Insomniac’s Web Tools (a postmortem)

Andreas Fredriksson
Lead Engine Programmer
Insomniac Games
Hi, I’m Andreas
Hi, I’m Andreas

- I lead the tools & infrastructure team at Insomniac
Hi, I’m Andreas

- I lead the tools & infrastructure team at Insomniac
- “So you guys have web tools?”
  - #1 conversation opener
Our web tools exodus
Our web tools exodus

- Disclosure: We have abandoned web tools development
Our web tools exodus

- Disclosure: We have abandoned web tools development
- This talk is about what was good
Our web tools exodus

- Disclosure: We have abandoned web tools development
- This talk is about what was good
- …And what wasn’t so good
The 2010 Web Tools Vision
The 2010 Web Tools Vision

- Overhauling tools+engine for cross-platform tech reboot
The 2010 Web Tools Vision

- Overhauling tools+engine for cross-platform tech reboot
- Games having ever-bigger data sets
  - New focus on visualization and mining
The 2010 Web Tools Vision

- Overhauling tools+engine for cross-platform tech reboot
- Games having ever-bigger data sets
  - New focus on visualization and mining
- More UI innovation in the web space!
  - Compared to desktop space
  - Also true of mobile space
The 2010 Web Tools Vision

- Overhauling tools+engine for cross-platform tech reboot
- Games having ever-bigger data sets
  - New focus on visualization and mining
- More UI innovation in the web space!
  - Compared to desktop space
  - Also true of mobile space
- Strong team drive to try new things
Florida: Search interest in hurricane preparedness

- Evacuation
- Emergency
- Hurricane update

Oct 7, 2016 at 9:00 AM
- Evacuation: 10
- Emergency: 23
- Hurricane update: 41
Click on a timeline year to skip to that point.

Dec 2001
12. BRAZIL
EXPORTS: -$1,099,600,000

-$4,118,975,000,000
* Cumulative trade imbalance since 2001
New in 2.0 New chart axis types

Plot complex, sparse datasets on date time, logarithmic or even entirely custom scales with ease.

New in 2.0 Animate everything!

Out of the box stunning transitions when changing data, updating colours and adding datasets.
2010 vision, continued

- Hire awesome web people!

2010 vision: “Everyone can make a web page”

- Lower the barrier to entry!
- Empower the team to roll out simpler stuff whenever
2010 vision: “There’s tons of off the shelf web stuff”
2010 vision: “There’s tons of off the shelf web stuff”

- Lots, lots, lots out there
2010 vision: “There’s tons of off the shelf web stuff”

- Lots, lots, lots out there
- Open ecosystem
2010 vision: “There’s tons of off the shelf web stuff”

- Lots, lots, lots out there
- Open ecosystem
- Fewer wheels to reinvent
2010 vision: Enforce one-way data pipeline

Tools

Engine
2010 vision: Enforce one-way data pipeline

Source Data
Tools

Built Data
Engine
2010 vision: Enforce one-way data pipeline

Source Data
Tools

Builders

Built Data
Engine
2010 vision: Web browser = Data Sandbox
2010 vision: Web browser = Data Sandbox

- Hard to call browser logic from engine
2010 vision: Web browser = Data Sandbox

- Hard to call browser logic from engine
- Hard to call engine logic from browser
2010 vision: Web browser = Data Sandbox

- Hard to call browser logic from engine
- Hard to call engine logic from browser
- Much easier to do the right thing instead (write a builder)
2010 grand vision

- Zero install tools
  - “Have a browser? You’re good to go.”
So that was the vision..
So that was the vision..

- ...and the team started building stuff
So that was the vision..

- …and the team started building stuff
- Core Team = a bunch of C++ experts
  - With almost no web experience at this point
So that was the vision..

- ...and the team started building stuff
- Core Team = a bunch of C++ experts
  - With almost no web experience at this point
- We had 3-5 people working on web tools full time for 5 years
Early days, early decisions
Early days, early decisions

- Single browser: Chrome
  - Avoid the web compatibility/standards problem
  - Leverage Chrome team smarts - let them worry about browser!
Early days, early decisions

• Single browser: Chrome
  • Avoid the web compatibility/standards problem
  • Leverage Chrome team smarts - let them worry about browser!

• 3D native engine view?
  • Jam a native child window into Chrome!
  • Tiny amount of glue code in a plugin needed
Early days, early decisions

- Single browser: Chrome
  - Avoid the web compatibility/standards problem
  - Leverage Chrome team smarts - let them worry about browser!
- 3D native engine view?
  - Jam a native child window into Chrome!
  - Tiny amount of glue code in a plugin needed
- Where do web pages come from?
  - A server!
NTFS
Source Asset Management

Mongo DB

LunaServer

NTFS
Source Asset Management

Mongo DB

LunaServer

NTFS

Web Tools

Chrome Tool

Tools

3D View
Vision meets reality: Early tradeoffs
Vision meets reality: Early tradeoffs

- Desktop look and feel
  - Early UX tests revealed dislike of “odd stuff”
  - Traditional concepts like menu bars strongly desired
Vision meets reality: Early tradeoffs

- Desktop look and feel
  - Early UX tests revealed dislike of “odd stuff”
  - Traditional concepts like menu bars strongly desired
- Has to work *together* with Maya, Photoshop, …
  - It’s not a web app on a deserted island, it exists side by side
Vision meets reality: Early tradeoffs

- Desktop look and feel
  - Early UX tests revealed dislike of “odd stuff”
  - Traditional concepts like menu bars strongly desired
- Has to work *together* with Maya, Photoshop, …
  - It’s not a web app on a deserted island, it exists side by side
- “We can’t do that because it’s a web app” not a valid excuse
Our web tools: A gallery

- A *complete* AAA tool suite in the browser
- Not trying to impress you
  - Need to understand the scope to get the rest of the talk
3D View
SceneEditor.exe attached to Chrome
3D View
SceneEditor.exe attached to Chrome
SceneEditor.exe attached to Chrome
LunaServer IP 10.10.106.25
Build Output\Built px-130.1.0 Release

Role: Review
Language: English

Level: levels/px_area_4/px_area_4.level

Checkpoint: CHK_SEG1_EAST_CLIFF_SIDE_PATH

Flags: -frontend

Platform: Windows
Build Config: Release

Install Simulation: Off

BugIt:

Commandline: -warn review -http 10.10.106.25 -frontend -level levels/px_area_4/px_area_4.level -checkpoint CHK_SEG1_EAST_CLIFF_SIDE_PATH

Launch Game  Screenshot  Copy Cmd Line
That’s a lot of stuff!
That’s a lot of stuff!

• We definitely answered the “is it even possible” question
That’s a lot of stuff!

- We definitely answered the “is it even possible” question
- 340,000 Javascript LOC, ~1,000 files
  - Excluding generated and 3rd party code
That’s a lot of stuff!

- We definitely answered the “is it even possible” question
- 340,000 Javascript LOC, ~1,000 files
  - Excluding generated and 3rd party code
- 500,000 C++ LOC of server infrastructure & builders
What went right
What went right
What went right

- Debugger and profiler on every machine
  - Chrome dev tools are decent
What went right

- Debugger and profiler on every machine
  - Chrome dev tools are decent
- Achieved clean data separation
What went right

- Debugger and profiler on every machine
  - Chrome dev tools are decent
- Achieved clean data separation
- Centralized undo/redo
What went right

- Debugger and profiler on every machine
  - Chrome dev tools are decent
- Achieved clean data separation
- Centralized undo/redo
- Symbolic data manipulation
Right: Central undo/redo
Right: Central undo/redo

Proposed Change: Delta JSON
Right: Central undo/redo

- Server
- Change Log
- Proposed Change: Delta JSON
- Tool
Right: Central undo/redo

Diagram:
- Server
- Change Log
- Change: Delta JSON
- Proposed Change: Delta JSON
- Tool
Right: Central undo/redo

Change: Delta JSON

Proposed Change: Delta JSON

Server

Change Log

Tool

Tool
Right: Central undo/redo

Undo/Redo!

Server
Right: Central undo/redo
Right: Central undo/redo

Undo/Redo!

Server

Change Log

Undo Change: Delta JSON

Tool

Undo Change: Delta JSON

Tool
Right: Symbolic Data Manipulation
Right: Symbolic Data Manipulation

- Need to be able to edit any piece of game data in web tools
Right: Symbolic Data Manipulation

- Need to be able to edit any piece of game data in web tools
- Can’t link to game (C++) code from web browser
Right: Symbolic Data Manipulation

- Need to be able to edit any piece of game data in web tools
- Can’t link to game (C++) code from web browser

```c
struct MySpecialObject {
    int32 Hitpoints; // Default = 100
    float Shininess; // Default = 3.5
    // ...
}
```
Right: Symbolic Data Manipulation

- Solution: Edit all data formats using meta-data
  - DDL compiler outputs both C++ code and metadata for tools
  - Use symbolic meta-data in tools to edit assets
  - Useful in native tools as well, reduces coupling

```
MySpecialObject metadata:
    Name="Hitpoints" Type=i32 Default=100
    Name="Shininess" Type=f32 Default=3.5
    // ...
```
So, what went wrong?
So, what went wrong?

{} yourselves
Wrong: Early web mindset
Wrong: Early web mindset

- “It’s just a web page”
Wrong: Early web mindset

- “It’s just a web page”
- In rush to try to get something out the door
Wrong: Early web mindset

- “It’s just a web page”
- In rush to try to get something out the door
- Poor engineering style early on
  - No tests
  - Global variables aplenty
  - Learning as we were going
Wrong: Javascript
Wrong: Javascript

- So much pain – a toy language that is still growing up
Wrong: Javascript

- So much pain – a toy language that is still growing up
- undefined is not a function
Wrong: Javascript

- So much pain – a toy language that is still growing up
- undefined is not a function
- Silently breaking other parts of the codebase
  - Renaming a function = super high risk
  - Adding a parameter? Good luck!
  - Relying on grep or IDE searching to fix up references often fails
Wrong: Javascript

- So much pain – a toy language that is still growing up
- undefined is not a function
- Silently breaking other parts of the codebase
  - Renaming a function = super high risk
  - Adding a parameter? Good luck!
  - Relying on grep or IDE searching to fix up references often fails
- Add quick fixes on top and you’ve got a nightmare
Wrong: No heavy lifting
Wrong: No heavy lifting

- JS single-threaded (web workers are a joke)
Wrong: No heavy lifting

- JS single-threaded (web workers are a joke)
- Garbage collection
Wrong: No heavy lifting

- JS single-threaded (web workers are a joke)
- Garbage collection
- Impractical to work with full game dataset
  - Sunset Overdrive's streaming optimization, for example
Wrong: LunaServer became a dumping ground
Wrong: LunaServer became a dumping ground

- Anything you can’t do in the browser goes in the server.
Wrong: LunaServer became a dumping ground

- Anything you can’t do in the browser goes in the server..
- LunaServer quickly grew bigger and bigger
  - Random program launching
  - File system interfaces
  - Desktop interop services
  - Perforce interfaces
Wrong: LunaServer became a dumping ground

- Anything you can’t do in the browser goes in the server..
- LunaServer quickly grew bigger and bigger
  - Random program launching
  - File system interfaces
  - Desktop interop services
  - Perforce interfaces
- Lots of technical debt unless planned for carefully!
Wrong: LunaServer became a dumping ground

- Anything you can’t do in the browser goes in the server..
- LunaServer quickly grew bigger and bigger
  - Random program launching
  - File system interfaces
  - Desktop interop services
  - Perforce interfaces
- Lots of technical debt unless planned for carefully!
- Is this still used? Grep all the code and hope for the best!
Mixed: Javascript on V8
Mixed: Javascript on V8

- Optimizing for performance can be non-intuitive
  - Programming several layers away from actual compiler
Mixed: Javascript on V8

- Optimizing for performance can be non-intuitive
  - Programming several layers away from actual compiler
- V8 optimization best practices change over time
  - Keeping up on this for 300k LOC is a hard problem
Mixed: Javascript on V8

- Optimizing for performance can be non-intuitive
  - Programming several layers away from actual compiler
- V8 optimization best practices change over time
  - Keeping up on this for 300k LOC is a hard problem
- Often conflicts between idiomatic vs fast
Mixed: Javascript on V8

- Optimizing for performance can be non-intuitive
  - Programming several layers away from actual compiler
- V8 optimization best practices change over time
  - Keeping up on this for 300k LOC is a hard problem
- Often conflicts between idiomatic vs fast
- Lack of information and “what good looks like”
  - Many web frameworks have poor performance
Wrong: V8 Stability
Wrong: V8 Stability

- We wanted to use ES6/Harmony features
  - In particular, ‘let’ fixes block scope "terribleness"
Wrong: V8 Stability

- We wanted to use ES6/Harmony features
  - In particular, ‘let’ fixes block scope terribleness
- Great until
  - Browser crashed in production, randomly
  - Debugger crashed trying to debug it
Wrong: V8 Stability

- We wanted to use ES6/Harmony features
  - In particular, ‘let’ fixes block scope terriblyness
- Great until
  - Browser crashed in production, randomly
  - Debugger crashed trying to debug it
- Resorted to taking it all out again
Wrong: Promises

- Lots of things you wait for in JS
  - Because you’re constantly talking to some server
Wrong: Promises

- Lots of things you wait for in JS
  - Because you’re constantly talking to some server
- Event handling this is messy
Wrong: Promises

- Lots of things you wait for in JS
  - Because you’re constantly talking to some server
- Event handling this is messy
- Promises hope to solve this problem
animSetMojo.Source.prototype.loadAnimSetActor = function() {
    if (!this.isConnected()) {
        return Q.resolve();
    }

    return this.onStage1FetchActor()
        .then( this._private.callbacks.onStage2PreLoadAssets )
        .then( this._private.callbacks.onWaitForAssetLoad )
        .then( this._private.callbacks.onStage3CreateActor )
        .then( this._private.callbacks.onWaitForAssetLoad )
        .then( this._private.callbacks.onStage4WatchActor )
        .then( this._private.callbacks.onStage5Focus );
};
animSetMojo.Source.prototype.loadAnimSetActor = function() {
    if ( !this.isConnected() ) {
        return Q.resolve();
    }
    return this.onStage1FetchActor()
        .then( this._private.callbacks.onStage2PreLoadAssets )
        .then( this._private.callbacks.onWaitForAssetLoad )
        .then( this._private.callbacks.onStage3CreateActor )
        .then( this._private.callbacks.onWaitForAssetLoad )
        .then( this._private.callbacks.onStage4WatchActor )
        .then( this._private.callbacks.onStage5Focus );
};
Promises = Now all your crashes look like this

TypeError: Cannot read property 'Path' of null
at vault.js:1614
at _fulfilled (q-0.9.6.js:714)
at self.promiseDispatch.done (q-0.9.6.js:743)
at Promise.promise.promiseDispatch (q-0.9.6.js:680)
at q-0.9.6.js:554
at MessagePort.flush (q-0.9.6.js:108)
msgLog.error @ msgLog.js:168
(anonymous) @ msgLog.js:195
defaultQPromiseErrorHandler @ env.js:928
(anonymous) @ q-0.9.6.js:1553
flush @ q-0.9.6.js:108
Promises = Now all your crashes look like this

TypeError: Cannot read property 'Path' of null
at vault.js:1614
at _fulfilled (q-0.9.6.js:714)
at self.promiseDispatch.done (q-0.9.6.js:743)
at Promise.promise.promiseDispatch (q-0.9.6.js:680)
at q-0.9.6.js:554
at MessagePort.flush (q-0.9.6.js:108)
msgLog.error @ msgLog.js:168
(anonymous) @ msgLog.js:195
defaultQPromiseErrorHandler @ env.js:928
(anonymous) @ q-0.9.6.js:1553
flush @ q-0.9.6.js:108
Promises = Now all your crashes look like this

```javascript
TypeError: Cannot read property 'Path' of null
at vault.js:1614
at _fulfilled (q-0.9.6.js:714)
at self.promiseDispatch.done (q-0.9.6.js:743)
at Promise.promise.promiseDispatch (q-0.9.6.js:680)
at q-0.9.6.js:554
at MessagePort.flush (q-0.9.6.js:108)
msgLog.error @ msgLog.js:168
(anonymous) @ msgLog.js:195
defaultQPromiseErrorHandler @ env.js:928
(anonymous) @ q-0.9.6.js:1553
flush @ q-0.9.6.js:108
```

Your code

Unrelated, generic & useless
Typescript
Typescript

- Adds a type system on top of JS
Typescript

- Adds a type system on top of JS
- We started migrating in late 2014
Typescript

- Adds a type system on top of JS
- We started migrating in late 2014
- Adding types means adding documentation/semantics
Typescript

- Adds a type system on top of JS
- We started migrating in late 2014
- Adding types means adding documentation/semantics
- It also means adding build steps
  - JS just lost 98% of the “quick fix” appeal
Right/Wrong: TypeScript
Right/Wrong: Typescript

- Definitely helps, but still not perfect
Right/Wrong: Typescript

- Definitely helps, but still not perfect
- Still very possible to write buggy code in it
  - Use *all* the strictness options if you have a choice
Right/Wrong: Typescript

- Definitely helps, but still not perfect
- Still very possible to write buggy code in it
  - Use *all* the strictness options if you have a choice
- AMD/CommonJS module system interop was hairy
Right/Wrong: Typescript

- Definitely helps, but still not perfect
- Still very possible to write buggy code in it
  - Use *all* the strictness options if you have a choice
- AMD/CommonJS module system interop was hairy
- Wrapping JS modules with declarations was hairy
Wrong: Too many frameworks
Wrong: Too many frameworks

- Web is all about rapid evolution
Wrong: Too many frameworks

- Web is all about rapid evolution
- Most web apps don’t have 300k LOC
  - Ours did, can’t keep up with web evolution at this codebase size
Wrong: Too many frameworks

- Web is all about rapid evolution
- Most web apps don’t have 300k LOC
  - Ours did, can’t keep up with web evolution at this codebase size
- Ended up with way to many 3rd party frameworks
  - All at once
Wrong: Too many frameworks

- Web is all about rapid evolution
- Most web apps don’t have 300k LOC
  - Ours did, can’t keep up with web evolution at this codebase size
- Ended up with way to many 3rd party frameworks
  - All at once
- Prevents sharing solutions between tools/pages
  - Too many one-offs that can’t be reused
Wrong: Off the shelf is mostly useless
Wrong: Off the shelf is mostly useless

- Most stuff simply doesn’t scale to AAA data sizes
  - Most web sites have dozens of things, we have tens of thousands
  - Different constraints
Wrong: Off the shelf is mostly useless

- Most stuff simply doesn’t scale to AAA data sizes
  - Most web sites have dozens of things, we have tens of thousands
  - Different constraints
- Often designed to be “easy to use” or “friendly”
Case in point: Tree views
Case in point: Tree views

- jstree is a popular web tree control we evaluated
Case in point: Tree views

- jstree is a popular web tree control we evaluated

I would think that performance with a mere 1,000 nodes should be unnoticeable. However, on my i7 machine with 12 GB ram, it takes almost a minute and a half in Chrome to drag and drop those 1,000 nodes somewhere else in the tree.
Case in point: Color picker

- “pick a color you like” vs hard core pro user requirements
Case in point: Color picker

- “pick a color you like” vs hard core pro user requirements

Most web controls
Case in point: Color picker

- “pick a color you like” vs hard core pro user requirements

Most web controls       Advanced web
Case in point: Color picker

- “pick a color you like” vs hard core pro user requirements

Most web controls

Advanced web

Pro space
Case in point: Color picker

- “pick a color you like” vs hard core pro user requirements

Most web controls
Advanced web
Pro space
Case in point: Color picker

- “pick a color you like” vs hard core pro user requirements

Most web controls  Advanced web  Pro space
Case in point: Color picker

- “pick a color you like” vs hard core pro user requirements

Most web controls  Advanced web  Pro space
Case in point: Color picker

- “pick a color you like” vs hard core pro user requirements

Most web controls  Advanced web  Pro space
Case in point: Color picker

- “pick a color you like” vs hard core pro user requirements

Most web controls | Advanced web | Pro space
Case in point: Color picker

- “pick a color you like” vs hard core pro user requirements

Most web controls  Advanced web  Pro space
Right/Wrong: node.js
Right/Wrong: node.js

- Using the web stack from scripts has value
  - Reuse well tested code in batch scripts
Right/Wrong: node.js

- Using the web stack from scripts has value
  - Reuse well tested code in batch scripts
- node.js is a terrible ecosystem
  - Dependency hell (remember left-pad?)
Right/Wrong: node.js

- Using the web stack from scripts has value
  - Reuse well tested code in batch scripts
- node.js is a terrible ecosystem
  - Dependency hell (remember left-pad?)
- After *light* use we lug around 90 MB of deps
  - 8,500 files
Right/Wrong: node.js

- Using the web stack from scripts has value
  - Reuse well tested code in batch scripts
- node.js is a terrible ecosystem
  - Dependency hell (remember left-pad?)
- After *light* use we lug around 90 MB of deps
  - 8,500 files
- Found debugging & profiling super flaky
Wrong: JSON data in code as objects
Wrong: JSON data in code as objects

- Mistake: Used JSON asset data as JS objects directly
  - ‘undefined’ everywhere when data formats change
  - Fix litters code with `if (typeof(thing) !== "undefined")`
Wrong: JSON data in code as objects

- **Mistake:** Used JSON asset data as JS objects directly
  - ‘undefined’ everywhere when data formats change
  - Fix litters code with `if (typeof(thing) !== "undefined")`
- **Obviously bad in hindsight, but was an easy trap to fall into**
  - Maintenance cost is astronomical, because it’s just strings
  - No compile time help, grep and test all code branches
Wrong: JSON data in code as objects

- Mistake: Used JSON asset data as JS objects directly
  - ‘undefined’ everywhere when data formats change
  - Fix litters code with `if (typeof(thing) !== "undefined")`
- Obviously bad in hindsight, but was an easy trap to fall into
  - Maintenance cost is astronomical, because it’s just strings
  - No compile time help, grep and test all code branches
- Partly fixed with symbolic access library (too late)
Wrong: “It’s OK, it’s just in the Javascript”
Wrong: “It’s OK, it’s just in the Javascript”

- Belief that it was OK to hack stuff in because “it’s just script”
  - Very easy to deploy hacks and local changes and get away with it
  - Tempting to work around hot issues this way
  - Probably our C++ bias led to this mistake
Wrong: “It’s OK, it’s just in the Javascript”

- Belief that it was OK to hack stuff in because “it’s just script”
  - Very easy to deploy hacks and local changes and get away with it
  - Tempting to work around hot issues this way
  - Probably our C++ bias led to this mistake

- Manifested as a legacy of unmaintainable hacks
  - Performance regressions, copy-pasta
  - Especially bad in first generation JS code which is still in production
The Document Object Model (DOM)
The Document Object Model (DOM)

- Hardest part of performant web tools
  - Black art
The Document Object Model (DOM)

- Hardest part of performant web tools
  - Black art
- Reflows kill your performance
  - Sandwich in a complete rebuild of your UI at “random” places
The Document Object Model (DOM)

- Hardest part of performant web tools
  - Black art
- Reflows kill your performance
  - Sandwich in a complete rebuild of your UI at “random” places
- Using animation frames is critical to performance
  - Not well understood
Lesson: JS needs *more* rigor than C++, not less
Lesson: JS needs *more* rigor than C++, not less

- Suspect this is true in any dynamically typed ecosystem
Lesson: JS needs more rigor than C++, not less

- Suspect this is true in any dynamically typed ecosystem
- Especially hard for people jumping in only occasionally
  - Typically “non JS” programmers left huge messes
  - Did not want to invest in learning the “web tools way”
Wrong: Have you cleared cache?
Wrong: Have you cleared cache?

- Chrome caches very aggressively
  - Clear cache, or risk running with random JS code out of date
Wrong: Have you cleared cache?

- Chrome caches very aggressively
  - Clear cache, or risk running with random JS code out of date
- Constant drain on support and user base
  - Handful of support issues every week due to this
Wrong: Have you cleared cache?

- Chrome caches *very* aggressively
  - Clear cache, or risk running with random JS code out of date
- Constant drain on support and user base
  - Handful of support issues every week due to this
- Laughably bad compared to native tool deployment
From: -REDACTED-
Sent: Tuesday, January 31, 2017 6:30 PM
To: Syndicate
Subject: FW: BUG - cannot create new collections

I swung by –REDACTED–,

No amount of clearing cache/reloading extensions/restarting chrome got this fixed. We cannot repro either. Anyone know what might be going on here?
Wrong: “Try reinstalling the plugin”
Wrong: “Try reinstalling the plugin”

- Plugin part of the desktop interop
  - 3D view, window management
Wrong: “Try reinstalling the plugin”

- Plugin part of the desktop interop
  - 3D view, window management
- NPAPI drop mandated rewrite
Wrong: “Try reinstalling the plugin”

- Plugin part of the desktop interop
  - 3D view, window management
- NPAPI drop mandated rewrite
- Forced Chrome Store requirement
  - Now have security warnings on every launch of the tools
Wrong: “Try reinstalling the plugin”

- Plugin part of the desktop interop
  - 3D view, window management
- NPAPI drop mandated rewrite
- Forced Chrome Store requirement
  - Now have security warnings on every launch of the tools
Wrong: You’re not in charge
Wrong: You’re not in charge

- Chrome team doesn’t care about your tools
Wrong: You’re not in charge

- Chrome team doesn’t care about your tools
- Auto-updates *will* break everything
  - Constant source of downtime & panic
  - Random API changes, flash 0-day bugfixes, V8 regressions
Wrong: You’re not in charge

- Chrome team doesn’t care about your tools
- Auto-updates *will* break everything
  - Constant source of downtime & panic
  - Random API changes, flash 0-day bugfixes, V8 regressions
- Can run Canary builds to catch some of it early
  - But you’re still not in charge
  - Always a scramble to fix things last moment
Policies = The Nuclear Option

- You *can* control some policies for Chrome
Policies = The Nuclear Option

- You *can* control some policies for Chrome
- Used auto-update disable hammer to ship Sunset
  - Frozen branch meant we couldn't get Chrome breakage fixes into that game's codebase
Policies = The Nuclear Option

- You *can* control some policies for Chrome
- Used auto-update disable hammer to ship Sunset
  - Frozen branch meant we couldn't get Chrome breakage fixes into that game's codebase
- Plenty of user pushback about restricting Chrome
  - Favored daily browser
Policies = The Nuclear Option

- You *can* control some policies for Chrome
- Used auto-update disable hammer to ship Sunset
  - Frozen branch meant we couldn't get Chrome breakage fixes into that game's codebase
- Plenty of user pushback about restricting Chrome
  - Favored daily browser
- Should have done this from day 1
  - And would have been hard sell then too
From: -REDACTED-
Sent: Monday, October 14, 2013 10:51 AM
To: Lunarest
Subject: ads in Luna

This awesome ad popped up in my workspace last week. Not sure if this issue has anything to do with whatever’s causing the white bar at the bottom of the screen.
1 Weird Tip For Gaining Muscle Fast
See how this 1 weird tip makes you gain muscle fast. *See How*
Wrong: A hostile environment for your tools
Wrong: A hostile environment for your tools

- Malware
Wrong: A hostile environment for your tools

- Malware
- Rampant extensions, addons and random junk
  - AdBlock added 30 ms of latency for each outgoing AJAX request
  - When you have 1000s of them, it starts to really suck
Wrong: A hostile environment for your tools

- Malware
- Rampant extensions, addons and random junk
  - AdBlock added 30 ms of latency for each outgoing AJAX request
  - When you have 1000s of them, it starts to really suck
- Themes
Wrong: Culture Shock
Wrong: Culture Shock

- JS and web stack is very different from engine dev
  - Deep specialization inevitable and necessary to do the job
Wrong: Culture Shock

- JS and web stack is very different from engine dev
  - Deep specialization inevitable and necessary to do the job
- Started to see cracks in the team culture
At road’s end (2015)
At road’s end (2015)

- Dev time for features snowballed
  - Frustrated at lack of progress
  - Lots of bugs were outside our control
  - Maintenance costs killing productivity
At road’s end (2015)

- Dev time for features snowballed
  - Frustrated at lack of progress
  - Lots of bugs were outside our control
  - Maintenance costs killing productivity
- Had lost 2 of our JS specialists
At road’s end (2015)

• Dev time for features snowballed
  • Frustrated at lack of progress
  • Lots of bugs were outside our control
  • Maintenance costs killing productivity
• Had lost 2 of our JS specialists
• Team tension was mounting
  • We needed a new direction
Our current tools direction (2016-)
Our current tools direction (2016-)

- Client layer in C++, with Qt for UI
  - Right cultural choice for us
  - Go back to mature dev tools, pipelines
Our current tools direction (2016-)

- Client layer in C++, with Qt for UI
  - Right cultural choice for us
  - Go back to mature dev tools, pipelines
- Keep what works!
  - Edit loop, server infrastructure – enables gradual transition
Our current tools direction (2016-)

• Client layer in C++, with Qt for UI
  • Right cultural choice for us
  • Go back to mature dev tools, pipelines
• Keep what works!
  • Edit loop, server infrastructure – enables gradual transition
• 9 months to get complete level editor beta up
  • Included rewriting all JS infrastructure in C++
2016 Surprises
2016 Surprises

- Maintenance cost went way down as we stopped JS dev
  - Breakage in production proportional to # of JS changes
2016 Surprises

- Maintenance cost went way down as we stopped JS dev
  - Breakage in production proportional to # of JS changes
- Still firefighting things outside our control
  - Auto-updates: gift that keeps on giving
  - Flash update broke all node graphs over night
  - About a week of engineering effort to drop everything and fix
How could you make it work?
How could you make it work?

- Embedded browser engine
  - No auto-updates, add-ins and other craziness
  - No more clear cache
How could you make it work?

- Embedded browser engine
  - No auto-updates, add-ins and other craziness
  - No more clear cache
- Use TypeScript exclusively
How could you make it work?

- Embedded browser engine
  - No auto-updates, add-ins and other craziness
  - No more clear cache
- Use TypeScript exclusively
- Settle on small number of technologies
  - Use them consistently even if they’re not the latest web craze
How could you make it work?

- Embedded browser engine
  - No auto-updates, add-ins and other craziness
  - No more clear cache
- Use TypeScript exclusively
- Settle on small number of technologies
  - Use them consistently even if they’re not the latest web craze
- Be conscious of specialization and team culture
Vision post-mortem: Hiring
Vision post-mortem: Hiring

- We put the word out, and people know about our web tools..
Vision post-mortem: Hiring

- We put the word out, and people know about our web tools..
- ..but no web specialists applied
Vision post-mortem: Hiring

- We put the word out, and people know about our web tools..
- But no web specialists applied
- Even if they had applied, would we have hired them?
  - We need generalists on the Core team
Vision post-mortem: Lower the UI barrier
Vision post-mortem: Lower the UI barrier

- We found that web apps of this magnitude *raise* the barrier
  - Extra context switching: JavaScript, client-server, DOM, HTML
  - Takes months to learn all the intricacies of the tech stack
Vision post-mortem: Lower the UI barrier

- We found that web apps of this magnitude raise the barrier
  - Extra context switching: JavaScript, client-server, DOM, HTML
  - Takes months to learn all the intricacies of the tech stack
- It’s no better than a traditional desktop GUI dev
  - At least not for apps this size
  - You can totally make a throwaway log viewer in 10 minutes
  - But no one wants that thing in production
Conclusion
Conclusion

- No regrets – 6 games shipped speak for themselves
Conclusion

- No regrets – 6 games shipped speak for themselves
- Our engine team is very versatile – but why stretch it?
Conclusion

- No regrets – 6 games shipped speak for themselves
- Our engine team is very versatile – but why stretch it?
- Probably wouldn't have done this if we knew the real costs
  - First explorer gets to discover all the traps 😊
Conclusion

- No regrets – 6 games shipped speak for themselves
- Our engine team is very versatile – but why stretch it?
- Probably wouldn't have done this if we knew the real costs
  - First explorer gets to discover all the traps 😊
- It could have been a lot smoother if we prepared better
Conclusion

- No regrets – 6 games shipped speak for themselves
- Our engine team is very versatile – but why stretch it?
- Probably wouldn't have done this if we knew the real costs
  - First explorer gets to discover all the traps 😊
- It could have been a lot smoother if we prepared better
- “Zero install” remains an attractive goal
  - Can probably get close with native tools too though
Thanks

- Q & A
- afredriksson@insomniacgames.com
- @deplinenoise on twitter