



Best Practices

For developing a web game
in modern browsers 

Colt "MainRoach" McAnlis
3.05.2012

The call me "web game"

Content Server

Database

Gameplay Server



Google App Engine



Google Analytics

Client

Scripting

Display



Social Graph



Chrome Web Store



Google Wallet

Monetize

The Rundown

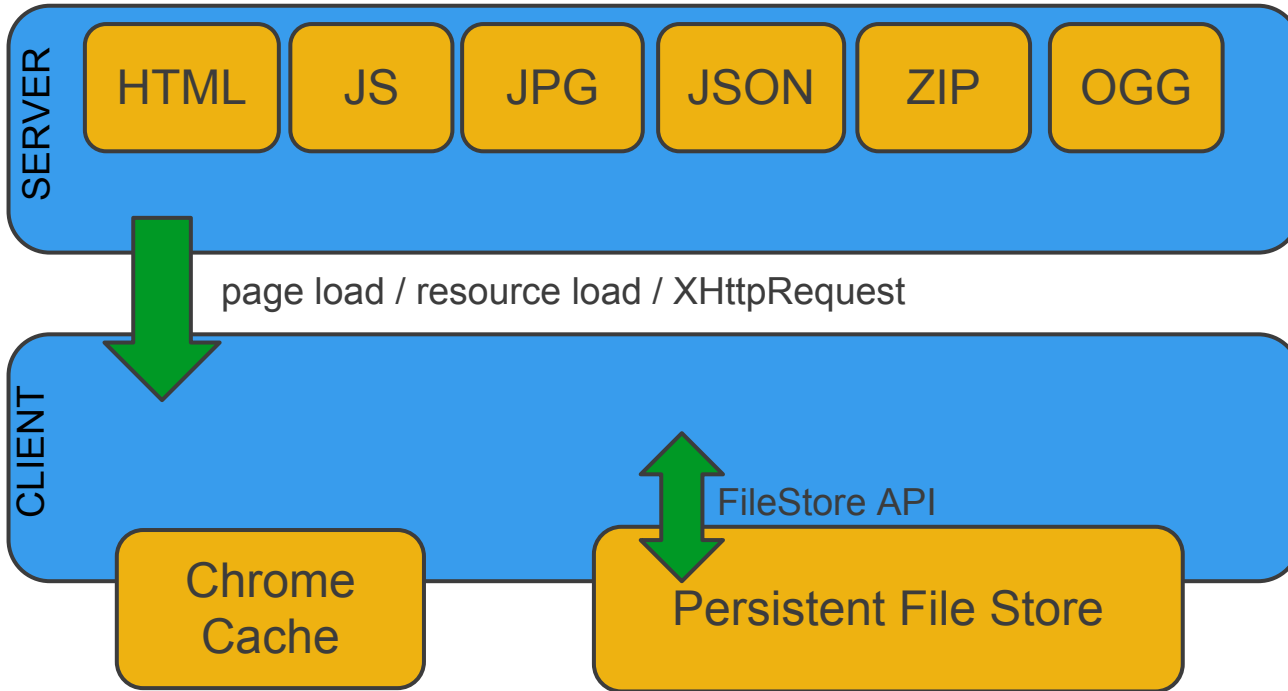
Content Distrib
Database
Login & Auth
Localization
Rendering
Web & Platform
User Metrics
Monetization



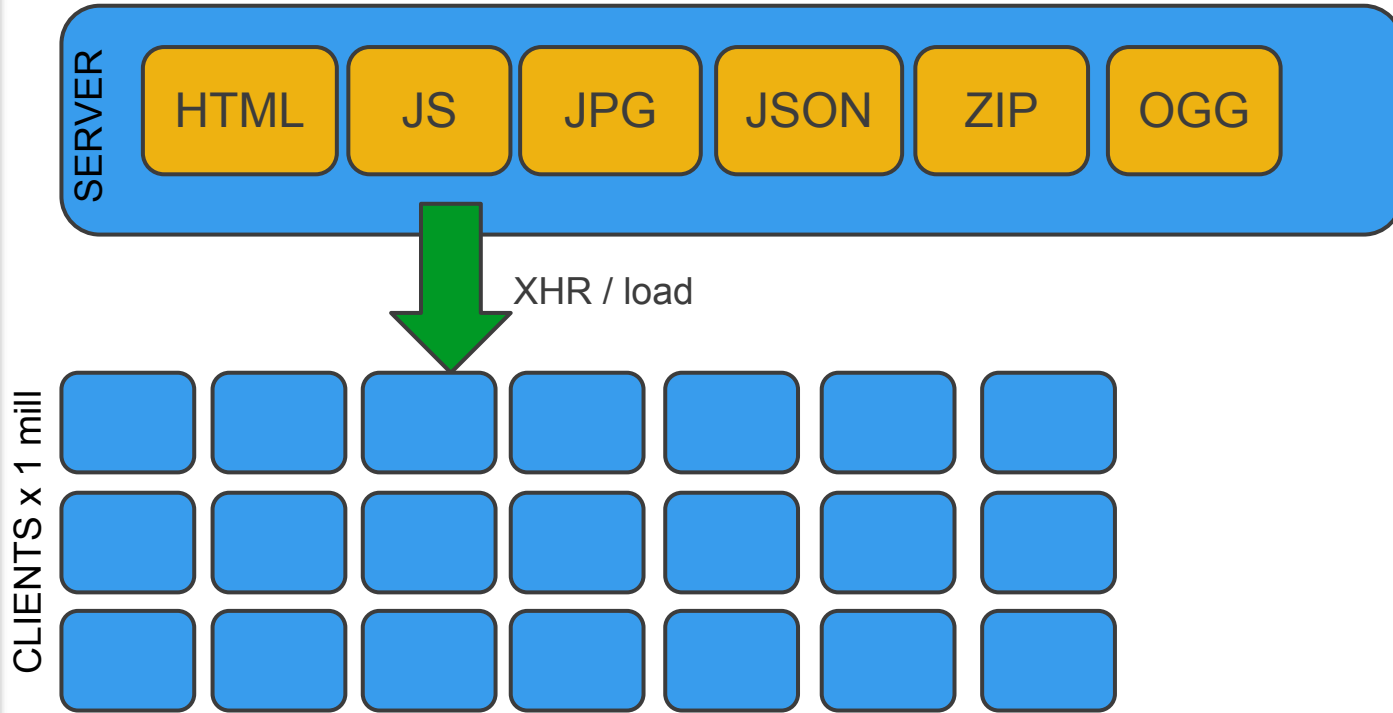
Serving content



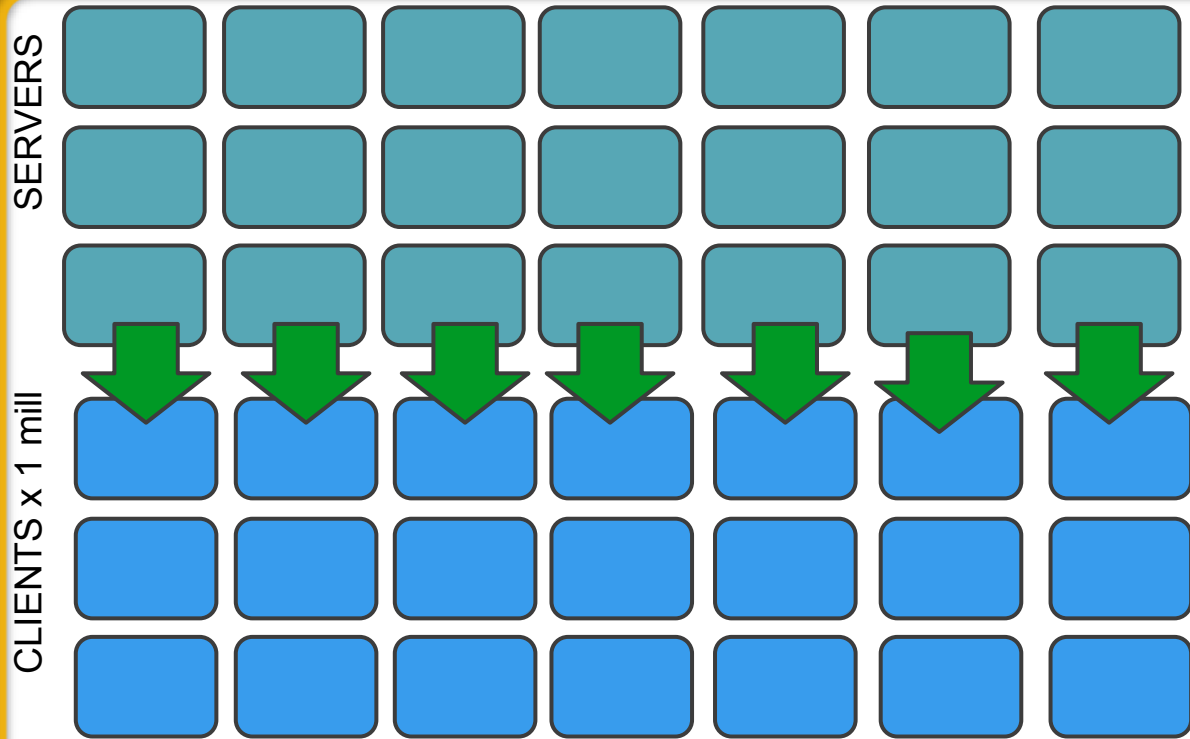
Serving & Caching



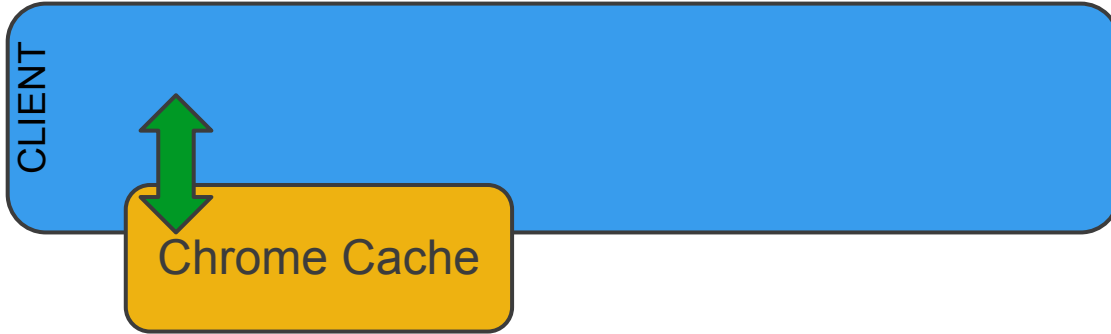
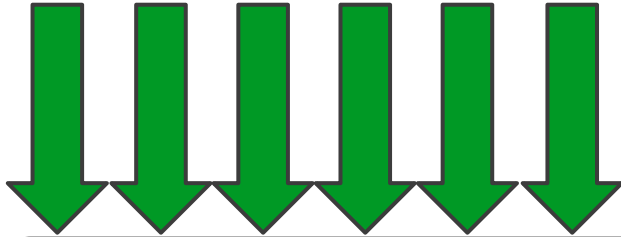
Serving & Scaling



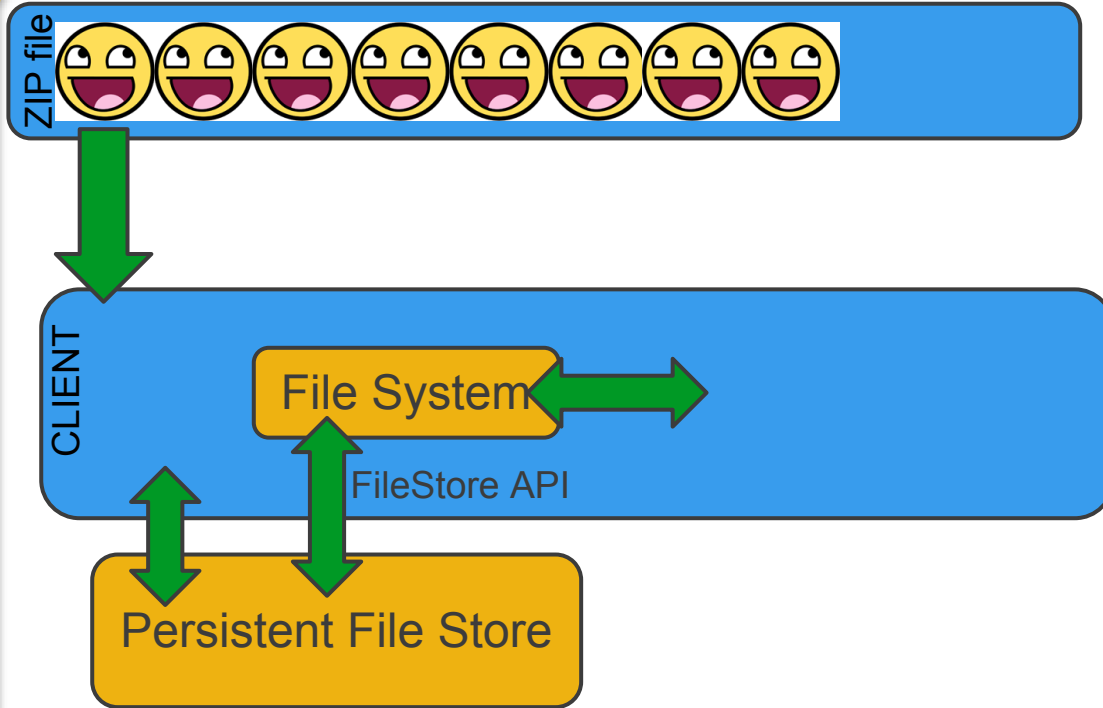
Serving & Scaling



Loose Files



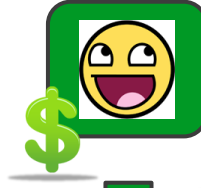
Archiving Wins



Segmenting & updating



ZIP
FILES



CLIENT



Persistent File Store

GAE Data storage



Store segmented binary archives here

Can carefully control serving costs

<32mb served HTML style

>=32mb served from [blobstore API](#)

Set Browser Cache expiration date - [link](#)

Lets the browse know how long an object is valid

Default is 10 min!

Public Service Announcement

Do Not use the 'existence' of data
to represent 'user has bought
this'

Database



App Engine



Available database using [GQL](#) query language

Auto-scales to handle demand

Cost based upon performance

App Engine



Set the cache-time for the GET responses

```
Date now = new Date();  
long age = 86400L;  
  
resp.setDateHeader("Date", now.getTime());  
resp.setHeader("Cache-Control", "public, s-maxage=" + age);
```

App Engine - Memcache

Use Memcache - [link](#)

Intended for fast access to cached results of queries

```
def get_data():  
    data = memcache.get("key")  
    if data is not None:  
        return data  
    else:  
        data = self.query_for_data()  
        memcache.add("key", data, 60)  
        return data
```


Sync data between clients

Polling

Each client asks the server continuously

Channel API - [link](#)

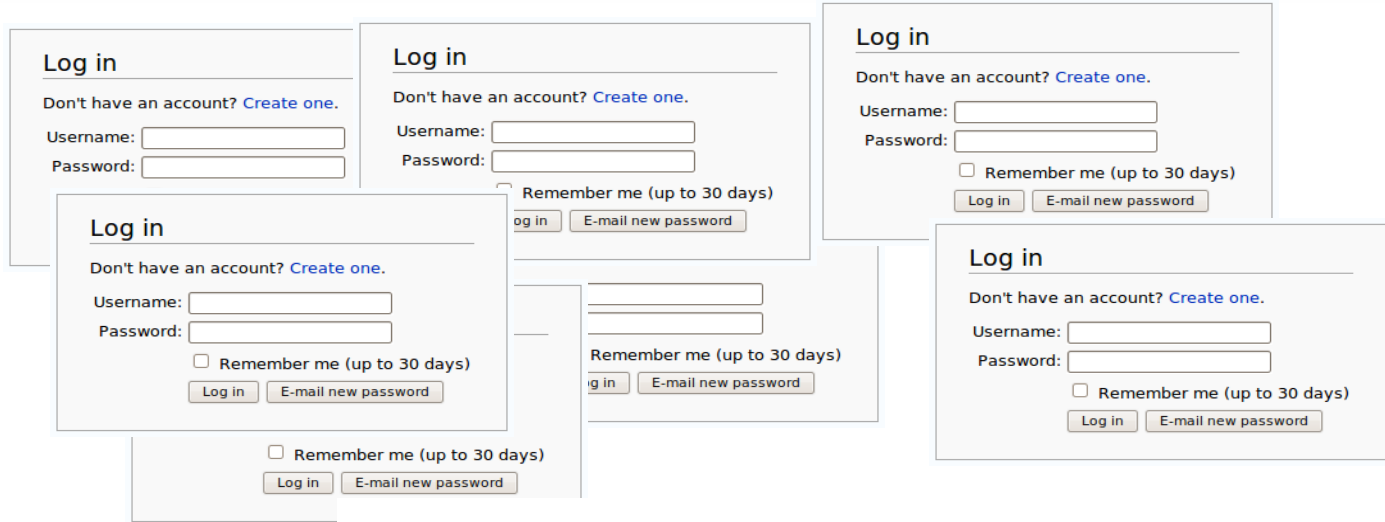
server tells client when data is ready

Can only communicate with javascript!

Login & Auth



Mo' logins, Mo' problems



The diagram illustrates the problem of having many different login interfaces by showing several overlapping login forms. Each form is titled "Log in" and contains the following elements:

- A link: "Don't have an account? [Create one.](#)"
- Input fields for "Username:" and "Password:"
- A checkbox: "Remember me (up to 30 days)"
- Buttons: "Log in" and "E-mail new password"

The forms are arranged in a way that suggests a user might encounter multiple, slightly different versions of these login screens, leading to confusion and inconsistency.



Mo' logins, less' problems

Sign-in or Create New Account

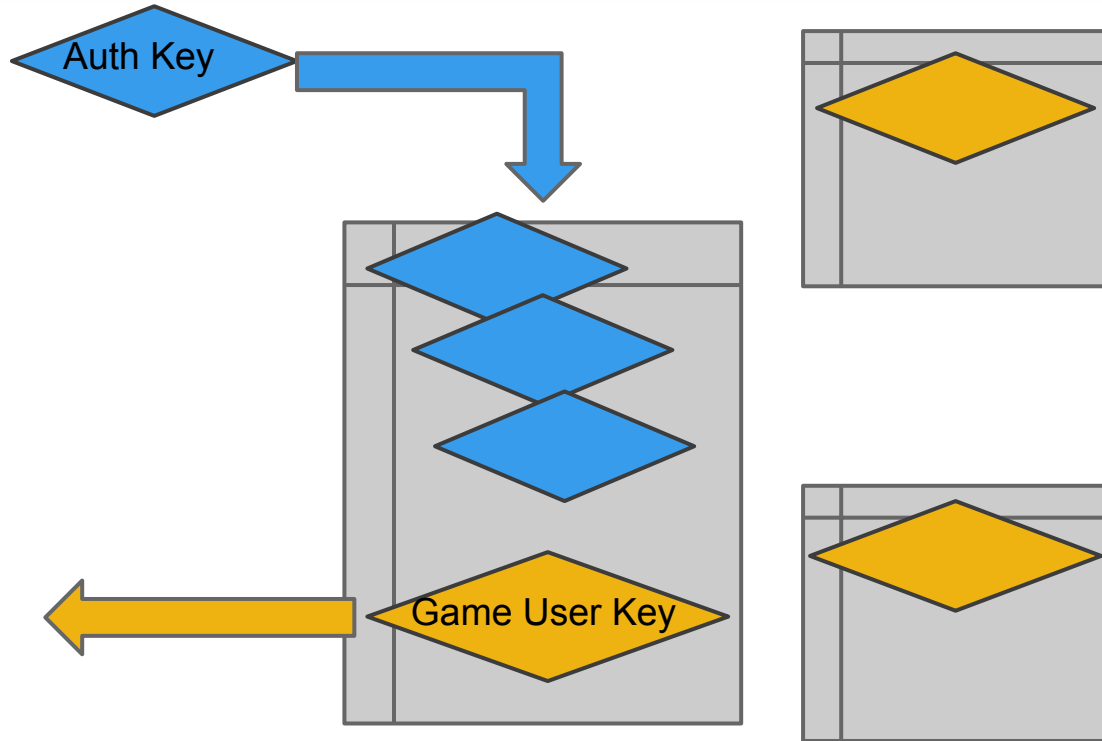
Please click your account provider:



Your Blogger account

Sign-In

Keep your keys




Two step security

Sign-in or Create New Account

Please click your account provider:

Google	YAHOO!	AOL	myOpenID	OpenID
--------	--------	-----	----------	--------



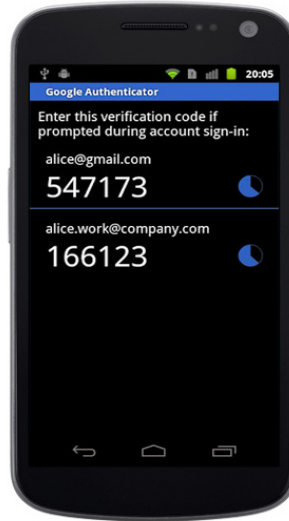
Your Blogger account

Enter the verification code generated by your mobile application.

Enter code:

☐ Remember this computer for 30 days.

[verification code »](#)



More - [link](#).

Localization



Localizing matters!

Big markets internationally

Languages don't act the same

Languages read in different directions

It changes your UI layout!

Long words / phrases

Detecting language

Use [navigator.language](#) in JS

browser settings don't change this though

Browser settings affect Http-Accept header

Fetch via polling a server function

Fetch Language

```
var language;  
$.ajax({  
  url: "http://ajaxhttpheaders.appspot.com",  
  dataType: 'jsonp',  
  success: function(headers) {  
    language = headers['Accept-Language'];  
    nowDoSomethingWithIt(language);  
  }  
});
```

More - [link](#).

Get your translate on

Translate

From: English - detected ▼



To: Spanish ▼

Translate

English Spanish French

HA HA I AM GIVING A PRESENTATION!



English Spanish Arabic

JA JA ME DA UNA PRESENTACIÓN!



New! Click the words above to view alternate translations. [Dismiss](#)

Translate API

<https://www.googleapis.com/language/translate/v2/detect?key=INSERT-YOUR-KEY&q=google+translate+is+fast>

Tooling!

Translator Toolkit [index](#) modified Jan 24 by colton

Share Show toolkit Edit View Save Save & Close

Original text:

Translation: English » Spanish

100% complete, 542 w

Native Client SDK Examples

This page lists all of the examples available in the most recent Native Client SDK bundle. Each example is designed to teach a few specific Native Client programming concepts.

[Hello World in C](#)

The Hello World In C example demonstrates the basic structure of all Native Client applications. This example loads a module and responds to button click events by showing alert panels.

Teaching focus: Basic HTML, JavaScript, and module architecture; Messaging API.

[Hello World in C++](#)

The Hello World C++ example demonstrates the basic structure of all Native Client applications. This example loads a module and responds to button click events by showing alert panels.

Teaching focus: Basic HTML, JavaScript, and module architecture; Messaging API.

[Load Progress](#)

The Load Progress example demonstrates how to listen for and handle events that occur while a NaCl module loads. It listens for different load event types and dispatches different events to their respective handler. This example also checks browser version and shows how to calculate and display loading progress.

Teaching focus: Progress event handling.

[Pi Generator](#)

The Pi Generator example demonstrates creating a helper thread that estimate pi using the Monte Carlo method while randomly putting 1,000,000,000 points inside a 2D square that shares two sides with a quarter circle.

Teaching focus: Thread creation, 2D graphics, view change events.

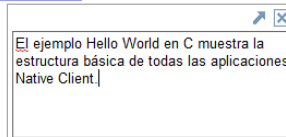
[Input Events](#)

The Input Events example demonstrates how to handle events triggered by the user. This example allows a user to interact

Native Client SDK Ejemplos

Esta página muestra todos los ejemplos disponibles en el más reciente paquete de Native Client SDK. Cada ejemplo está diseñado para enseñar algunos conceptos específicos de programación nativo del cliente.

[Hola Mundo en C](#)



Characters: 97

Actions

[«Previous](#) [Next»](#)

En este ejemplo se carga un módulo Native Client y responde a los eventos de clic de botón que muestra los paneles de alerta.

Centrar la enseñanza: básica HTML, JavaScript, y la arquitectura del módulo, la API de mensajería.

[Hola Mundo en C++](#)

El Hola Mundo en C++ ejemplo, se muestra la estructura básica de todas las aplicaciones Native Client. En este ejemplo se carga un módulo Native Client y responde a los eventos de clic de botón que muestra los paneles de alerta.

Centrar la enseñanza: básica HTML, JavaScript, y la arquitectura del módulo, la API de mensajería.

[El progreso de carga](#)

En el ejemplo de progreso de la carga se muestra cómo detectar y controlar los eventos que ocurren mientras se carga un módulo NaCl. En este ejemplo se escucha para diferentes tipos de eventos de carga y distribuye eventos diferentes a su controlador correspondiente. Este ejemplo también se comprueba la versión del navegador válido y muestra cómo calcular y mostrar el progreso de carga.

Display & Rendering



Using GPU on the web



GPU Drivers not always safe...

Black-listed drivers

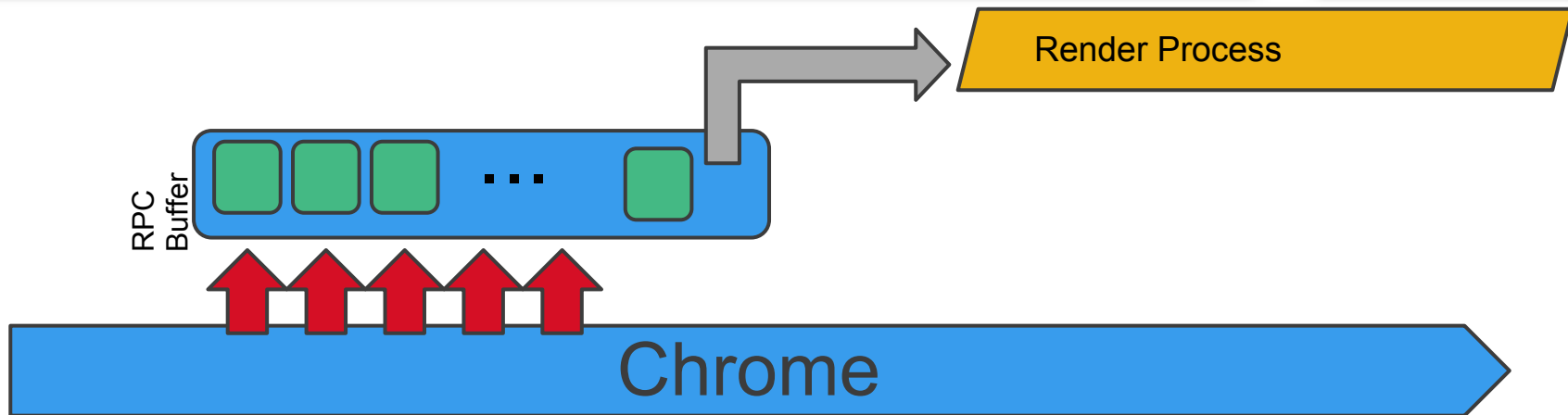
Detect and respond early!

Blacklisted driver test

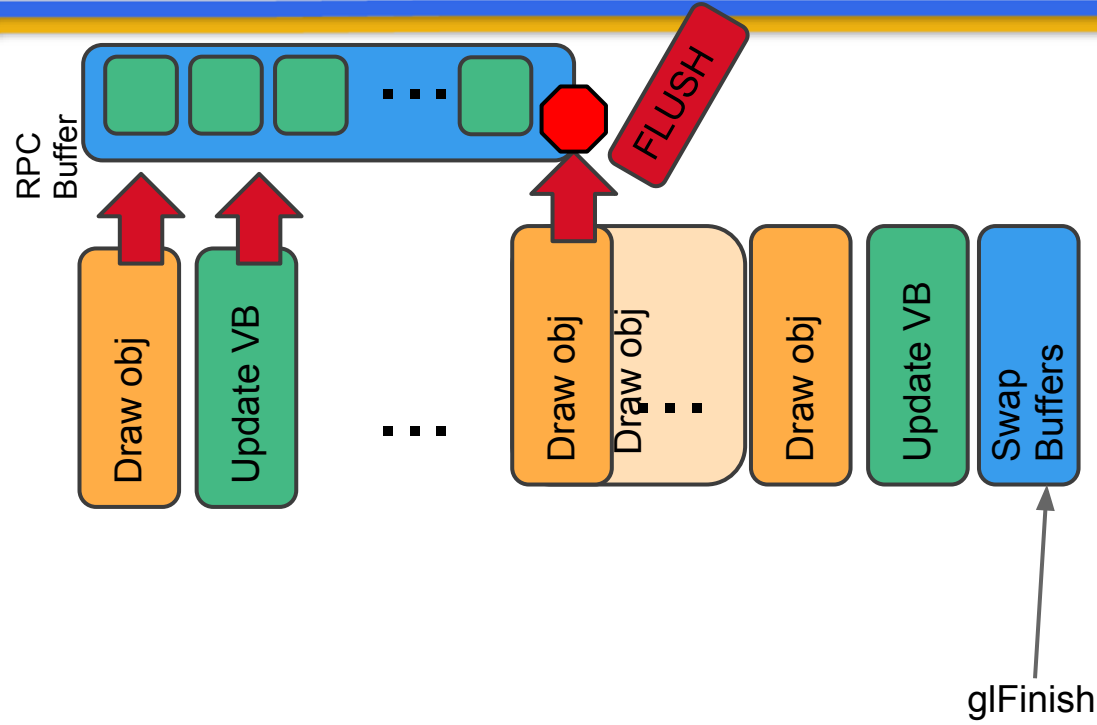
//has blacklisted hardware / feature sets?

```
function textureSizeTest(size) {  
  var canvas = document.createElement('canvas');  
  var gl = canvas.getContext('webgl') || canvas.getContext('experimental-webgl');  
  
  if (gl) {  
    return gl.getParameter(gl.MAX_TEXTURE_SIZE) >= size;  
  }  
}
```

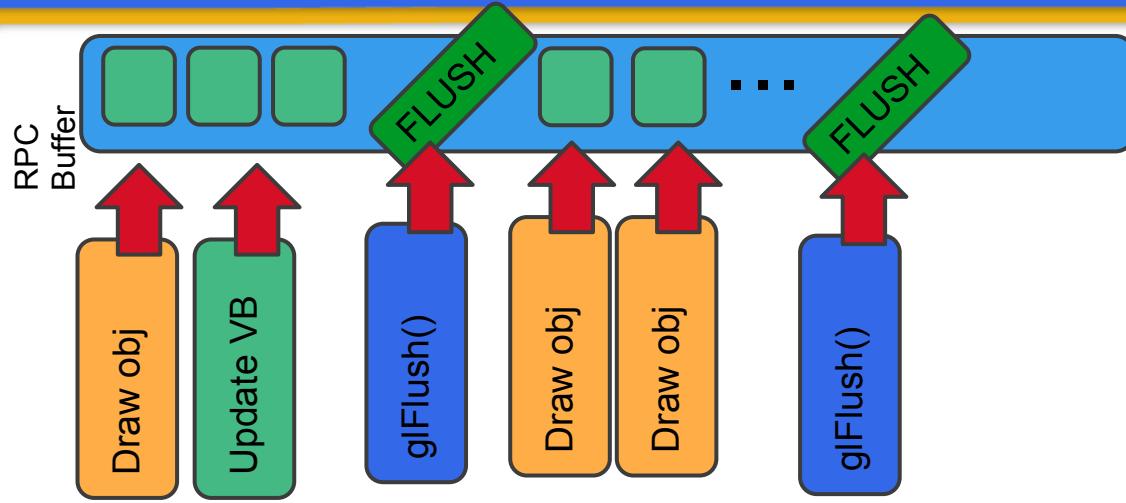

Sandbox Rendering



Render Stalls



Render Stalls



Canvas rendering

GPU accelerated canvas rendering

Chrome 17

Hands-free usage

Setup behind the scenes

User can toggle flag

Platform env



Lost attention

Users will tab away from your page.

Detect this - [visibilitychange](#) event

Respond accordingly

- pause game, reduce volume

- setInterval will slow down to 1000ms

- RequestAnimationFrame will stop completely

Bugs in the wild

Detect this

[window.onerror](#), v8-[Error.captureStackTrace](#)

Respond accordingly

Gather -

game specific - game state, OS, player ID, etc

[env - window.performance](#) / [chrome.tabs.captureVisibleTab](#)

Upload -

[jsErrLog](#) / [jdrops](#)

Lots 'o bugs

Make sure your error reporting is stable

Display errors to user

be friendly, avoid technical language

Respect your users' privacy

Ensure user knows what's being reported

Offer ability to opt-out

Minimize & obfuscate

Closure compiler - [link](#)

plovr - [link](#)

HTML5 boiler plate - [link](#)

User Metrics



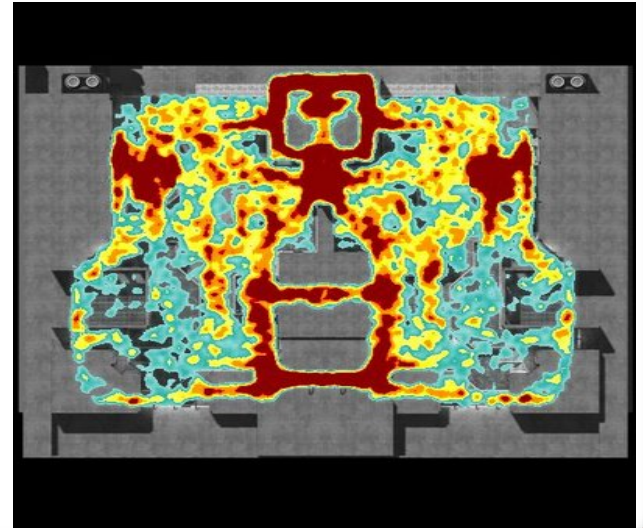
Know thine players

Use Google Analytics!

Can kick off custom events too!

Custom game events

finish lvl 2, finish a game,
chatted, join a group,
assault a chicken...



The high points

Use GA

Whats your daily, weekly, monthly averages?

% of players come from what site

of players that click 'buy'

of players that complete a 'buy'

Use own server

track in-game events

crashes, bugs, specific errors

AB testing

Is there a secret sauce to a better game?

Disconnect between devs & users

Issue separate builds to 50% of your users

Track desired outcome somehow

Make changes to optimize your outcome

Might be multiple split by region / user

Monetization



Guide the user

Monetization is a dark art.

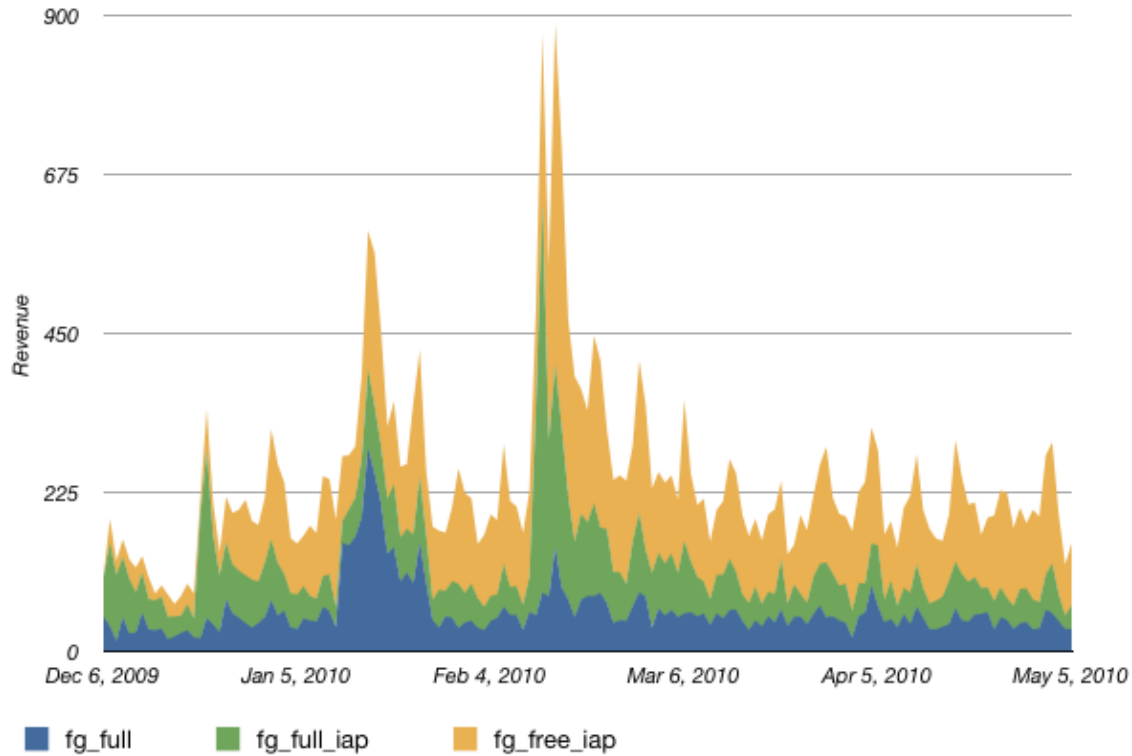
game specific, platform specific etc

Hold the user's hand

How do they purchase

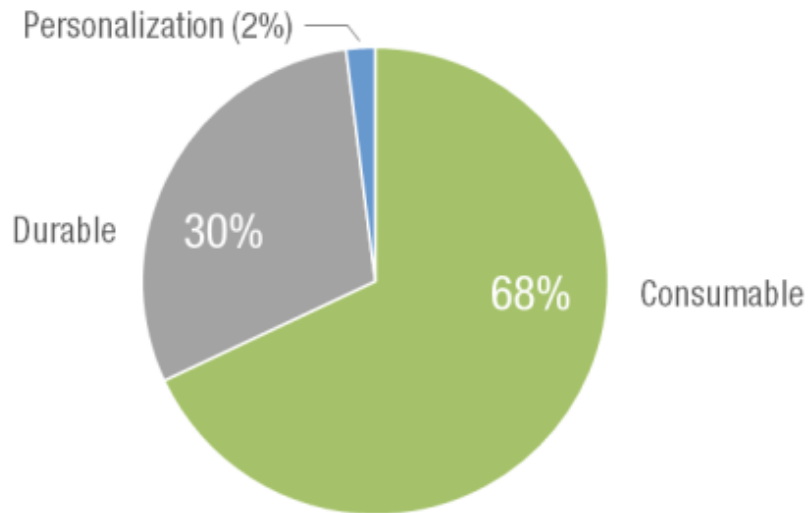
Why should they purchase

Premium vs freemium



For sale

iOS & Android Freemium Games, Dollars Spent on Virtual Goods



Virtual currency

Have a multiple currency system

- Premium currency from IAP

- Grind currency from gameplay

- Enable trading between currencies

Enable Instant gratification

Simplified purchase flow

Leverage In-App Payment APIs
rather than directing user to a pay-site

Great features

Access to millions of existing CCs

Powerful fraud engine

Plug and play purchase flow

Pay in a few clicks



Google Wallet

Take outs



Takeouts

Bundle assets into segmented archives

Set proper cache times for data & query fetches

Use a single-login service for your users

Localize your content properly

Detect GPU blacklist and performance changes

Takeouts

Detect and fetch bugs in the wild

Detect and respond to user focus loss

Track as much user data as possible

Choose monetization metric that fits your game

Focus on Consumable and durable goods

Focus on in-app API for purchases.

Takeouts

Lots of work to develop a game

That has nothing to do with your game

Leverage Google services

Free to get started



Thanks!



colton@google.com



[+Colt McAnlis](https://plus.google.com/+ColtMcAnlis)



[@duhroach](https://twitter.com/duhroach)



mainroach.blogspot.com

default text color

Charts, Strokes

supported text

Google

Additional 1