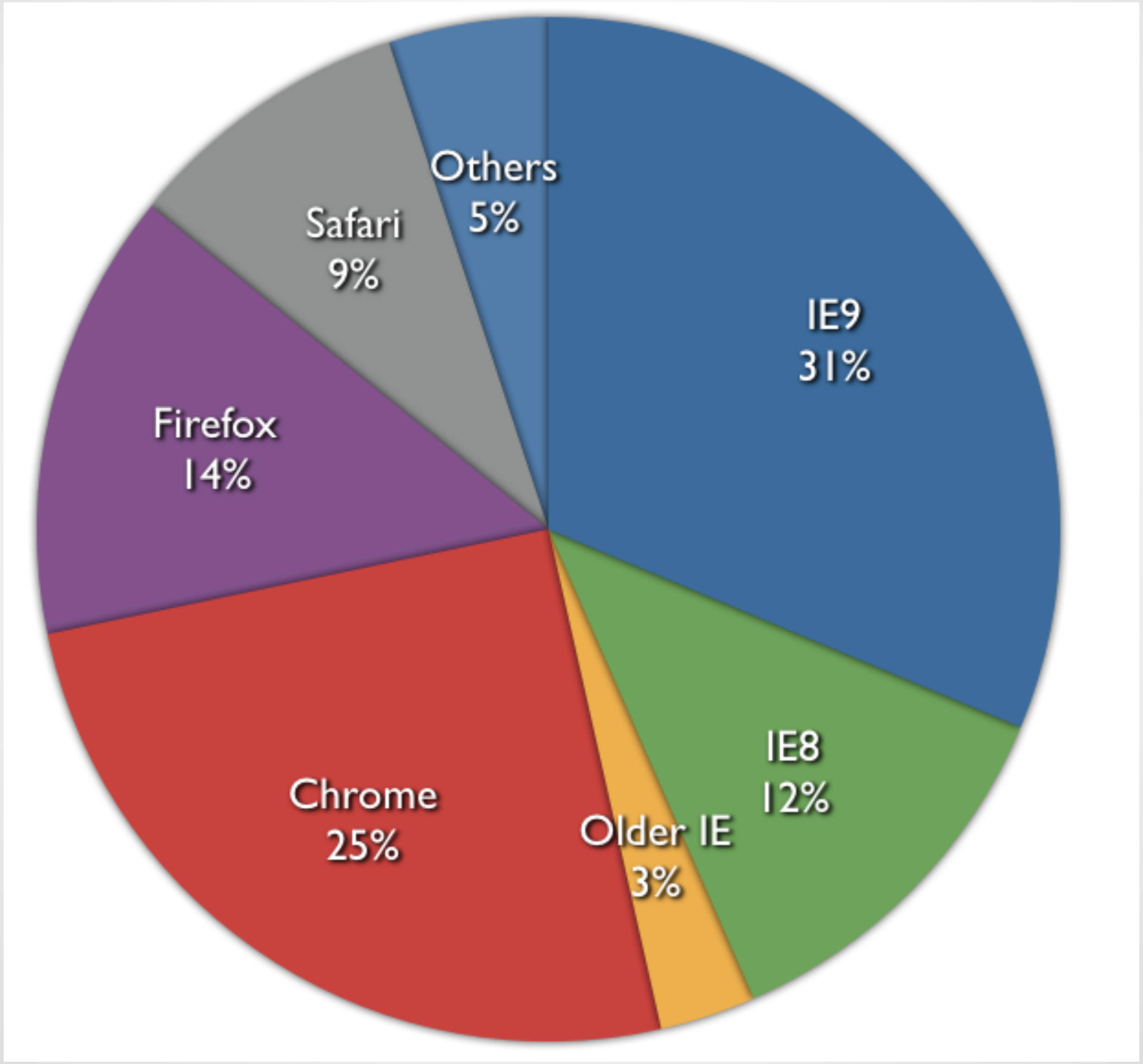
Chrome Dev Tools

@matenadasdi

Ustream

Ustream browser stats



Why do we need a dev tool?

- Javascript is an interpreted language
- HTML, CSS debugging and performance optimization is impossible
- Logging, debugging network requests is essential
- Source and the final output could be totally different

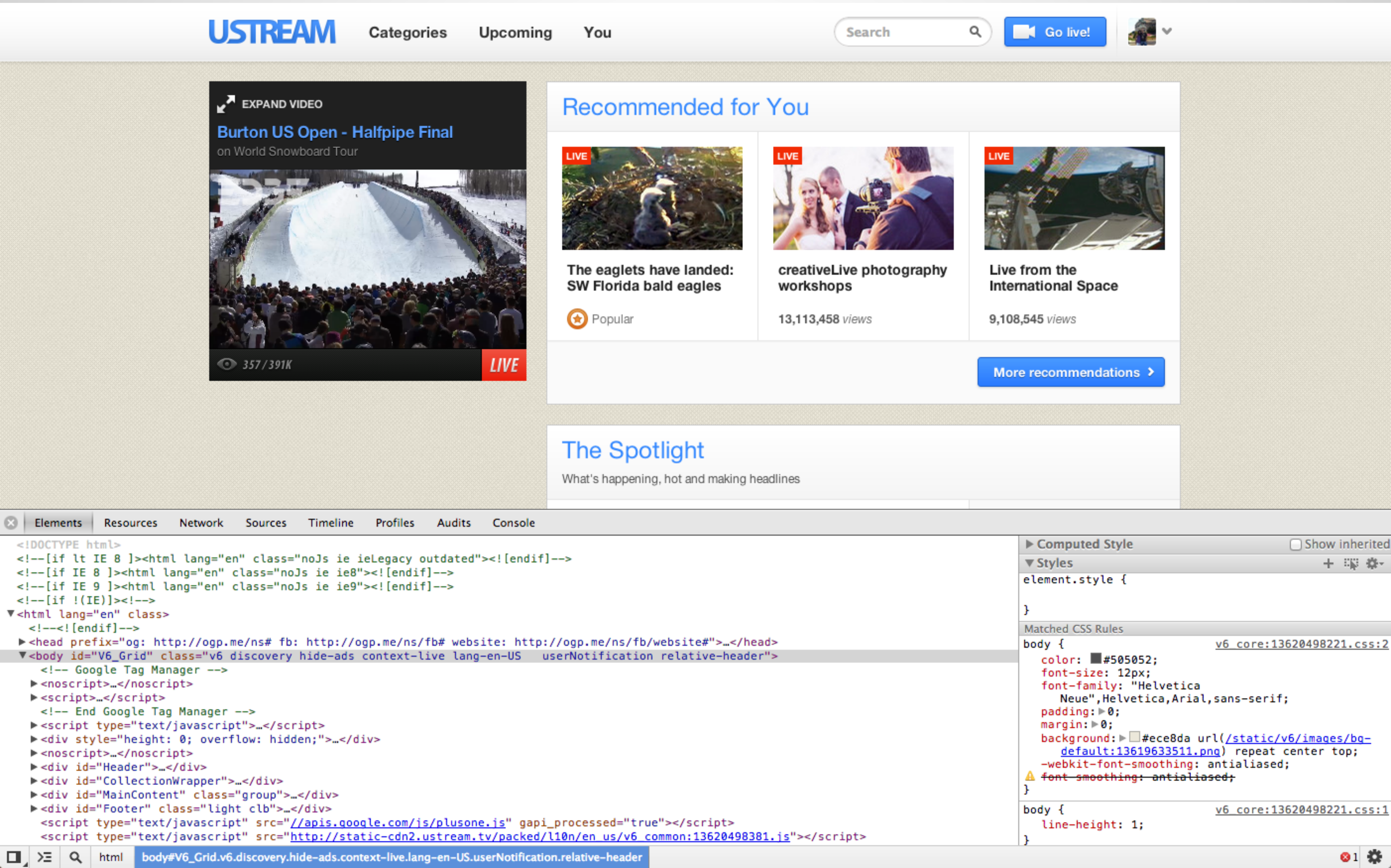
Why Chrome?

- Native and really fast support
- Canary build implements new features in short intervals
- Frame / Memory Profiling tools
- Action history
- Google is leading in new technologies
- Firefox native dev tools could be a rival

Red - Blue - Yellow

- Chrome channels
  - Stable *(Releases in every 6 weeks)*
  - Beta *(1 month before stable, weekly releases)*
  - Dev *(twice weekly)*
  - Canary *(daily)*
- Chromium

DevTools UI Basics



```
<!DOCTYPE html>
<!--[if lt IE 8 ]><html lang="en" class="noJs ie ieLegacy outdated"><![endif]-->
<!--[if IE 8 ]><html lang="en" class="noJs ie ie8"><![endif]-->
<!--[if IE 9 ]><html lang="en" class="noJs ie ie9"><![endif]-->
<!--[if !(IE)]><!-->
▼<html lang="en" class>
  <!--<![endif]-->
 ▶<head prefix="og: http://ogp.me/ns# fb: http://ogp.me/ns/fb# website: http://ogp.me/ns/fb/website#">…</head>
 ▼<body id="V6_Grid" class="v6 discovery hide-ads context-live lang-en-US   userNotification relative-header">
    <!-- Google Tag Manager -->
   ▶<noscript>…</noscript>
   ▶<script>…</script>
    <!-- End Google Tag Manager -->
   ▶<script type="text/javascript">…</script>
   ▶<div style="height: 0; overflow: hidden;">…</div>
   ▶<noscript>…</noscript>
   ▶<div id="Header">…</div>
   ▶<div id="CollectionWrapper">…</div>
   ▶<div id="MainContent" class="group">…</div>
   ▶<div id="Footer" class="light clb">…</div>
    <script type="text/javascript" src="//apis.google.com/js/plusone.js" gapi_processed="true"></script>
    <script type="text/javascript" src="http://static-cdn2.ustream.tv/packed/l10n/en_us/v6_common:13620498381.js"></script>
```

```
element.style {

}
Matched CSS Rules
body {                                      v6_core:13620498221.css:2
   color: ■#505052;
   font-size: 12px;
   font-family: "Helvetica
      Neue",Helvetica,Arial,sans-serif;
   padding:▶0;
   margin:▶0;
   background:▶□#ece8da url(/static/v6/images/bg-
      default:13619633511.png) repeat center top;
   -webkit-font-smoothing: antialiased;
⚠ ~~font-smoothing: antialiased;~~
}
body {                                      v6_core:13620498221.css:1
   line-height: 1;
}
```

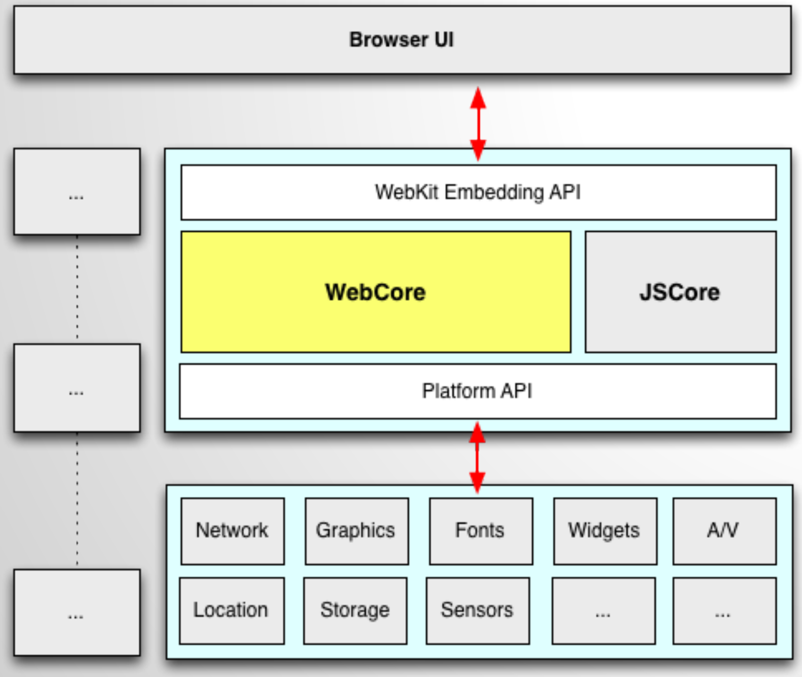
What is Webkit?

Browser components:

- Parsing (HTML, XML, CSS, Javascript)
- Layout
- Text and graphics rendering
- Image Decoding
- GPU interaction
- Network access
- Hardware acceleration

Webkit - different ports

	Chrome (OS X)	Safari (OS X)	QtWebKit	Android Browser	Chrome for iOS
Rendering	Skia	CoreGraphics	QtGui	Android stack/Skia	CoreGraphics
Networking	Chromium network stack	CFNetwork	QtNetwork	Fork of Chromium's network stack	Chromium stack
Fonts	CoreText via Skia	CoreText	Qt internals	Android stack	CoreText
JavaScript	V8	JavaScriptCore	JSC (V8 is used elsewhere in Qt)	V8	JavaScriptCore (without JITting) *

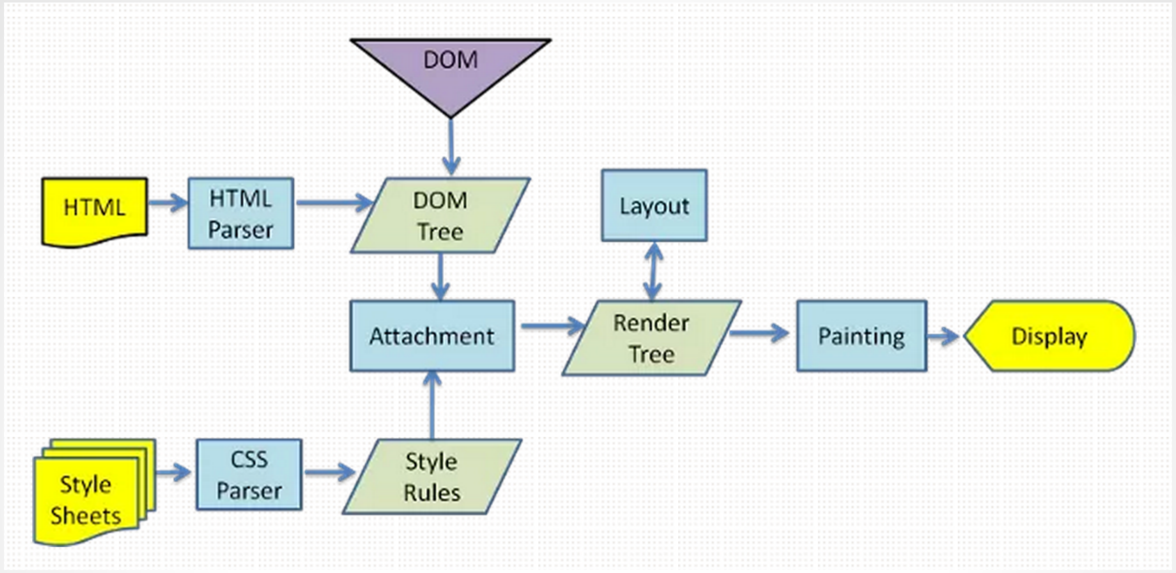


Common webkit features

- DOM, CSSOM
- CSS Parsing
- HTML parsing, DOM construction
- Layout, positioning
- DevTools UI (except Safari)
- Features like: File API, CSS Transform Math, web Audio API, localStorage

source: Paul Irish

Webkit - From source to DOM tree



- DOM Tree
- Render Object Tree
- Render Layer Tree (forces to get new: transparency, relative, absolute, transform, filter, root element of the page, etc.)
- GraphicsLayer Tree (HW acc path)
- Continuous output

Elements panel - DOM Manipulation

- Keyboard action support
- History
- Magnifier
- Parent-child horizontal representation

Elements panel - Styles

- Computed styles exactly as the renderer sees
- Force new rules as inline styles
- Force states
- Matched CSS rules
- Color picker with different formats

Resources panel - Browser storages and their specialties

- WebSQL (SQLite, deprecated, needs permission to break the limit)
- IndexedDB
  - Asnyc, transactional, noSQL
  - FF asks for permission over ~50MB
  - Chrome lets (Disk space / 2 * 0.20) space for offline storage
- Application cache: Cache / Network / fallback (~5 MB / origin)
- LocalStorage - **Permanent** (2.5MB Chrome - 5 MB FF - 10MB IE / Origin)
- SessionStorage - Session only (**System memory**)
- Cookies - Not modifiable (~4KB / origin)

Network panel

- Timeline waterfall for visualization
- Check parallelism
- HAR export (HTTP Archive)
- Cache efficiency

Don't be shy to use breakpoints!

- For debugging Javascript execution
  - Watch expressions
  - Call stack
  - Scope variables
- Event listener breakpoints
- DOM Breakpoints
- XHR breakpoints

Timeline panel - Reflow & Repaint

Reflow:

Parts of the render tree needs to be revalidated and node dimensions should be recalculated.

Repaint:

Some part of the screen needs to be updated because of a reflow, node geometric change or style change.

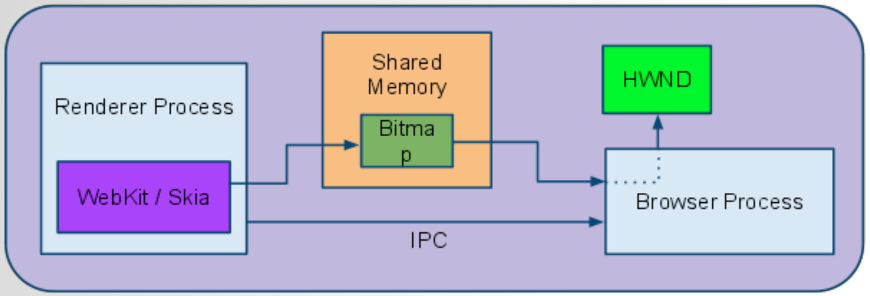
Can be tested with 'show paint rectangles' feature.

Timeline panel - Reflow & Repaint optimization

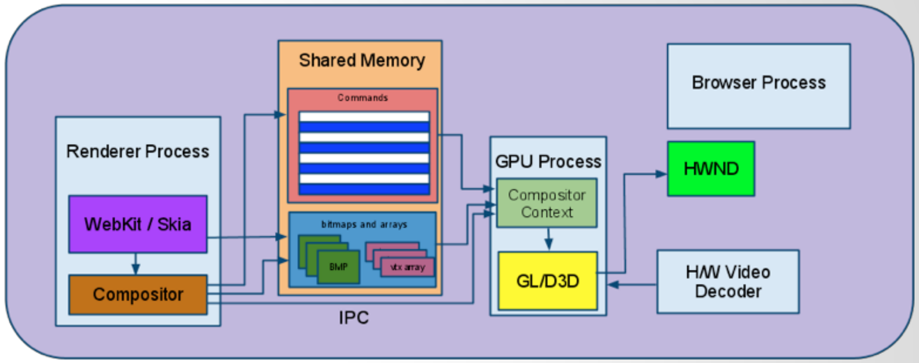
- Display: none and visibility: hidden is a good example for reflow & repaint
- Do not change styles one-by-one
- Detach DOM parts before huge manipulation
- Querying the DOM could be painful too
  - offsetTop, offsetLeft, offsetHeight, offsetWidth
  - scrollTop, scrollLeft, scrollHeight, scrollWidth
  - clientTop, clientLeft, clientHeight, clientWidth
  - getComputedStyle()
- Do not forget the browser queue of changes

GPU Accelerated rendering

Software path:



Hardware path:



Differences:

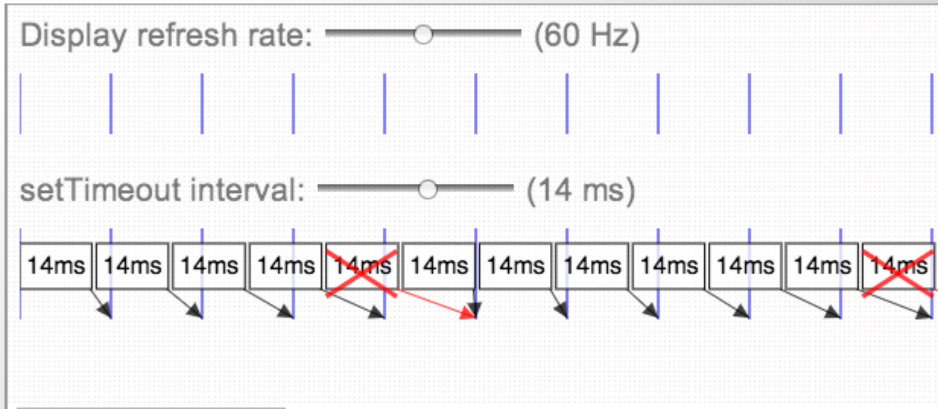
- RenderLayers could paint to an own backing surface called Compositing Layer
- Each compositing layer has its own GraphicContext
- New tree added: **GraphicsLayer Tree**
- **DOM Tree - Render Object Tree - Render layer Tree - GraphicsLayer Tree**
- **H/W acc. Force examples:** video tag, canvas 3D context, transform transition, CSS filter, etc

GPU Accelerated rendering - scrolling

- Damage Rect
- Repaint only intersected RenderLayers
- GPU 256x256px tiles
- Paint and upload the damaged tiles
- Tile-prepainting
- Different thread for compositing

Timeline panel - Frames

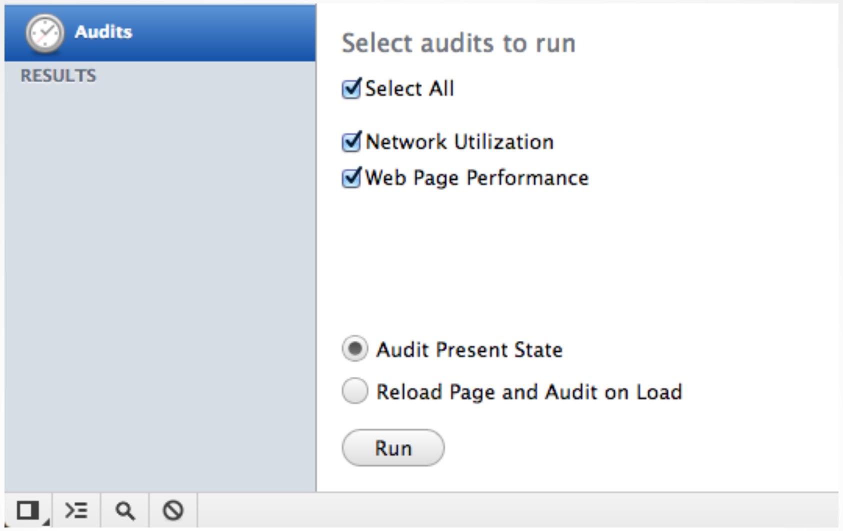
- Tearing
- vSync - generating new frames only between screen refreshes
- 60 HZ = We have got 16 ms only! (60 Hz = 1 / ~0,016)
- setInterval, setTimeout fails because of javascript timers, and different frame rates
- requestAnimationFrame



Memory profiles

- GC Cycles
- Heap snapshot
- Comparison
- Containment

Audits & Console

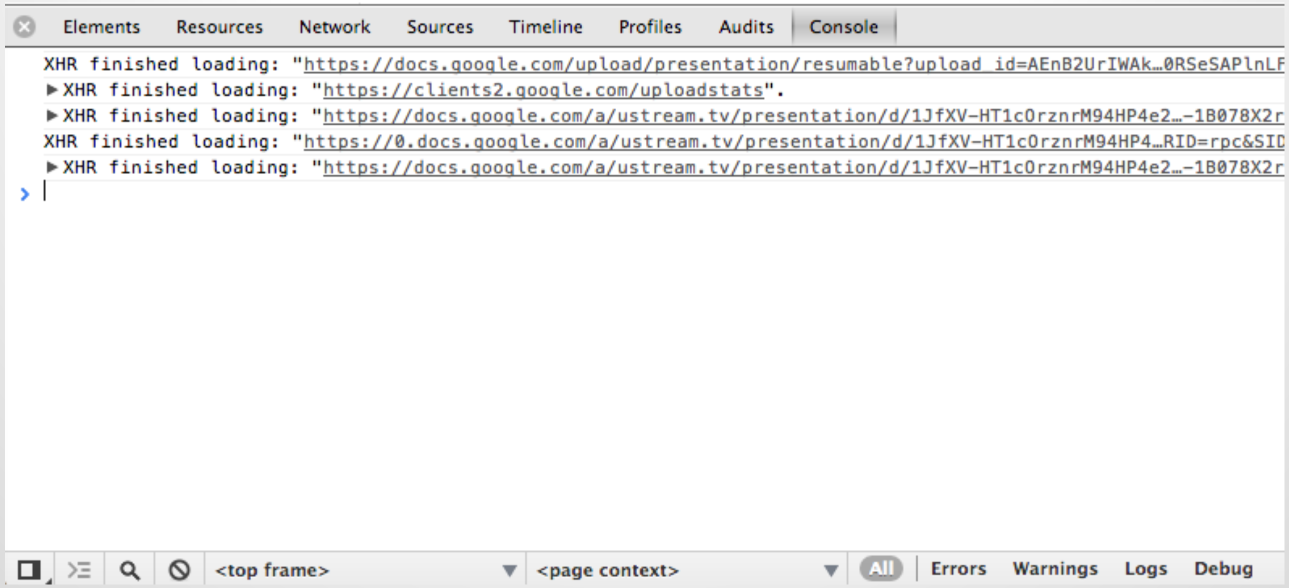


Select audits to run

[✓] Select All

[✓] Network Utilization
[✓] Web Page Performance

[✓ Audit Present State]
Reload Page and Audit on Load



```
XHR finished loading: "https://docs.google.com/upload/presentation/resumable?upload_id=AEnB2UrIWAk…0RSeSAPlnLF
 ▶XHR finished loading: "https://clients2.google.com/uploadstats".
 ▶XHR finished loading: "https://docs.google.com/a/ustream.tv/presentation/d/1JfXV-HT1cOrznrM94HP4e2…-1B078X2r
XHR finished loading: "https://0.docs.google.com/a/ustream.tv/presentation/d/1JfXV-HT1cOrznrM94HP4…RID=rpc&SID
 ▶XHR finished loading: "https://docs.google.com/a/ustream.tv/presentation/d/1JfXV-HT1cOrznrM94HP4e2…-1B078X2r
>
```

Settings panel

- chrome://flags
- Overrides, emulations
- Rulers
- Disable cache / JS
- Show Paint rectangles, Composited layer borders, Continuous repainting

Thanks!

Mate Nadasdi