsFromTagAndPayload(JSVAL_TAG_UNDEFINED, 0);
  }

  void setInt32(int32_t i) {
    asBits_ = bitsFromTagAndPayload(
                                JSVAL_TAG_INT32, uint32_t(i));
  }

  void setDouble(double d) {
    asBits_ = bitsFromDouble(d);
  }

  void setString(JSString* str) {
    asBits_ = bitsFromTagAndPayload(
                             JSVAL_TAG_STRING, uint64_t(str));
  }
  ...

  /*** Extract the value's typed payload ***/
  int32_t toInt32() const {
    return int32_t(asBits_);
  }

  double toDouble() const {
    return mozilla::BitwiseCast<double>(asBits_);
  }

  JSString* toString() const {
    return unboxGCPointer<JSString, JSVAL_TAG_STRING>();
  }
  ...

} JS_HAZ_GC_POINTER MOZ_NON_PARAM;
```

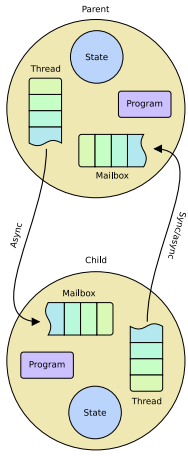
INTER-THREAD/PROCESS MESSAGES

1. Inter-Thread and Inter-Process Message Passing
2. Inspired by the *actor model* - Parent-Child pairs
3. Actors bound to an endpoint, bound to a thread
4. Actor framework schedules tasks to run on event target
5. Inter-[thread|process] Protocol Definition Language



INTER-THREAD/PROCESS MESSAGES

1. Message ordering is presevered
2. Only child actors can send sync messages or break the connection
3. Look like asynchronous method calls
4. IPDL files generate parent and child C++ code
5. Clients subclass the generated interfaces



```
nested(upto inside_sync) sync protocol PEmbedLiteView
{
child:
    async LoadURL(nsString url, bool aFromExternal);
    ...
parent:
    async OnLoadStarted(nsCString aLocation);
    ...
};
```

```
class PEmbedLiteViewParent :
    public mozilla::ipc::IProtocol
{
    ...
public:
    [[nodiscard]] bool
    SendLoadURL(
            const nsString& url,
            const bool& aFromExternal);
    ...
};
```

```
class PEmbedLiteViewChild :
    public mozilla::ipc::IProtocol
{
    ...
public:
    bool
    SendOnLoadStarted(const nsCString& aLocation);
    ...
};
```

```
class EmbedLiteViewParent :
    public PEmbedLiteViewParent
{
    ...
protected:
    virtual mozilla::ipc::IPCResult RecvOnLoadStarted(
        const nsCString &aLocation
    );
    ...
};
```

```
class EmbedLiteViewChild :
    public PEmbedLiteViewChild
{
    ...
protected:
    virtual mozilla::ipc::IPCResult RecvLoadURL(
        const nsString &,
        const bool& aFromExternal
    );
    ...
};
```

```
auto PEmbedLiteViewParent::OnMessageReceived(
  const Message& msg__) -> PEmbedLiteViewParent::Result
{
  switch (msg__.type()) {
  ...
  case PEmbedLiteView::Msg_OnLoadStarted__ID:
    {
      PickleIterator iter__{msg__};
      nsCString aLocation{};

      if ((!(ReadIPDLParam((&(msg__)), (&(iter__)), this, (&(
            aLocation))))) {
        FatalError("Error deserializing 'nsCString'");
        return MsgValueError;
      }
      ...
      msg__.EndRead(iter__, msg__.type());
      if ((!((static_cast<EmbedLiteViewParent*>(this))->
            RecvOnLoadStarted(std::move(aLocation)))) {
        mozilla::ipc::ProtocolErrorBreakpoint("Handler
              returned error code!");
        // Error handled in mozilla::ipc::IPCResult
        return MsgProcessingError;
      }

      return MsgProcessed;
    }
  ...
  default:
    return MsgNotKnown;
  }
}
```

WORKING WITH GECKO

1. Very mature codebase
2. An ecosystem of software techniques
3. Deep stack of technologies
4. Constant introduction of new ideas
5. A critical operating system component

FURTHER INFO

Dev Diary	`https://www.flypig.co.uk/gecko`
Slides source	`https://github.com/llewelld/techtalk-gecko-dev`
Gecko source	`https://github.com/llewelld/gecko-dev`
CEF	`https://bitbucket.org/chromiumembedded/cef`
QtWebEngine	`https://doc.qt.io/qt-6/qtwebengine-index.html`
WebView	`https://sailfishos.org/develop/docs/sailfish-components-webview`