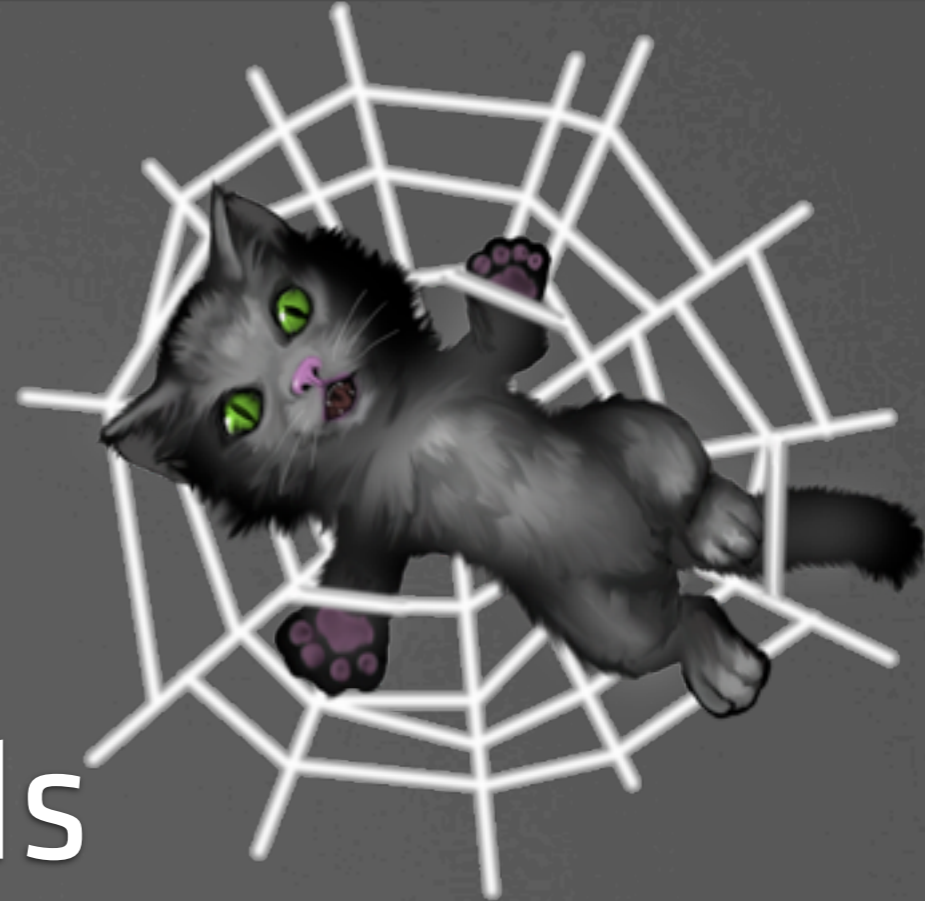


# Hacking WebKit & Its JavaScript Engines





Jarred Nicholls  
Work @ Sencha  
WebKit Committer

Doing webkitty things...



# Our Mission if we choose to accept it

Define: What is WebKit?

Some Simple Ways to Leverage WebKit

Little investment, Large ROI

Web Inspector Hacking

“Headless” WebKit



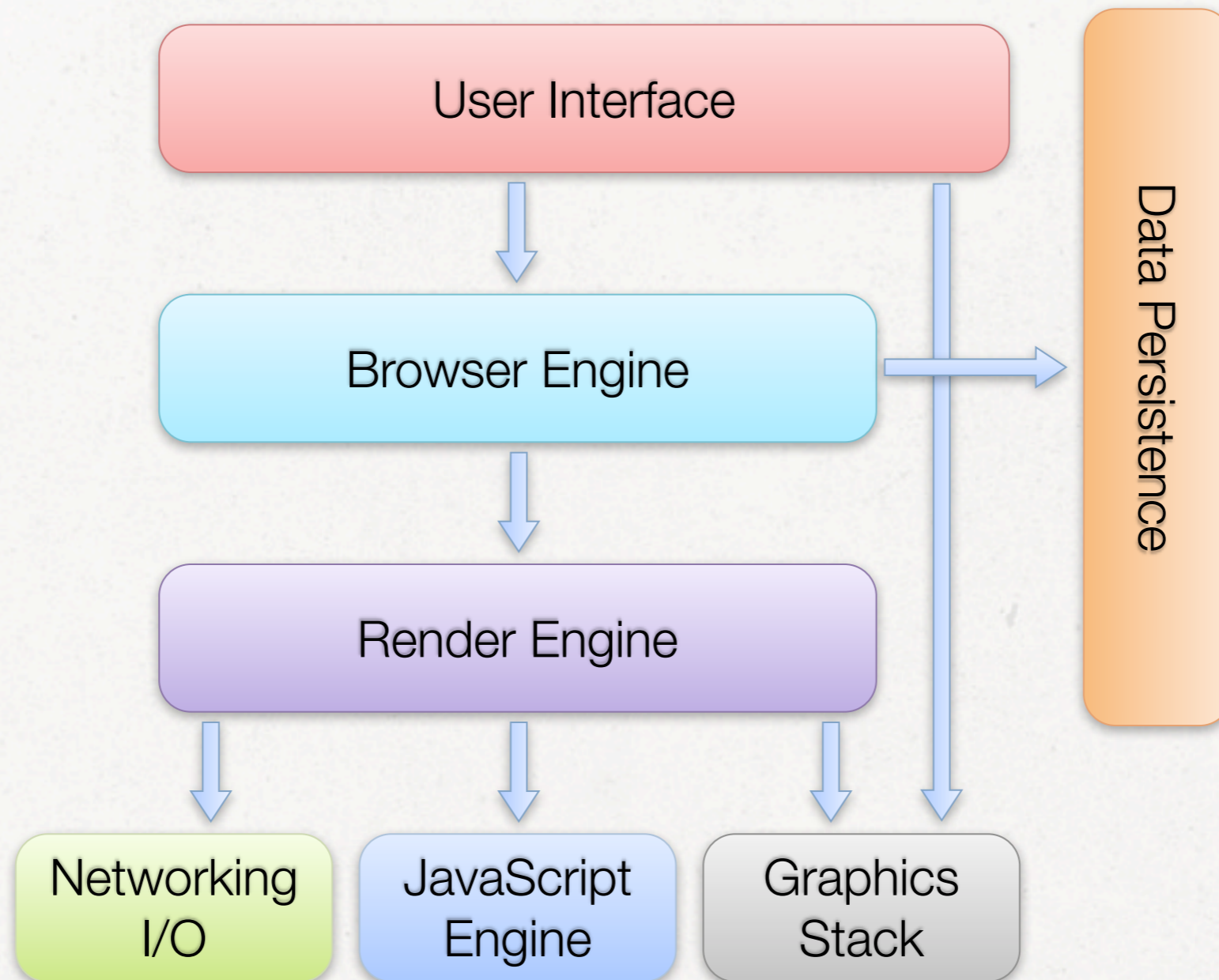
# What is WebKit?



#

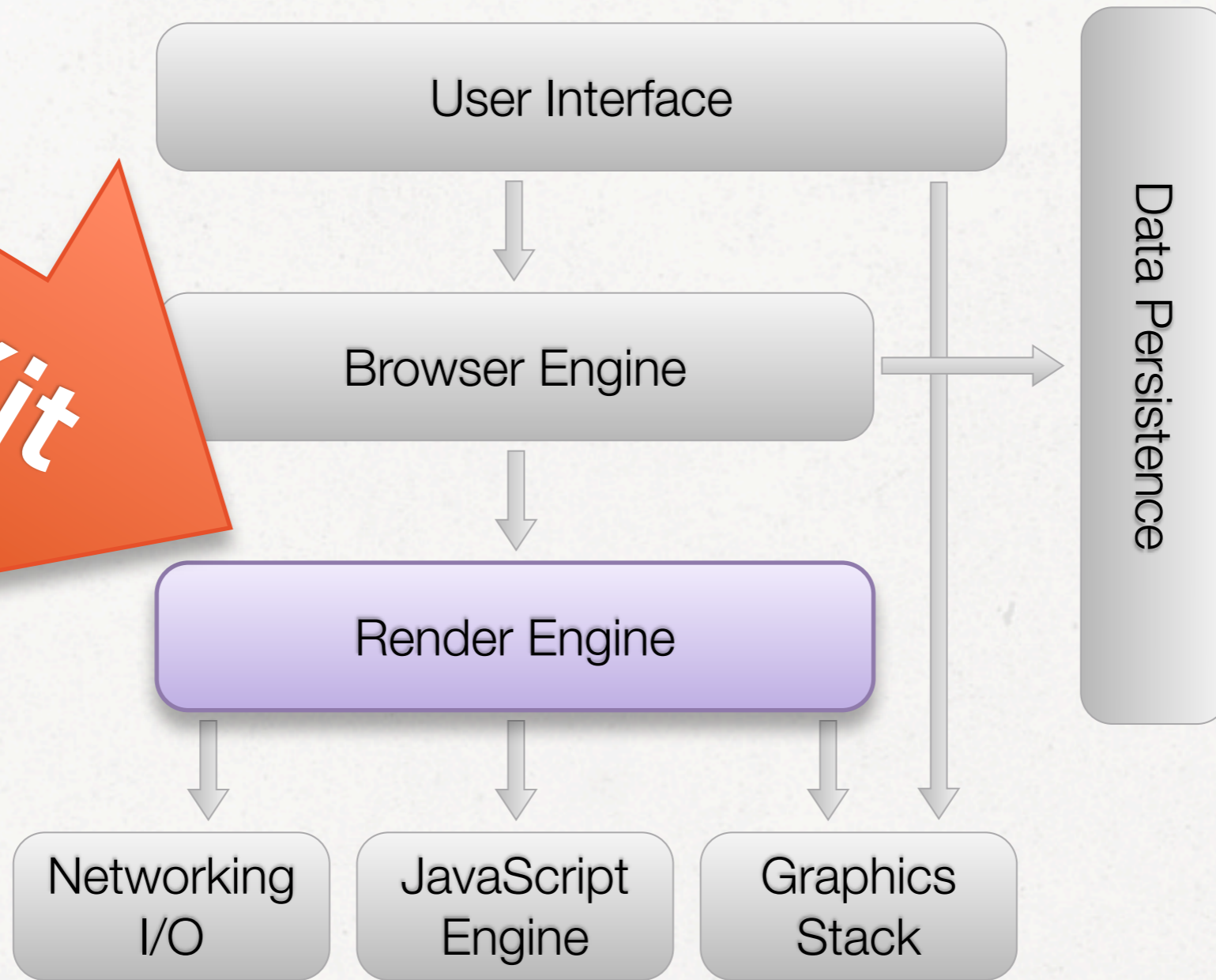


# Browser at a High Level



# Browser at a High Level

**WebKit**



#





C





# WebKit is EVERYWHERE!





**Even in your living room!**

## Portability

The WebKit project seeks to address a variety of needs. ***We want to make it reasonable to port WebKit to a variety of desktop, mobile, embedded and other platforms.*** We will provide the infrastructure to do this with tight platform integration, reusing native platform services where appropriate and providing friendly embedding APIs.

<http://www.webkit.org/projects/goals.html>



**BlackBerry**

**iOS**



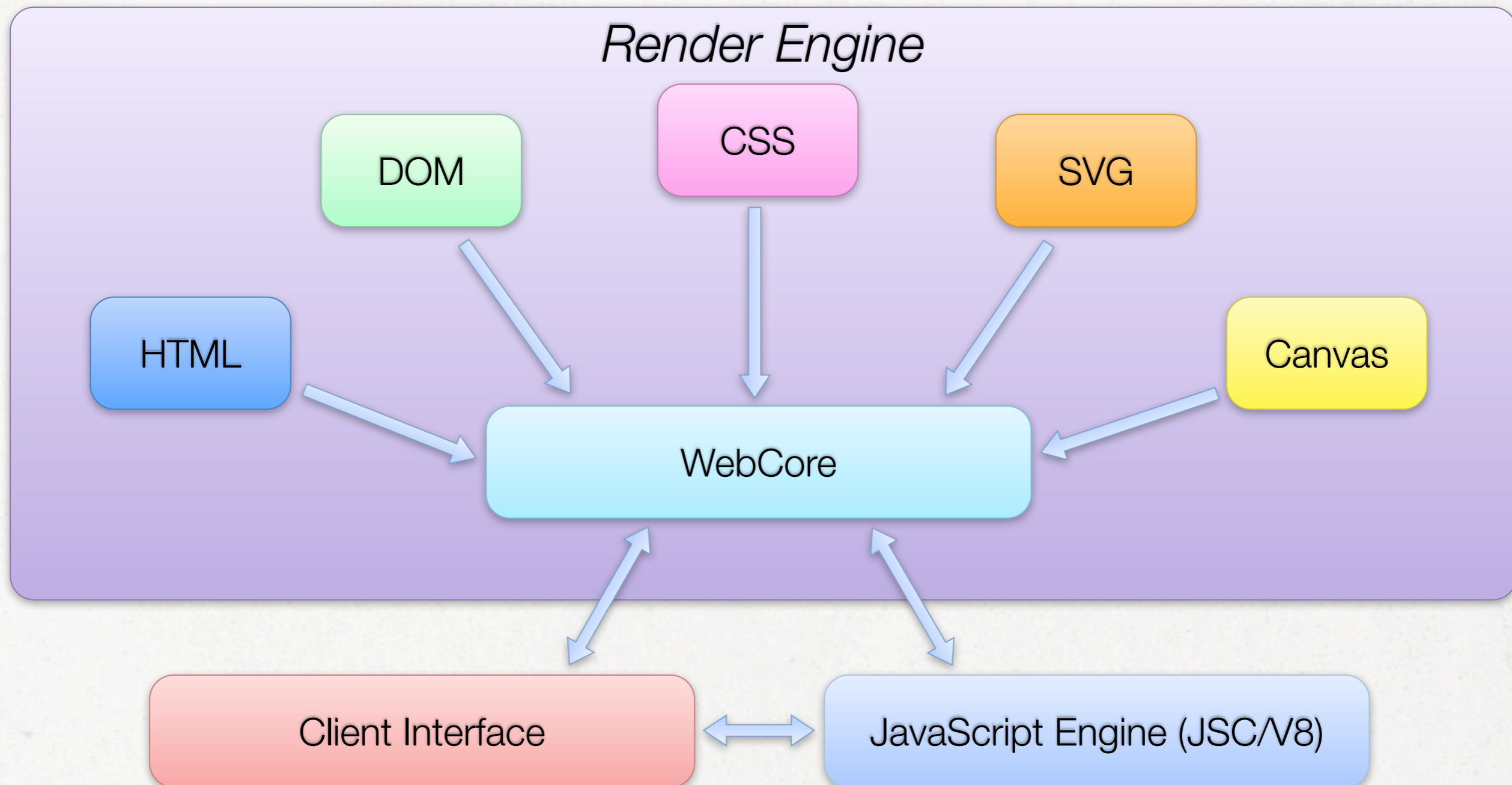
**How?**



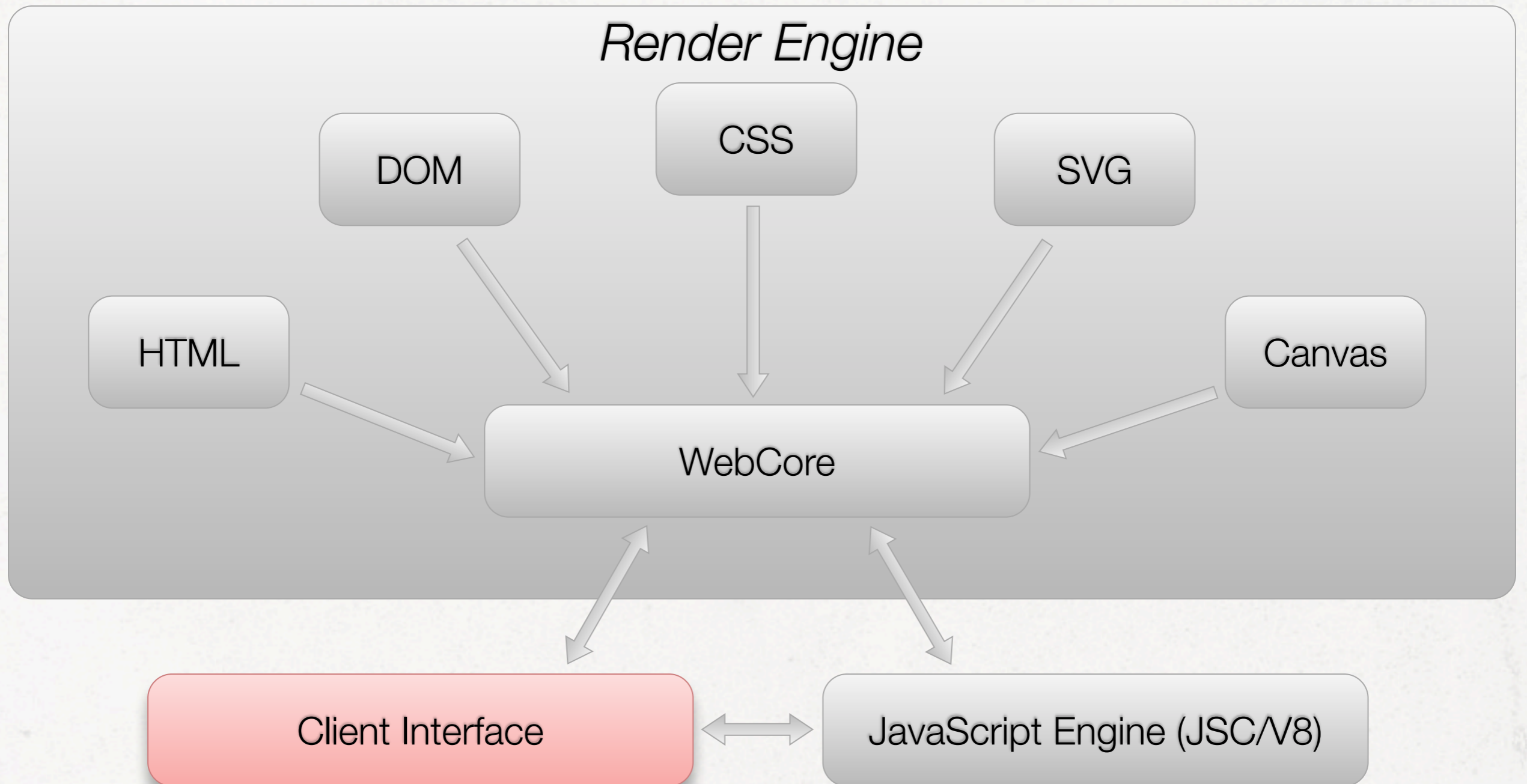


# Client Interface

# WebKit Components

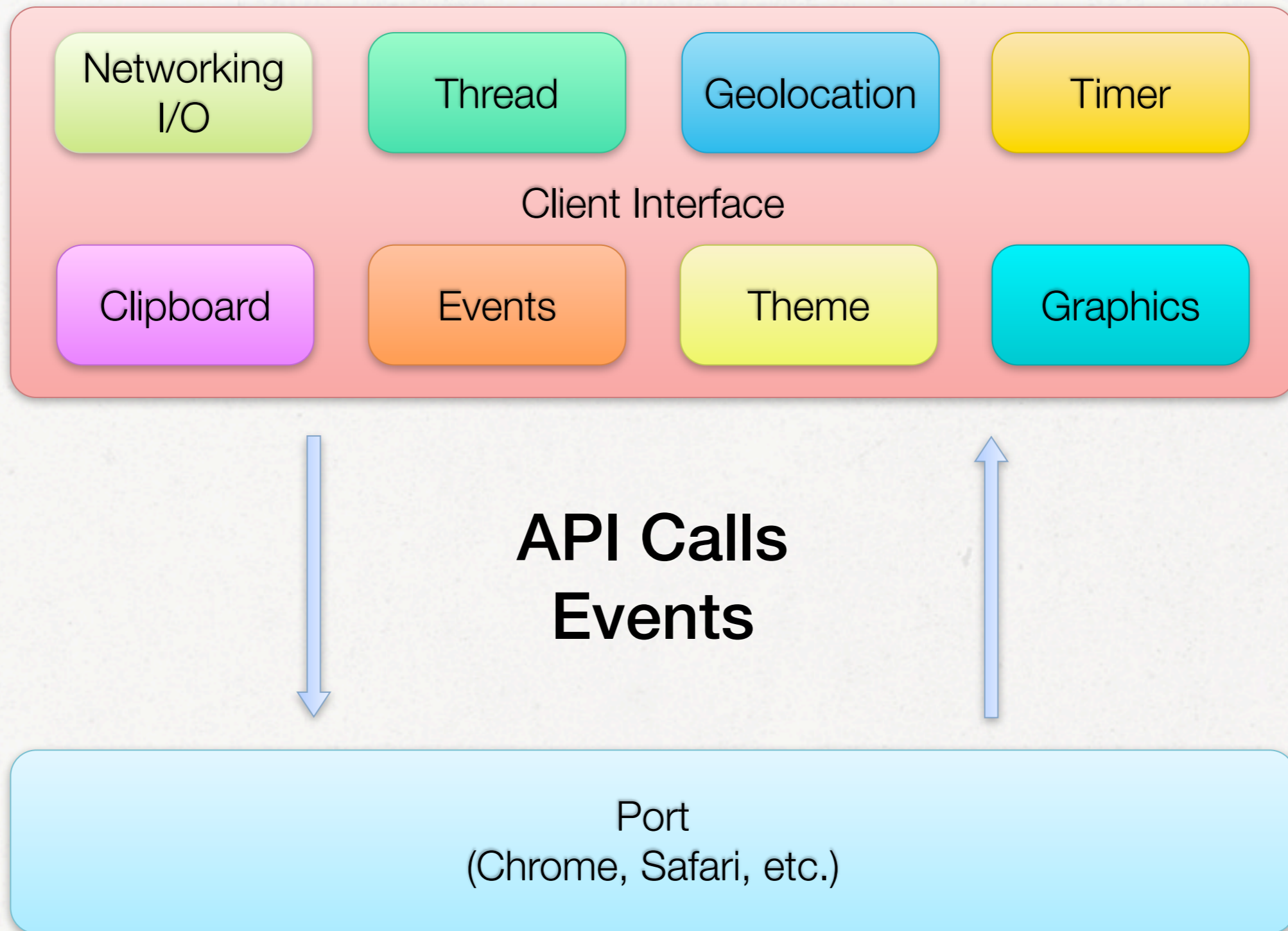


# WebKit Components

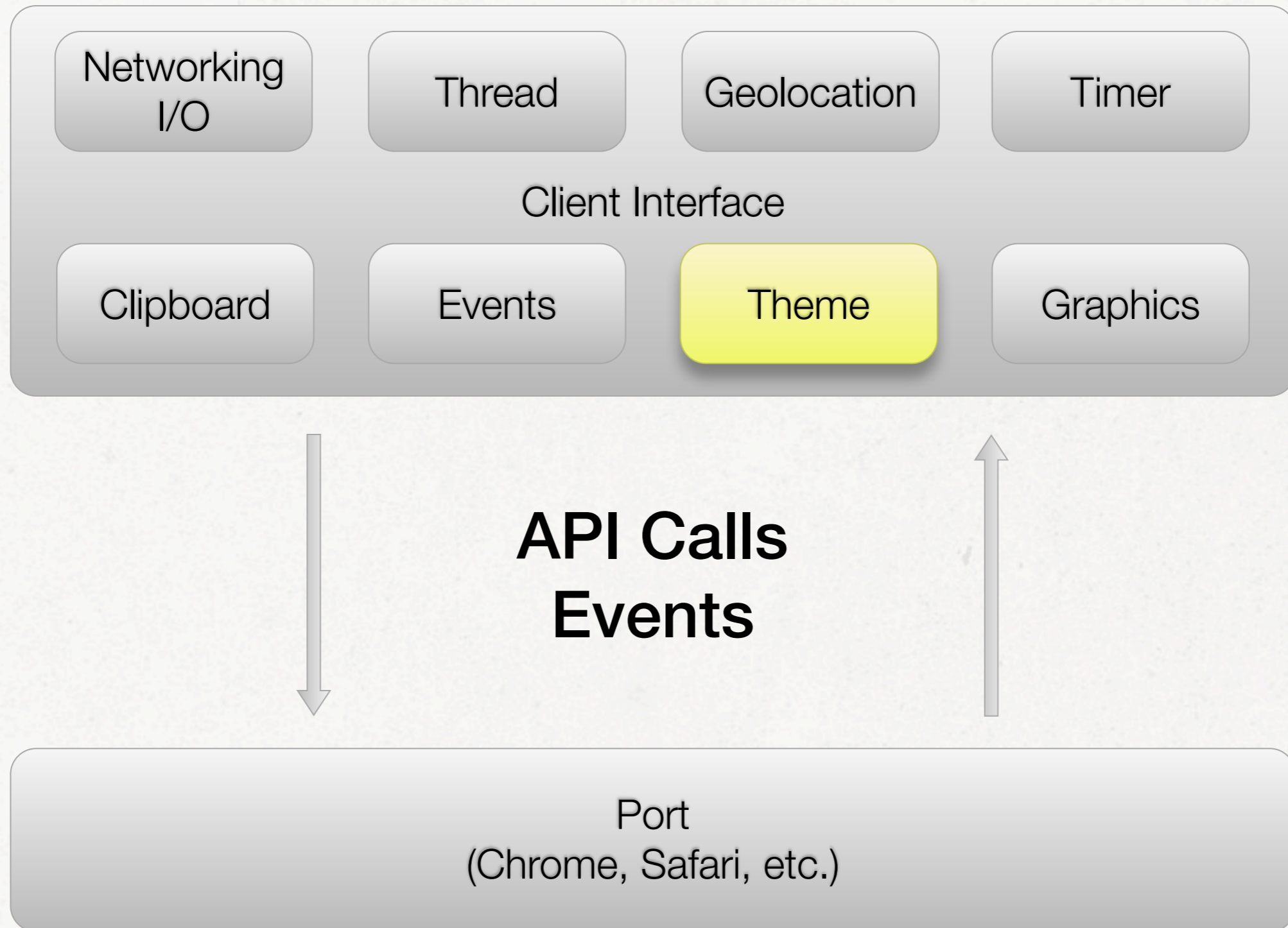




# Port Abstraction



# Port Abstraction



`<input type="number" />`



Paint me  
some spin  
buttons, STAT!

I know how to  
do that!



**WebKit**

**Port**

`<input type="number" />`



**WebKit**



`<input type="number" />`

Piece of cake!

lookin' sexy!



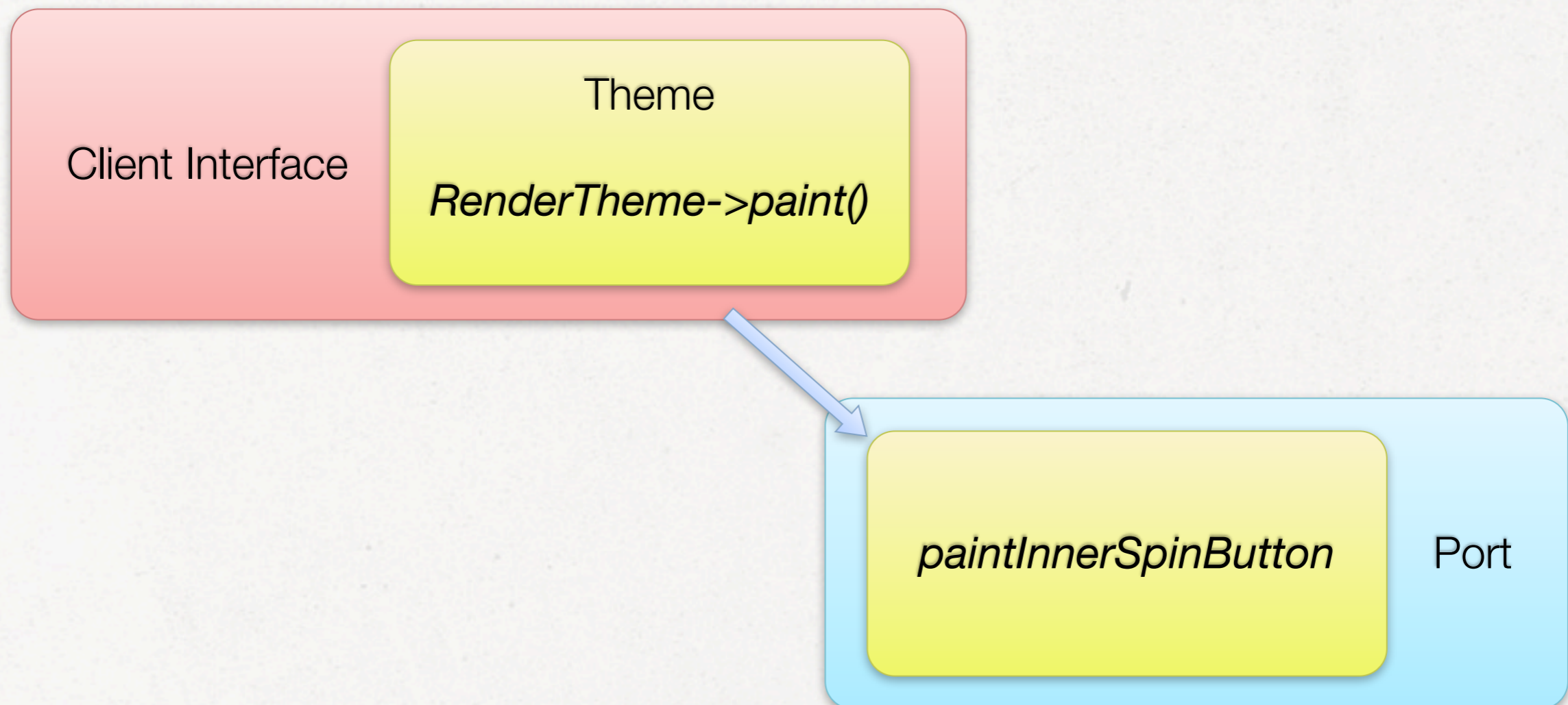
**Port**



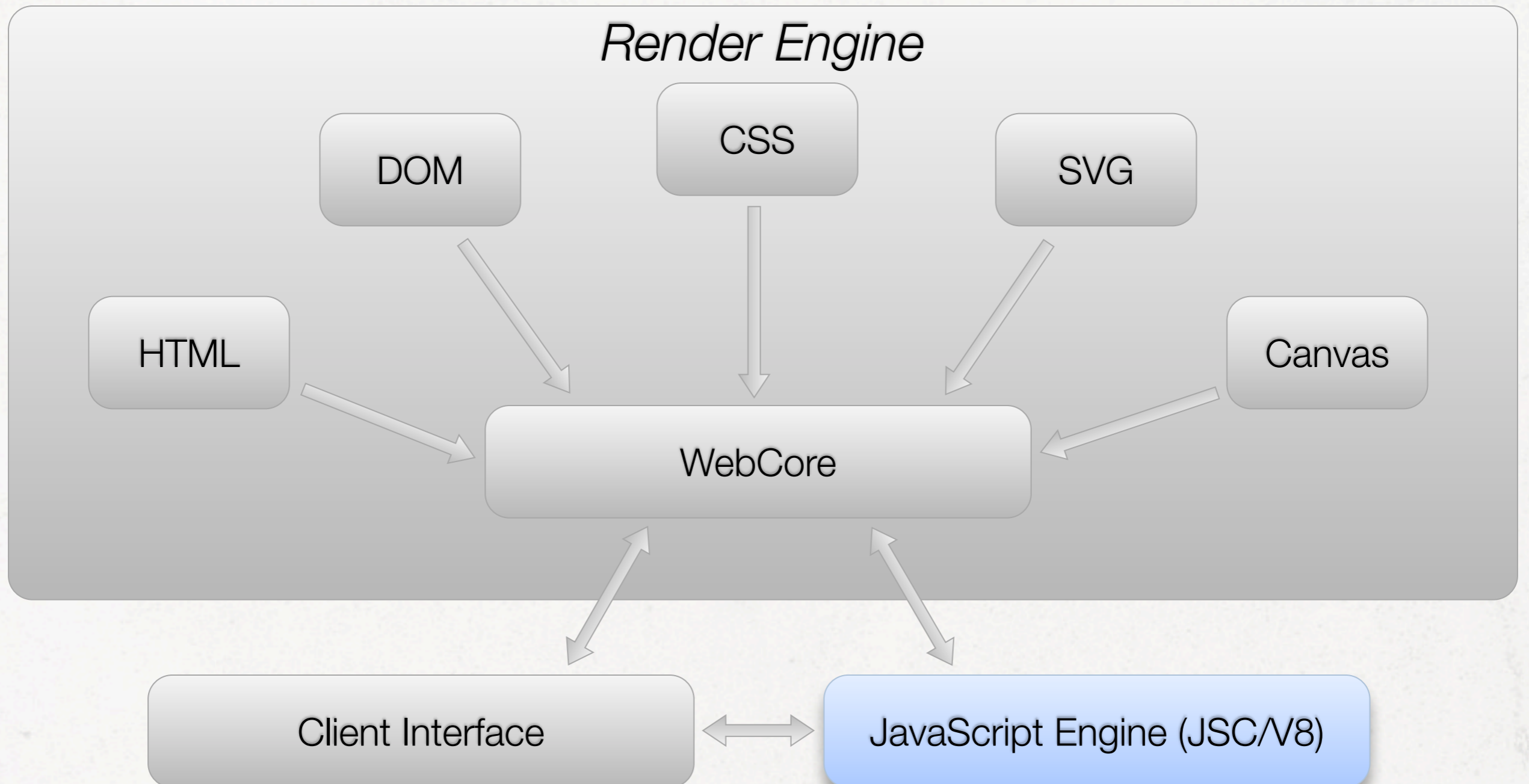
# <input type="number" />

## Theme Interface

```
bool paintInnerSpinButton(RenderObject*, const  
                          PaintInfo&, const IntRect&);
```



# WebKit Components

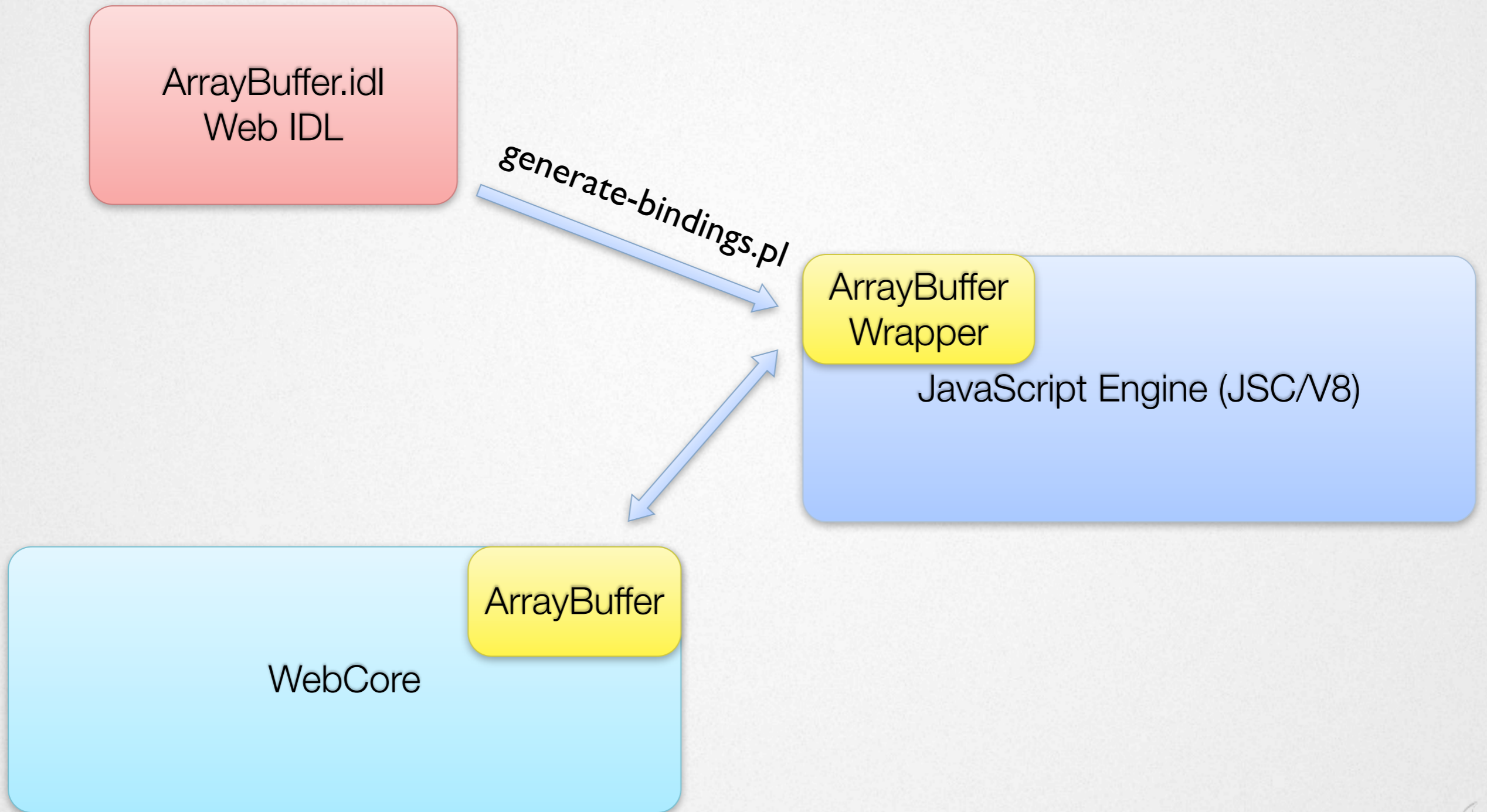




# JavaScript Bindings



# JavaScript Bindings



# ArrayBuffer.h

```
class ArrayBuffer : public RefCounted<ArrayBuffer> {  
public:  
    // ...  
    inline unsigned byteLength() const;  
  
    inline PassRefPtr<ArrayBuffer> slice(int begin, int end) const;  
    inline PassRefPtr<ArrayBuffer> slice(int begin) const;  
    // ...  
};
```



# Web[Kit]IDL

```
interface [  
    JSGenerateIsReachable=Impl,  
    CustomConstructor,  
    ConstructorParameters=1,  
    JSNoStaticTables  
] ArrayBuffer {  
    readonly attribute int byteLength;  
    ArrayBuffer slice(in long begin, in [Optional] long end);  
};
```

- > perl generate-bindings.pl --generator JS ArrayBuffer.idl
- > perl generate-bindings.pl --generator V8 ArrayBuffer.idl



# JavaScriptCore

```
JSValue jsArrayBufferByteLength(ExecState* exec, JSValue slotBase, PropertyName)
{
    JSArrayBuffer* castedThis = jsCast<JSArrayBuffer*>(asObject(slotBase));
    UNUSED_PARAM(exec);
    ArrayBuffer* impl = static_cast<ArrayBuffer*>(castedThis->impl());
    JSValue result = jsNumber(impl->byteLength());
    return result;
}
```



# V8

```
static v8::Handle<v8::Value> byteLengthAttrGetter(v8::Local<v8::String> name,  
    const v8::AccessorInfo& info)  
{  
    INC_STATS("DOM.ArrayBuffer.byteLength._get");  
    ArrayBuffer* imp = V8ArrayBuffer::toNative(info.Holder());  
    return v8::Integer::New(imp->byteLength());  
}
```



# Hacking WebKit For Fun & Profit



# Building WebKit Nightly





# Prerequisites



# Prerequisites

## Mac

Xcode 3.1.4+ & command-line tools  
> `which g++ => /usr/bin/g++`

## Windows

Don't even bother (see "Building Chromium" :)  
...but if you're really brave: <http://www.webkit.org/building/tools.html>



# Get the Source



# Get the Source

## SVN

```
svn checkout http://svn.webkit.org/repository/webkit/trunk WebKit
```

## Git (RECOMMENDED!!!)

```
git clone git://git.webkit.org/WebKit.git
```

*or*

```
git clone git://github.com/WebKit/webkit.git WebKit
```



# Build WebKit



# Build WebKit

- > cd WebKit
- > Tools/Scripts/build-webkit

Go have lunch...

- > Tools/Scripts/run-safari



# Build WebKit

- > cd WebKit
- > Tools/Scripts/build-webkit

Go have lunch...

- > Tools/Scripts/run-safari

*runs local safari, but w/ new  
webkit framework*



# Building Chromium





# Prerequisites



# Prerequisites

**Depot Tools:** <http://www.chromium.org/developers/how-tos/install-depot-tools>

## Mac

Xcode 4.3+ & command-line tools  
> which clang => /usr/bin/clang

## Windows

Visual Studio (Express)  
Windows SDK  
DirectX SDK  
Windows Driver Development Kit

Details: <http://www.chromium.org/developers/how-tos/build-instructions-windows#TOC-Prerequisite-software>



# Get the Source



# Get the Source

- > mkdir chromium && cd chromium
- > gclient config <http://src.chromium.org/chrome/trunk/src>
- > gclient sync

Details: <http://dev.chromium.org/developers/how-tos/get-the-code>



# Build Chromium



# Build Chromium

- > `cd chromium/src`
- > `GYP_GENERATORS=ninja gclient runhooks --force`
- > `ninja -C out/Release chrome`

Go on vacation...

- > `open out/Release/Chromium.app`



# Build Chromium

- > cd chromium/src
- > GYP\_GENERATORS=ninja gclient runhooks --force
- > ninja -C out/Release chrome

*super fast "make" equivalent  
comes with depot tools*

Go on vacation...

- > open out/Release/Chromium.app



**Let's Hack**





# Web Inspector

The screenshot displays the Web Inspector interface with the following components:

- Top Bar:** Contains tabs for Elements, Resources, Network, Scripts, Timeline, Profiles, Audits, and Console. A search bar labeled "Search Elements" is on the right.
- DOM Tree (Left):** Shows the document structure. The selected element is `<body class="mainsite home guest chrome mac hi_again ext-webkit ext-chrome ext-mac" >`. Its children include `<header id="header">`, `<div id="content-wrapper">`, `<div id="home-hero">`, `<section>`, `<section>`, `<!-- /#content -->`, `<!-- /#content-wrapper -->`, `<footer id="footer">`, and several `<script>` and `<img>` tags.
- Computed Style Panel (Right):** Shows the computed style for the selected element. It includes a "Show inherited" checkbox and a "Styles" section with icons for adding, deleting, and resetting styles. The "Matched CSS Rules" section lists three rules:
  - Rule 1: `media="screen, http://www.sencha.com/projection, print"` for `body` with properties: `background: #EDED`, `color: #434343`, `font: 14px/1.4em "Helvetica Neue", A...`, and `-webkit-font-smoothing: antialiased`.
  - Rule 2: `media="screen, http://www.sencha.com/projection, print"` for `body` with property: `line-height: 1`.
  - Rule 3: `media="screen, http://www.sencha.com/projection, print"` for `html`.
- Bottom Bar:** Shows the breadcrumb path: `html.wf-klavikaweb-n3-active.wf-active > body.mainsite.home.guest.chrome.mac.hi_again`.

# Add in Hacks

*inspector.js*

```
var re = /command=([^&]+)/,  
    match = re.exec(location.href),  
    command;
```

```
if (!match) {  
    return;  
}
```

```
try {  
    command = eval('(' + decodeURIComponent(match[1]) + ')');  
} catch (e) {  
    return;  
}
```

```
switch (command.type) {  
    default:  
        break;  
}
```

*command handler*



# Add in Hacks

*inspector.js*

*persist data*

```
var saveData = function(filePath, data) {  
  filePath = '/Users/jarred/qconny/' + filePath;  
  
  var xhr = new XMLHttpRequest();  
  xhr.onload = function() {  
    alert('Data saved successfully to ' + filePath);  
  };  
  xhr.onerror = function() {  
    alert('Data failed to save to ' + filePath);  
  };  
  xhr.open('GET', '/savedata');  
  xhr.setRequestHeader('DevTools-Data', JSON.stringify(data));  
  xhr.setRequestHeader('DevTools-FilePath', filePath);  
  xhr.send(null);  
};
```



# Add in Hacks

*devtools\_http\_handler\_impl.cc*

```
void DevToolsHttpHandlerImpl::OnHttpRequest(  
    int connection_id,  
    const net::HttpServerRequestInfo& info) {  
    // ...  
  
    if (info.path.find("/savedata") == 0) {  
        BrowserThread::PostTask(  
            BrowserThread::FILE,  
            FROM_HERE,  
            base::Bind(&DevToolsHttpHandlerImpl::OnSaveDataRequestFILE,  
                this,  
                connection_id,  
                info));  
  
        return;  
    }  
    // ...  
}
```

*persist data* →



# Add in Hacks

`devtools_http_handler_impl.cc`

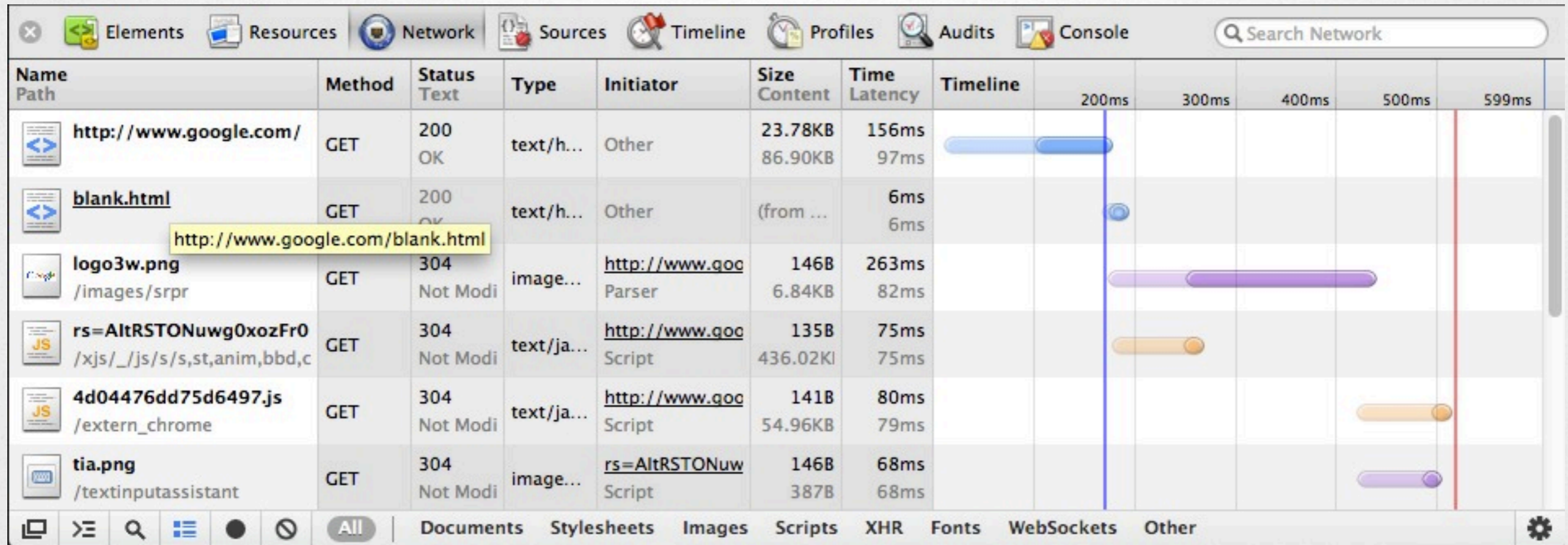
```
void DevToolsHttpHandlerImpl::OnSaveDataRequestFILE(int connection_id, const
net::HttpServerRequestInfo& info) {
    bool success = false;
    FILE* fp = fopen(info.GetHeaderValue("DevTools-FilePath").c_str(), "wb");
    if (fp) {
        std::string data = info.GetHeaderValue("DevTools-Data");
        fwrite(data.data(), 1, data.size(), fp);
        fclose(fp);
        success = true;
    }

    Send200(connection_id, success ? "true" : "false", "application/json; charset=UTF-8");
}
```

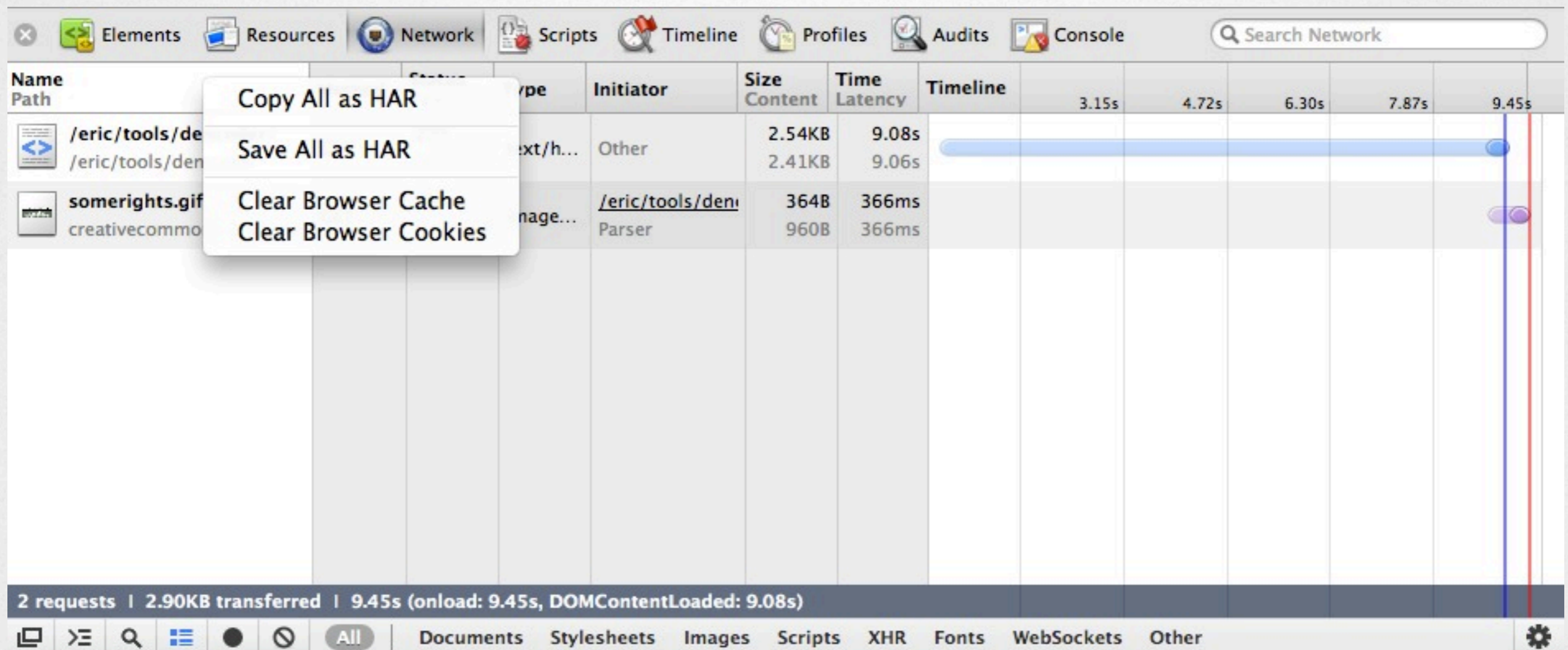
*persist data*



# Web Inspector Network



# Export Network HAR Data



The screenshot shows the Chrome DevTools Network tab with a context menu open over a network request. The menu options are:

- Copy All as HAR
- Save All as HAR
- Clear Browser Cache
- Clear Browser Cookies

The network request table is as follows:

Name	Path	Type	Initiator	Size Content	Time Latency	Timeline
	/eric/tools/de			2.54KB	9.08s	
	/eric/tools/den	text/h...	Other	2.41KB	9.06s	
somerights.gif		image...	/eric/tools/den	364B	366ms	
creativecommo			Parser	960B	366ms	

At the bottom of the Network tab, a summary bar shows: 2 requests | 2.90KB transferred | 9.45s (onload: 9.45s, DOMContentLoaded: 9.08s). The filter bar at the bottom is set to 'All' and includes categories: Documents, Stylesheets, Images, Scripts, XHR, Fonts, WebSockets, and Other.

More info: <http://www.softwareishard.com/blog/har-1.2-spec/>



# Add in Hacks

*inspector.js*

*exportHAR cmd*

```
var exportHAR = function(cmd) {  
  // Force a reload of the page to grab network stats.  
  PageAgent.reload(true /* bypass cache */);  
  
  setTimeout(function() {  
    var networkData =  
WebInspector.panels.network._networkLogView._getExportData();  
    saveData((cmd.filename || WebInspector.inspectedPageDomain)  
      + '.har', networkData);  
  }, cmd.timeout);  
};  
  
case 'exportHAR':  
  exportHAR(command);  
  break;
```






# Add in Hacks

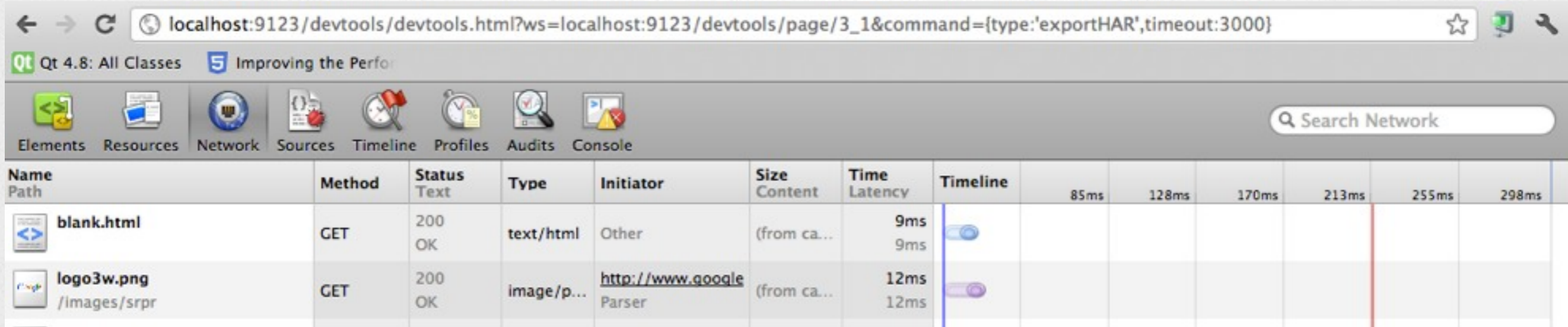
NetworkPanel.js

export data fn





```
_getExportData: function() {  
    return { log: (new WebInspector.HARLog(this._requests)).build() };  
}
```



# Trigger Hack



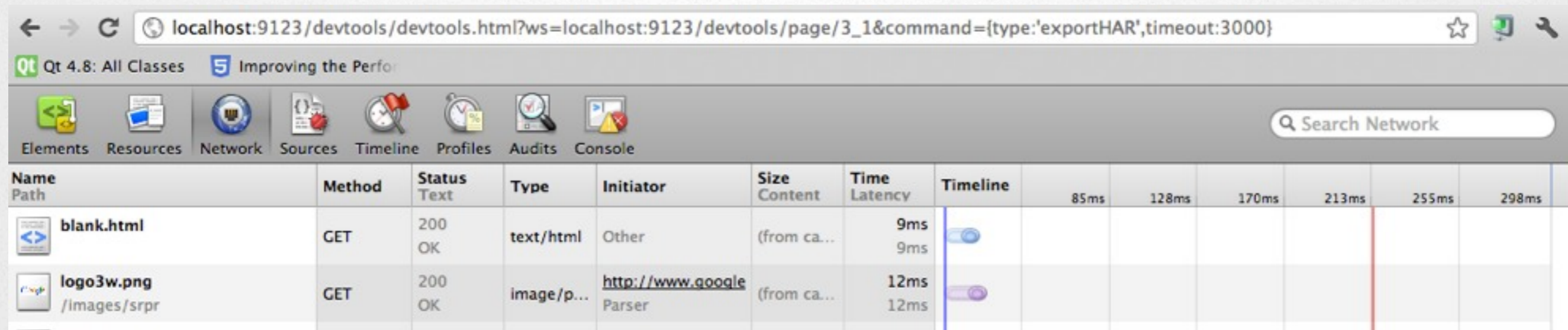
The screenshot shows the Chrome DevTools interface. The address bar contains the URL: `localhost:9123/devtools/devtools.html?ws=localhost:9123/devtools/page/3_1&command={type:'exportHAR',timeout:3000}`. The Network tab is active, displaying a table of network requests. The table has columns for Name/Path, Method, Status/Text, Type, Initiator, Size/Content, Time/Latency, and Timeline. Two requests are visible: `blank.html` and `logo3w.png`.

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline	85ms	128ms	170ms	213ms	255ms	298ms
 <code>blank.html</code>	GET	200 OK	text/html	Other	(from ca...)	9ms 9ms							
 <code>logo3w.png</code> <code>/images/srpr</code>	GET	200 OK	image/p...	<a href="http://www.google.com">http://www.google</a> Parser	(from ca...)	12ms 12ms							







# Trigger Hack

`http://localhost:9123/devtools/devtools.html?ws=localhost:9123/devtools/page/3_1&command={type:'exportHAR',timeout:3000}`



The screenshot shows a web browser window with the address bar containing the URL: `localhost:9123/devtools/devtools.html?ws=localhost:9123/devtools/page/3_1&command={type:'exportHAR',timeout:3000}`. The browser's developer tools are open, and the Network tab is selected. The network tab displays a list of requests:

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline	85ms	128ms	170ms	213ms	255ms	298ms
 blank.html	GET	200 OK	text/html	Other	(from ca...)	9ms 9ms							
 logo3w.png /images/srpr	GET	200 OK	image/p...	<a href="http://www.google.com">http://www.google</a> Parser	(from ca...)	12ms 12ms							

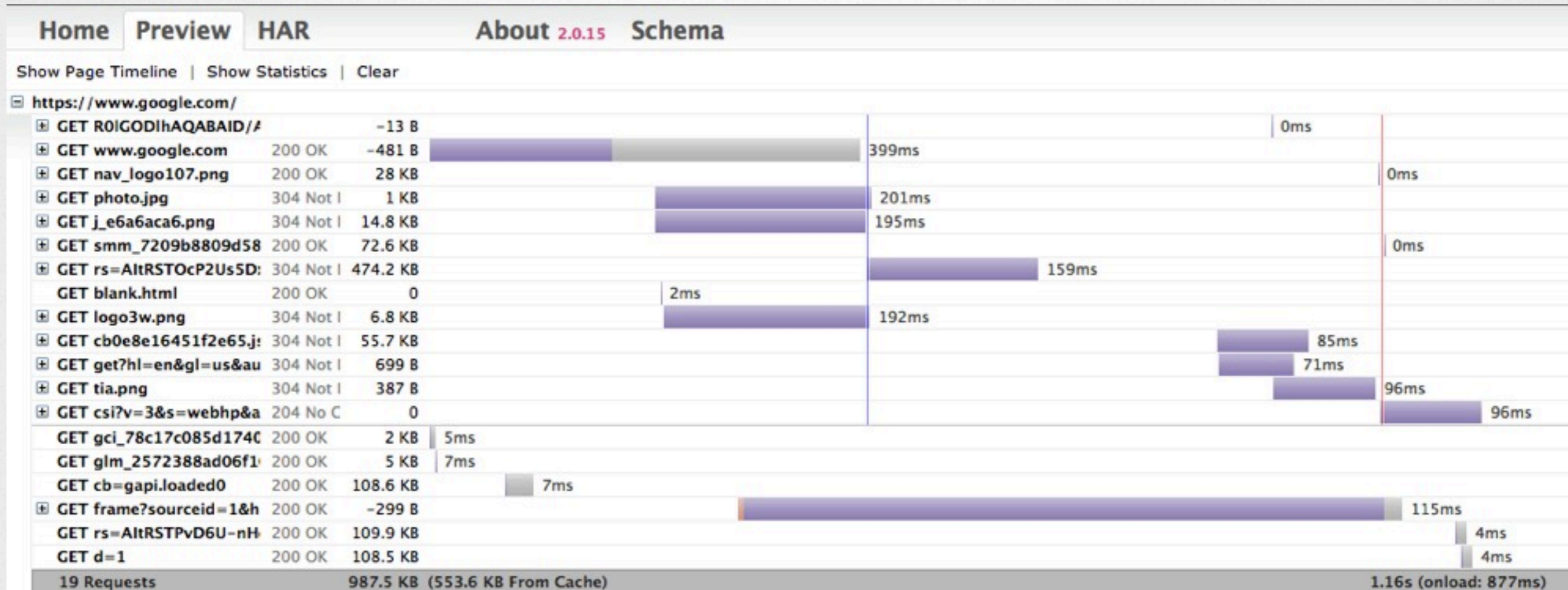


# HAR Output

```
{ "log": { "version": "1.2", "creator":  
  { "name": "WebInspector", "version": "536.5" }, "pages": [], "entries":  
  [ { "startedDateTime": "2012-06-19T18:02:16.852Z", "time": 8, "request":  
    { "method": "GET", "url": "http://www.google.com/  
blank.html", "httpVersion": "HTTP/1.1", "headers": [], "queryString":  
    [], "cookies": [], "headersSize": 47, "bodySize": 0 }, "response":  
    { "status": 200, "statusText": "OK", "httpVersion": "HTTP/1.1", "headers":  
    [], "cookies": [], "content": { "size": 0, "mimeType": "text/  
html" }, "redirectURL": "", "headersSize": 17, "bodySize": 0 }, "cache":  
    {}, "timings": { "blocked":  
0, "dns": -1, "connect": -1, "send": -1, "wait": -1, "receive": 0, "ssl": -1 } } ],
```



# HAR Output



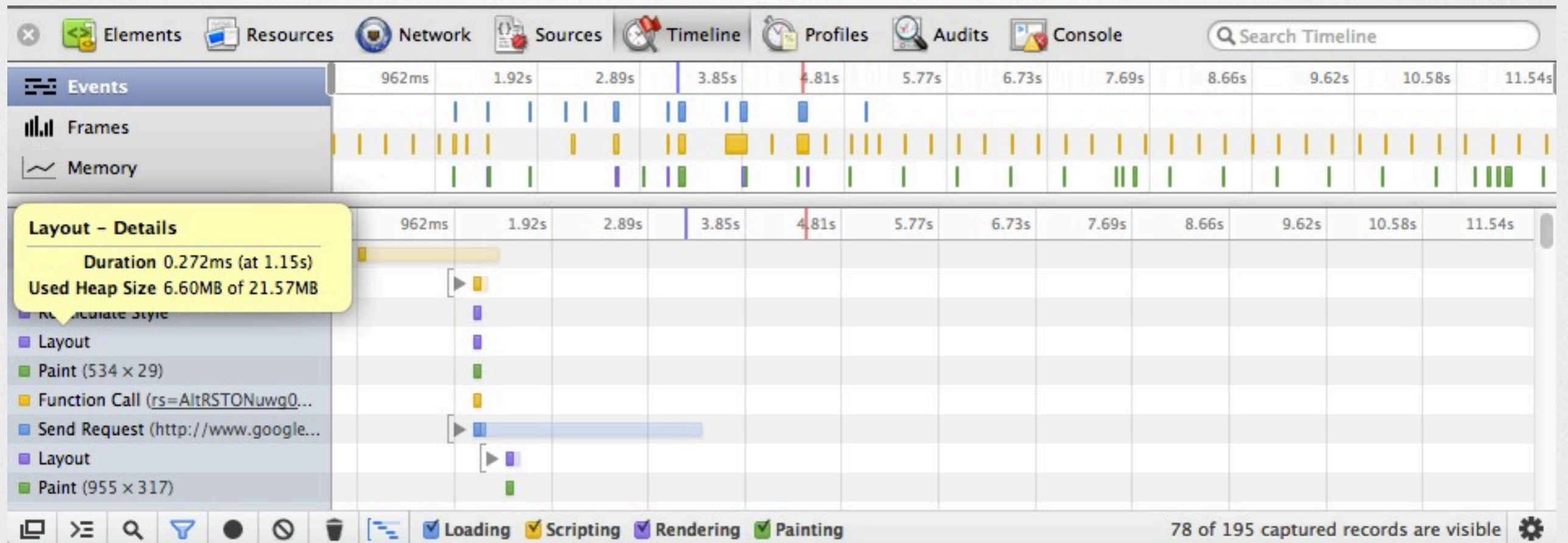
<http://www.softwareishard.com/har/viewer/>



# Demo



# Export Timeline Data



# Add in Hacks

*inspector.js*

*exportTimeline cmd*

```
var exportTimeline = function(cmd) {
  setTimeout(function() {
    // Start the timeline profiler.
    WebInspector.panels.timeline._toggleTimelineButtonClicked();
    // Force a reload of the page to grab latest timeline stats.
    PageAgent.reload(true /* bypass cache */);

    setTimeout(function() {
      // Stop the timeline profiler.
      WebInspector.panels.timeline._toggleTimelineButtonClicked();

      var data = WebInspector.panels.timeline._model.getExportData();
      saveData(cmd.filename || (WebInspector.inspectedPageDomain + '-
timeline.json'), data);
    }, cmd.timeout);
  }, 1000);
};

case 'exportTimeline':
  exportTimeline(command);
  break;
```





# Add in Hacks

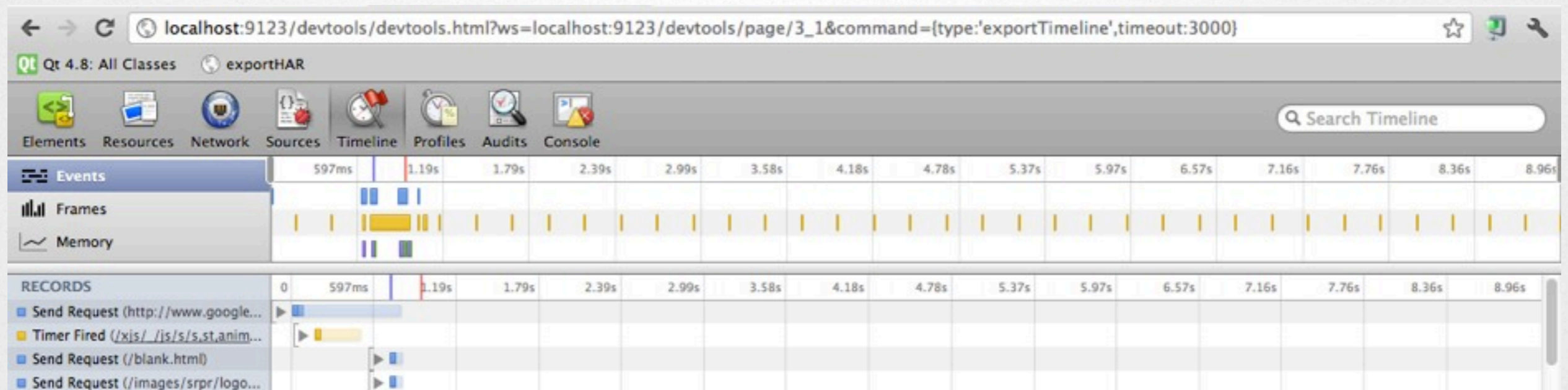
*TimelineModel.js*

*export data fn*

```
getExportData: function() {  
  var records = [];  
  for (var i = 0; i < this._records.length; i++) {  
    records.push(this._records[i]);  
  }  
  return records;  
}
```

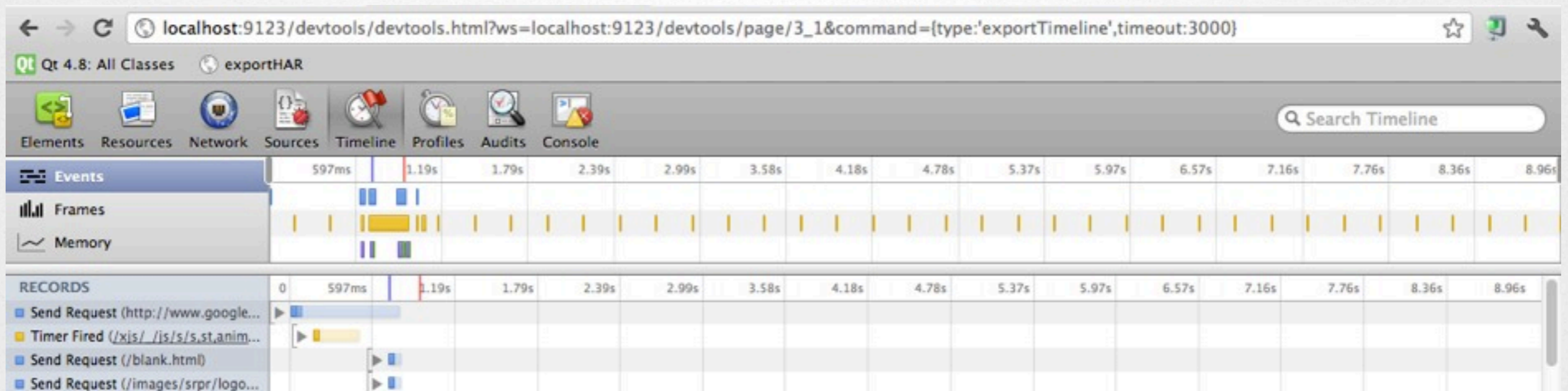


# Trigger Hack



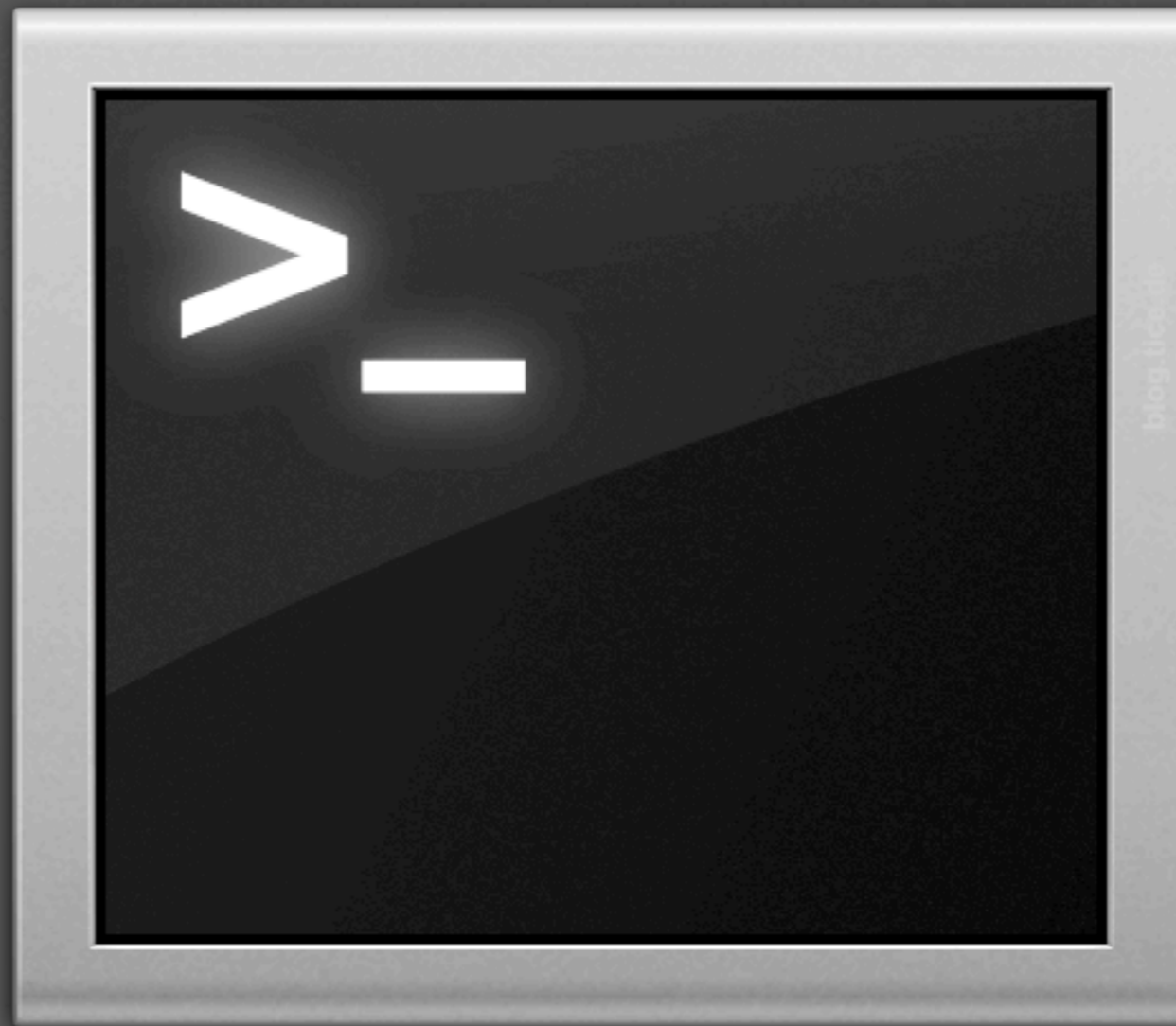
# Trigger Hack

[http://localhost:9123/devtools/devtools.html?ws=localhost:9123/devtools/page/3\\_1&command={type:'exportTimeline',timeout:3000}](http://localhost:9123/devtools/devtools.html?ws=localhost:9123/devtools/page/3_1&command={type:'exportTimeline',timeout:3000})



# Demo





# Command Line WebKit



# Demo



**How'd you do that!?**





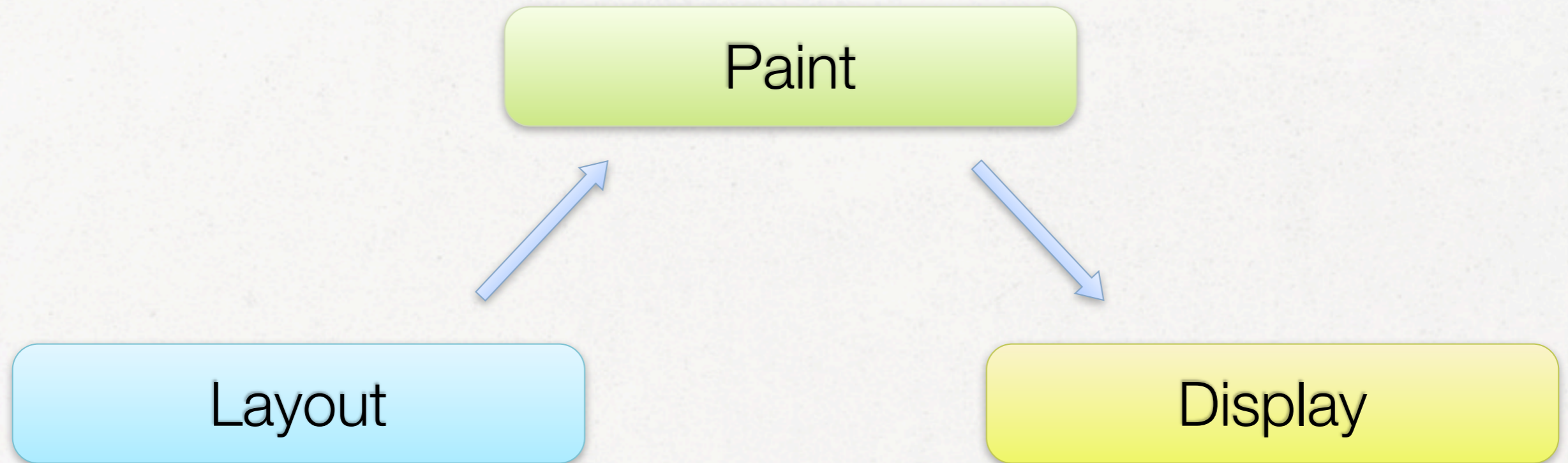
# Headless WebKit



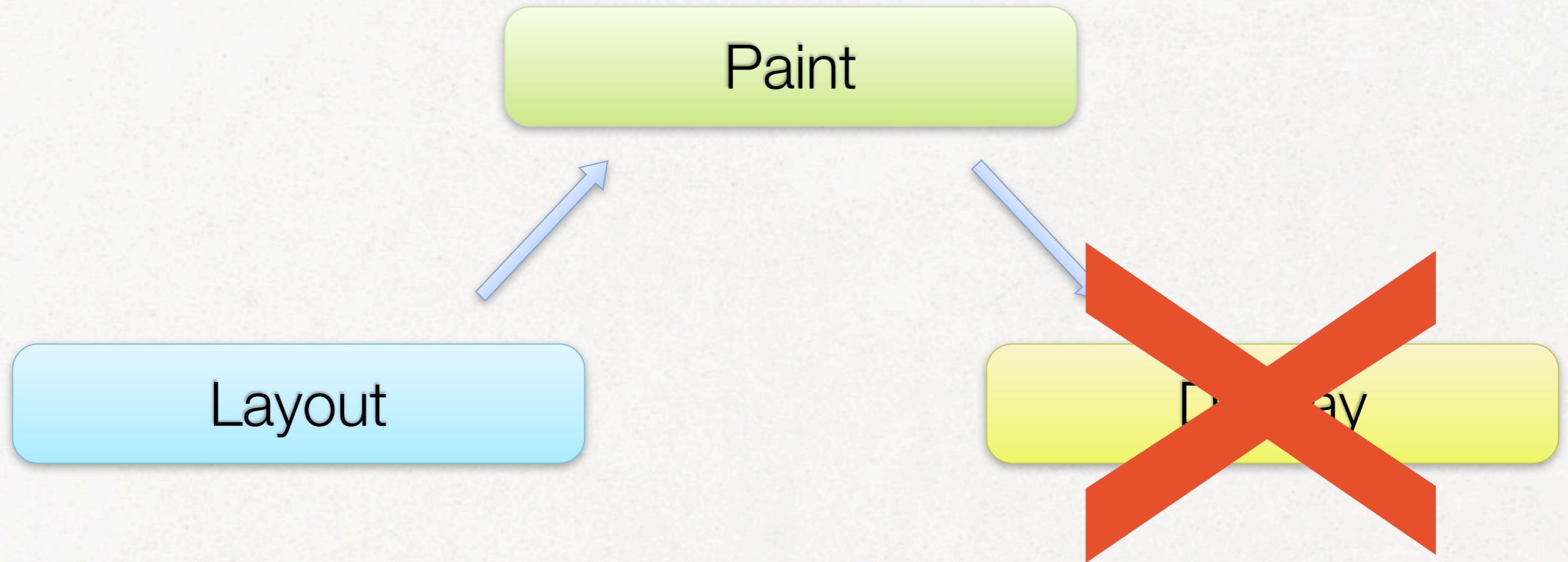
**“Headless”**



# Normal Browser



# Headless



What's in it for me?



# What's in it for me?

Automated “smoke” tests

Continuous integration

Pixel-perfect regression tests

Web scraping

...



# PhantomJS

Headless WebKit

Based on QtWebKit port

Full Web Stack

JS, DOM, CSS3, Canvas, SVG, etc.

Command Line Interface

JavaScript APIs

License: BSD / Open Source

<http://www.phantomjs.org>



**Demo**



# Wrap Up





# Hacking WebKit

## Automation

- Page scraping

- Behavior/input emulation

- Web Inspector hijacking

## Quality Assurance / Continuous Integration

- Smoke tests

- Visual/pixel regression tests

- Unit tests

- Performance or Load regression tests





**Jarred Nicholls**

@jarrednicholls

**Thanks!**

