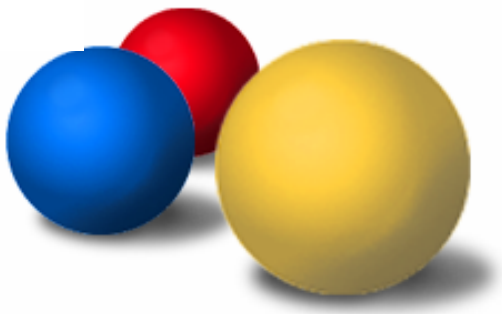
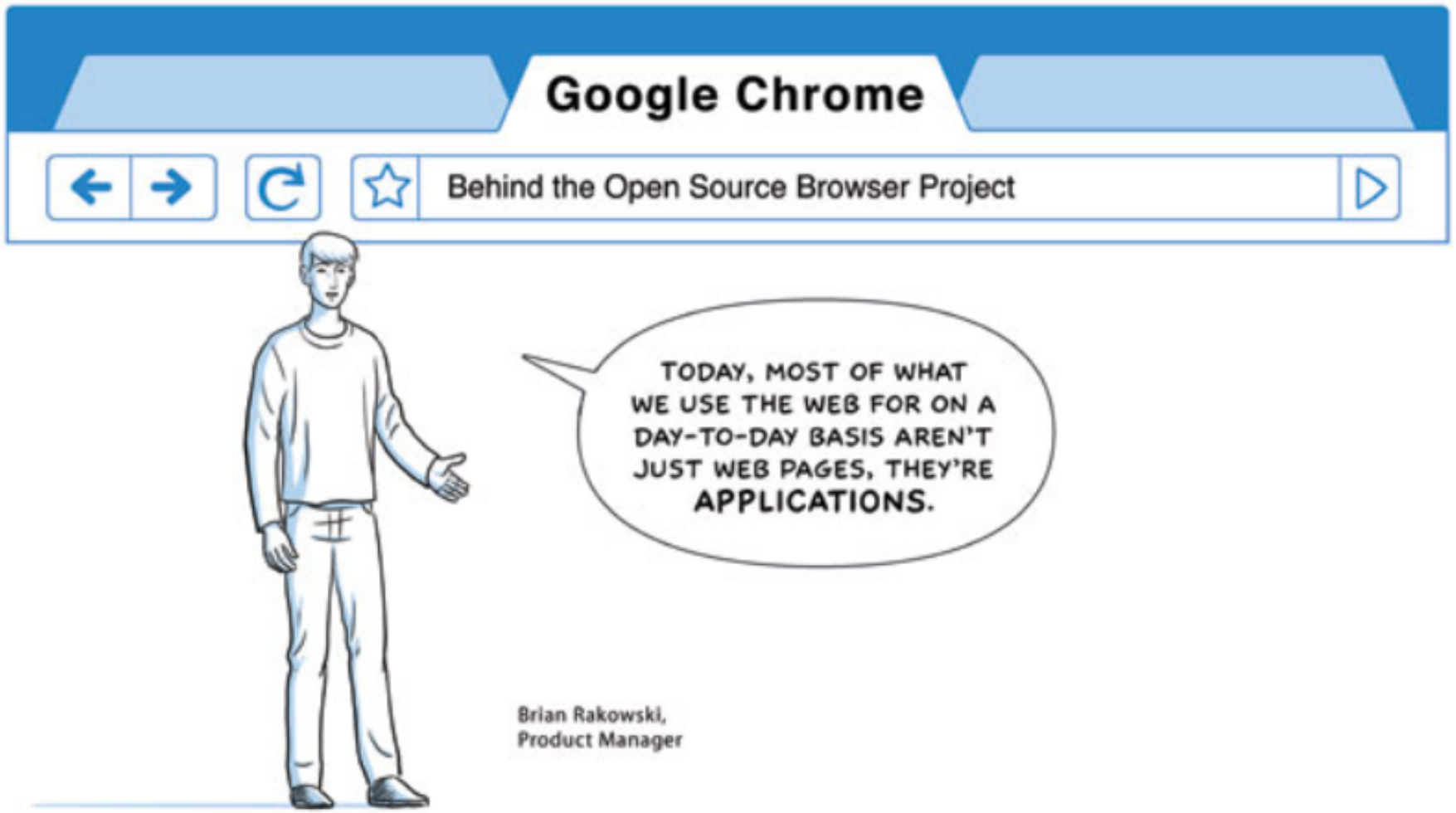
Google Chrome

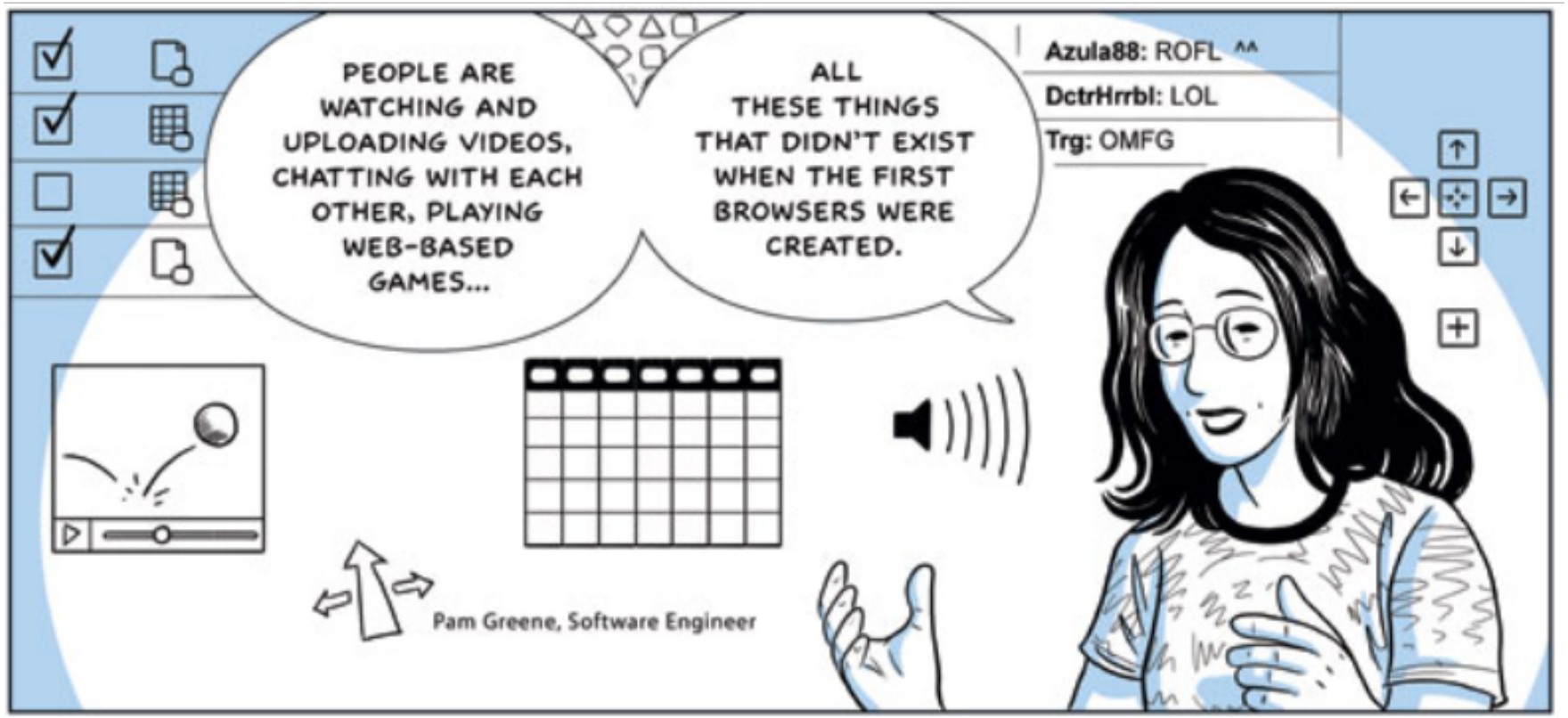


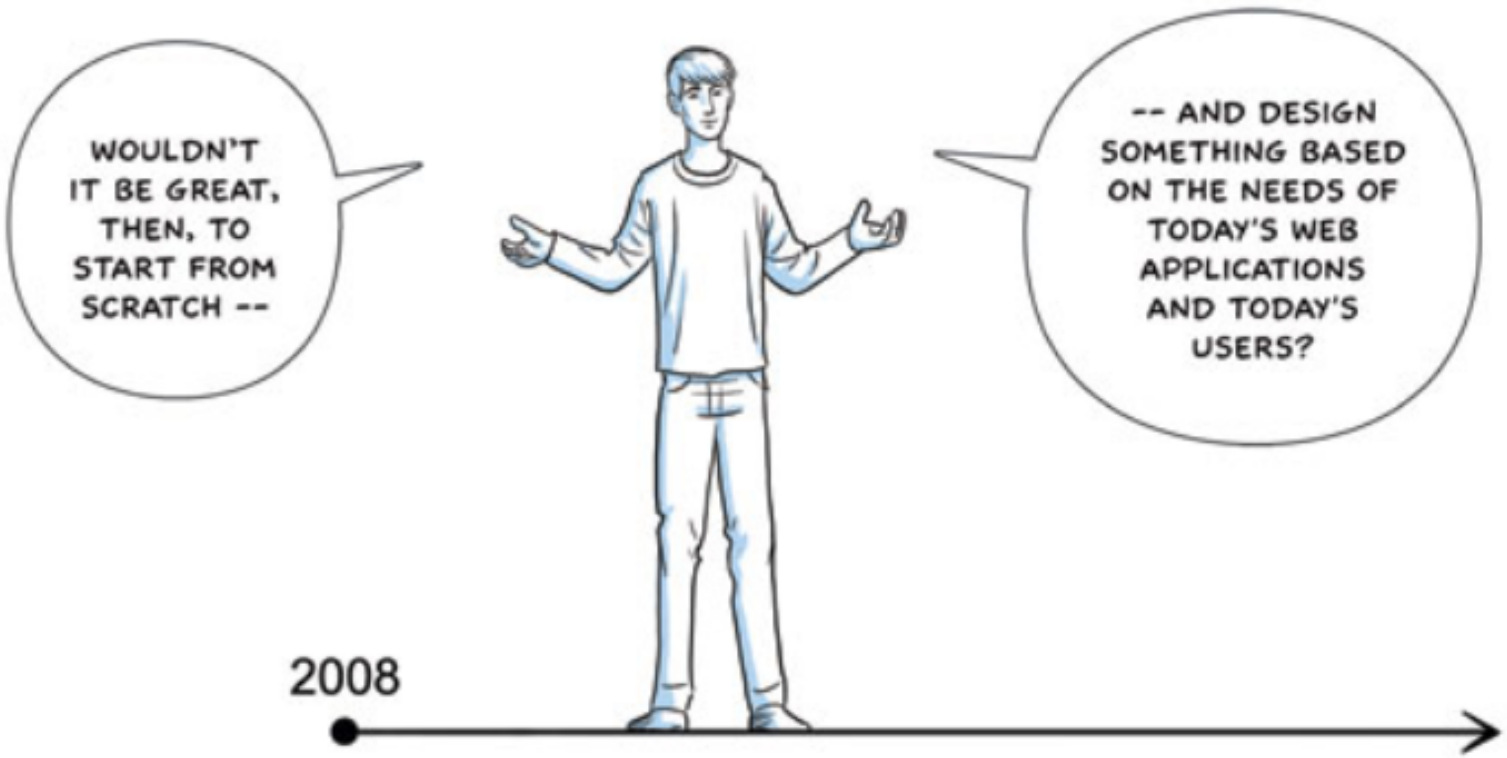
Presented by Alex Nicolaou

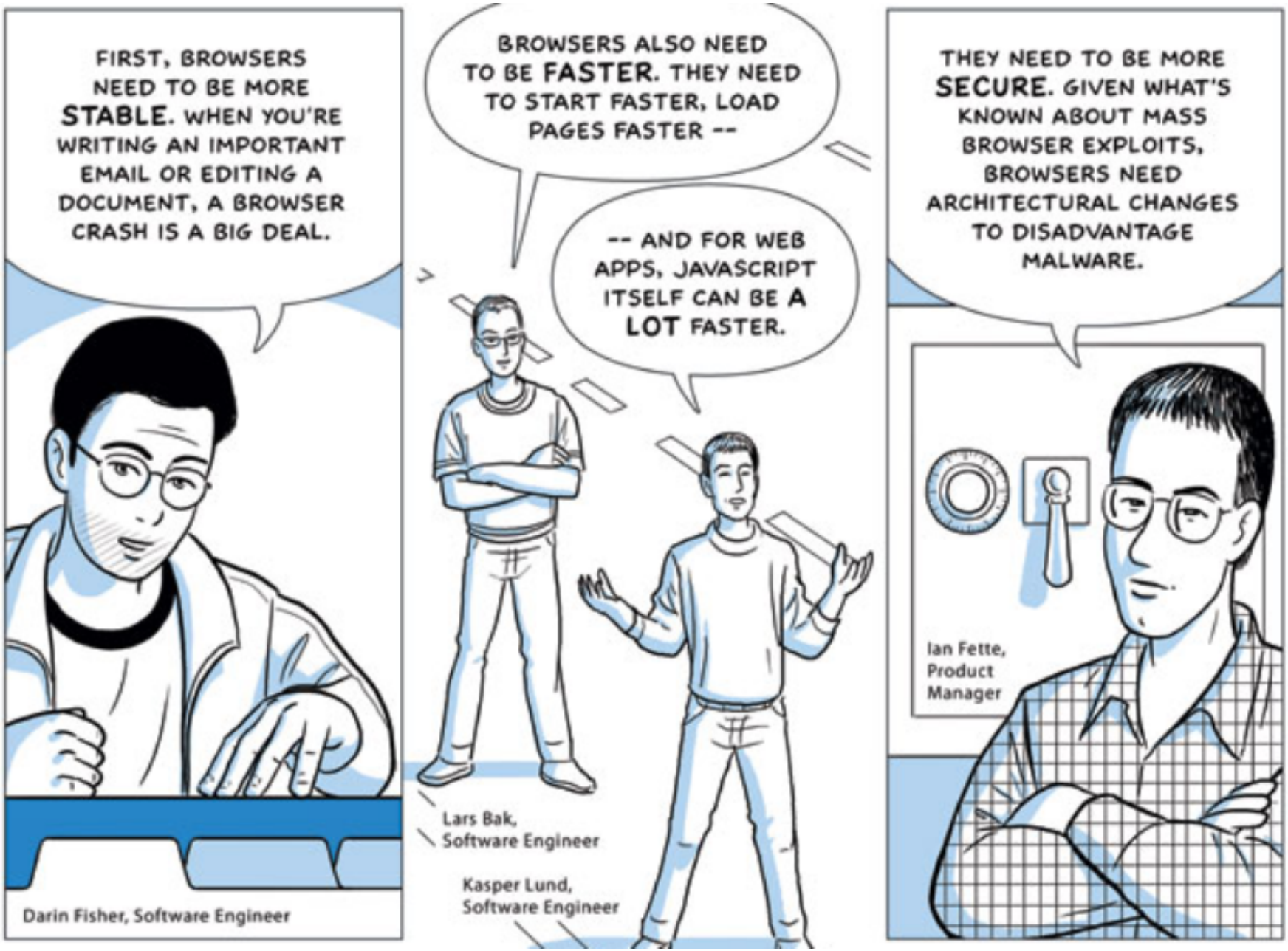
The world wide Application Server

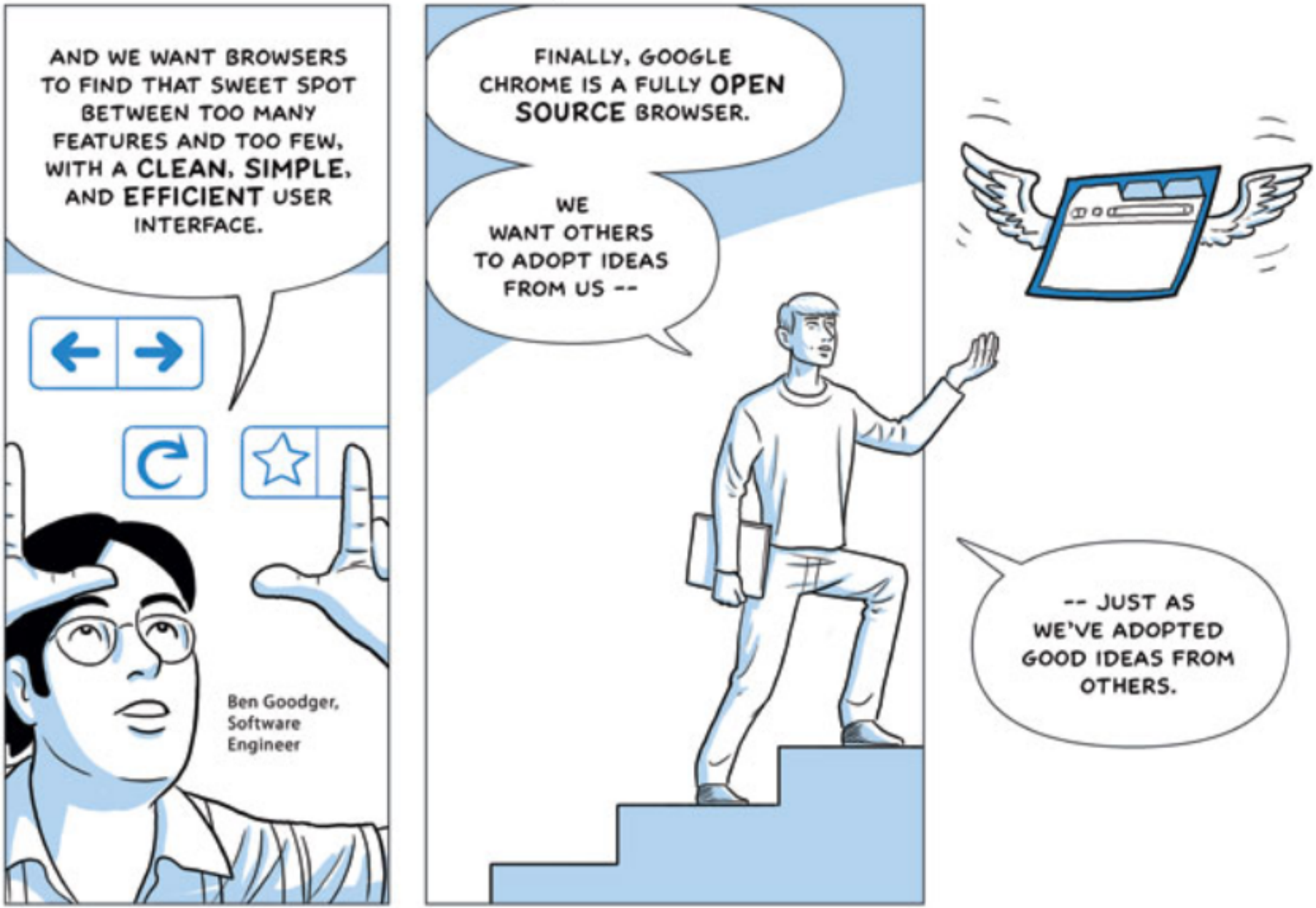
Google

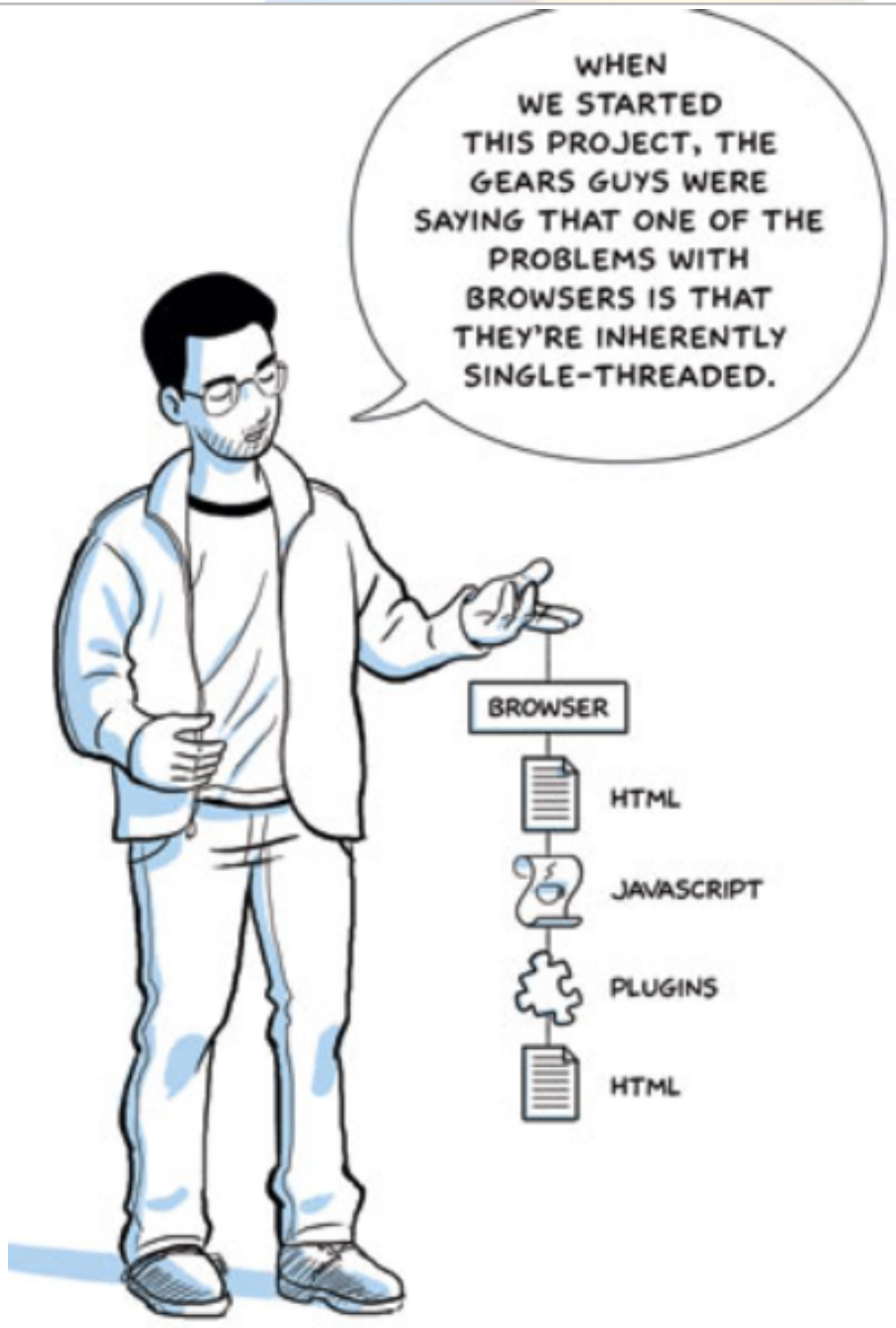


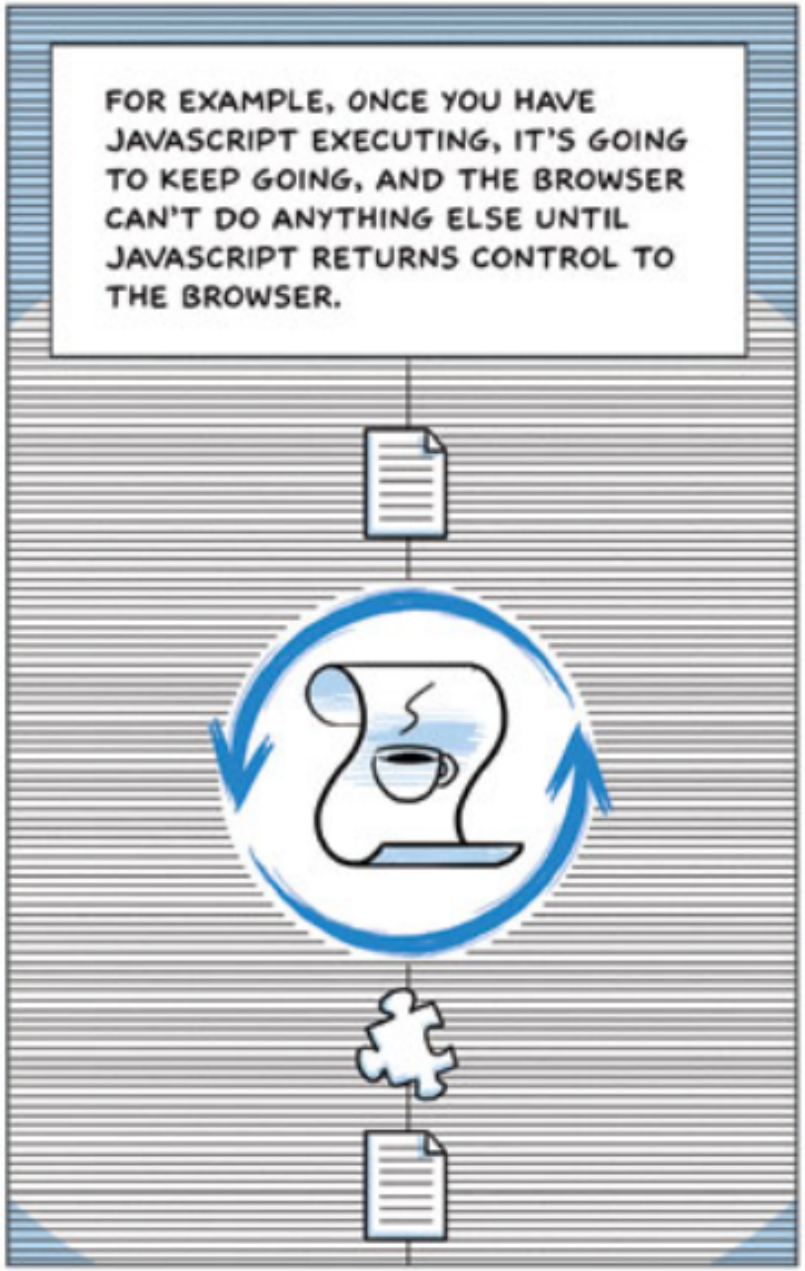

PEOPLE ARE WATCHING AND UPLOADING VIDEOS, CHATTING WITH EACH OTHER, PLAYING WEB-BASED GAMES...
ALL THESE THINGS THAT DIDN'T EXIST WHEN THE FIRST BROWSERS WERE CREATED.
Azula88: ROFL ^^
DctrHrrbl: LOL
Trg: OMFG
Pam Greene, Software Engineer

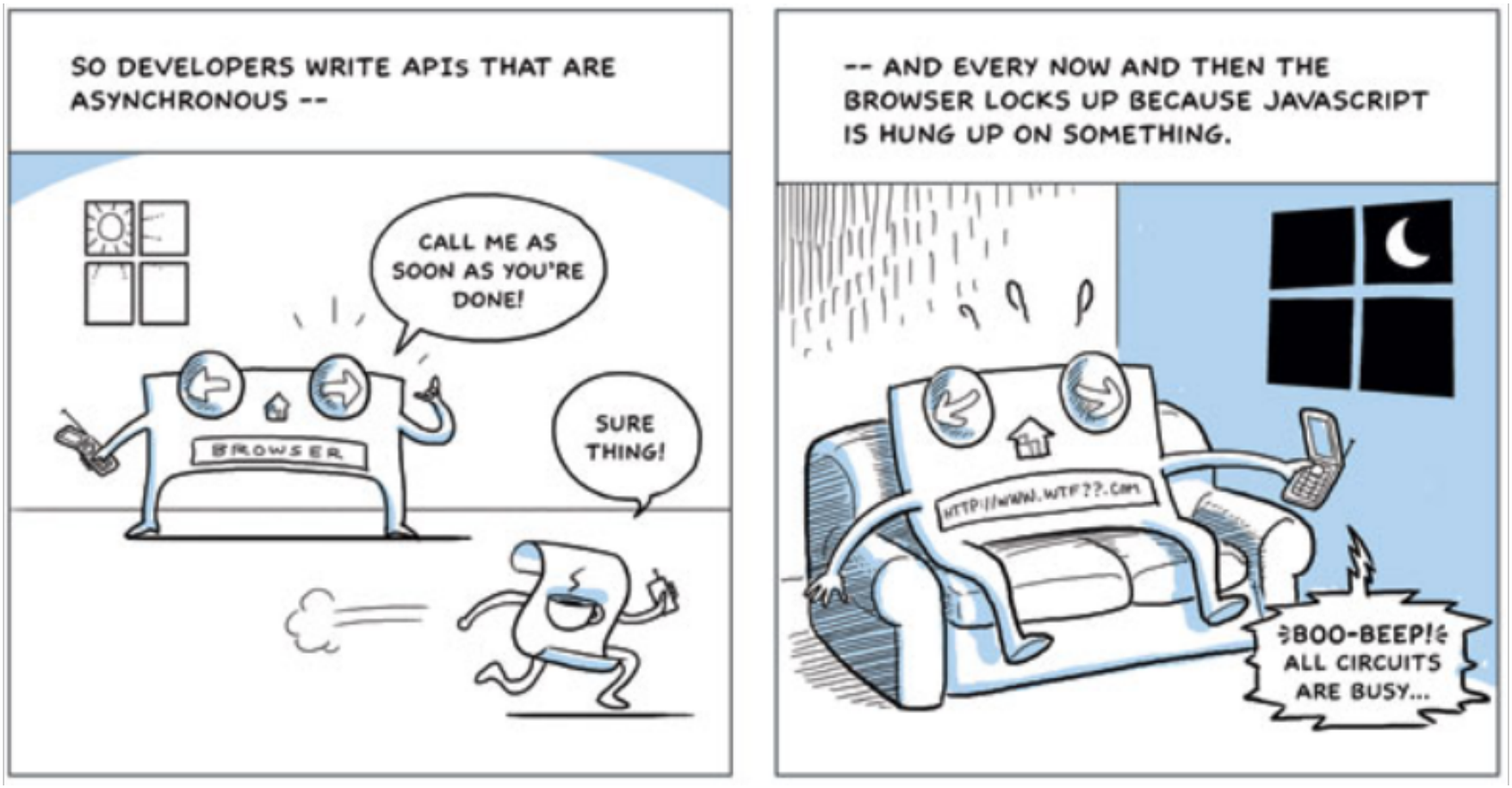

WOULDN'T IT BE GREAT, THEN, TO START FROM SCRATCH --
-- AND DESIGN SOMETHING BASED ON THE NEEDS OF TODAY'S WEB APPLICATIONS AND TODAY'S USERS?
2008

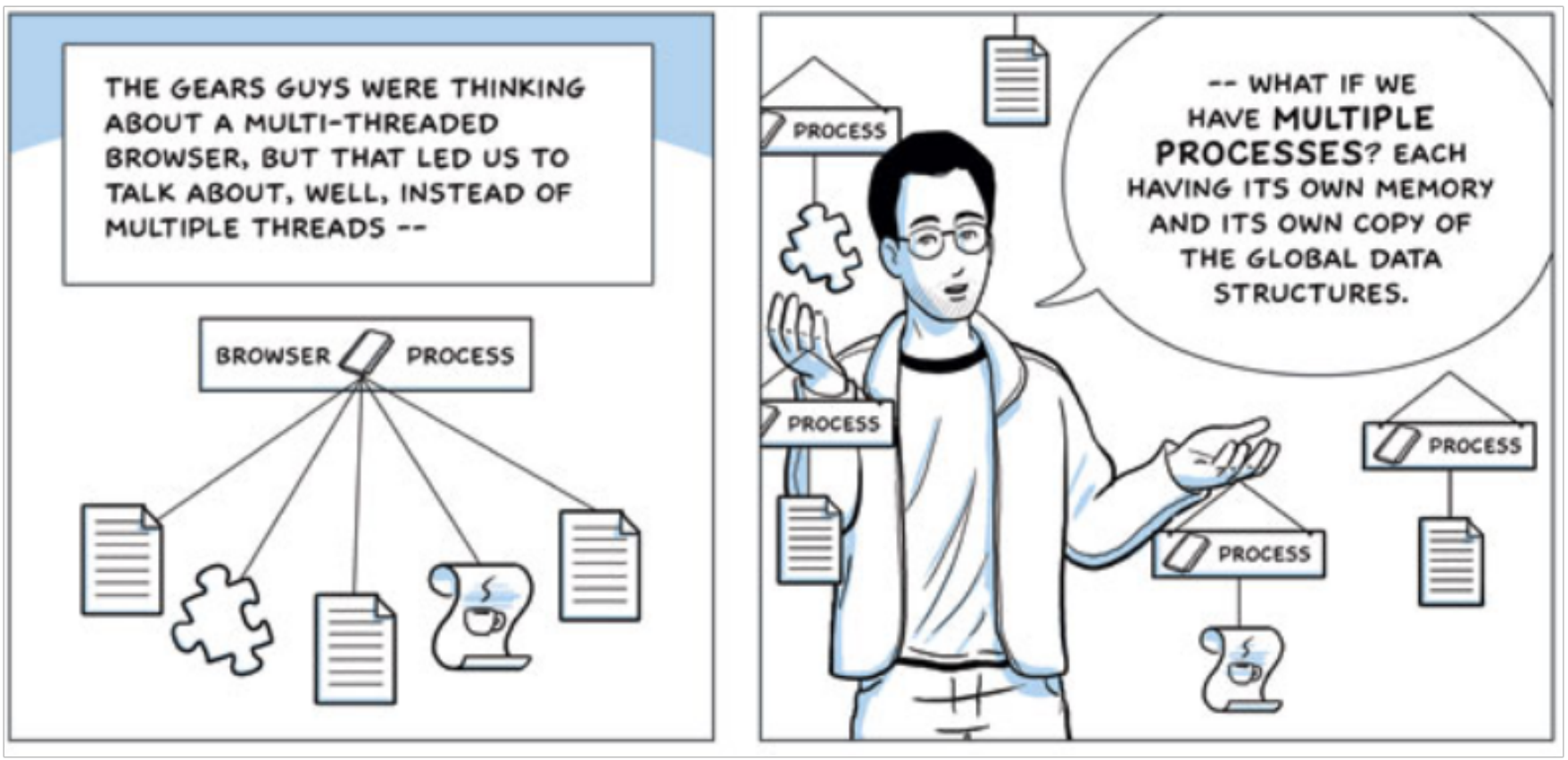

FIRST, BROWSERS NEED TO BE MORE STABLE. WHEN YOU'RE WRITING AN IMPORTANT EMAIL OR EDITING A DOCUMENT, A BROWSER CRASH IS A BIG DEAL.
Darin Fisher, Software Engineer
BROWSERS ALSO NEED TO BE FASTER. THEY NEED TO START FASTER, LOAD PAGES FASTER --
-- AND FOR WEB APPS, JAVASCRIPT ITSELF CAN BE A LOT FASTER.
Lars Bak, Software Engineer
Kasper Lund, Software Engineer
THEY NEED TO BE MORE SECURE. GIVEN WHAT'S KNOWN ABOUT MASS BROWSER EXPLOITS, BROWSERS NEED ARCHITECTURAL CHANGES TO DISADVANTAGE MALWARE.
Ian Fette, Product Manager


AND WE WANT BROWSERS TO FIND THAT SWEET SPOT BETWEEN TOO MANY FEATURES AND TOO FEW, WITH A CLEAN, SIMPLE, AND EFFICIENT USER INTERFACE.
Ben Goodger, Software Engineer
FINALLY, GOOGLE CHROME IS A FULLY OPEN SOURCE BROWSER.
WE WANT OTHERS TO ADOPT IDEAS FROM US --
-- JUST AS WE'VE ADOPTED GOOD IDEAS FROM OTHERS.

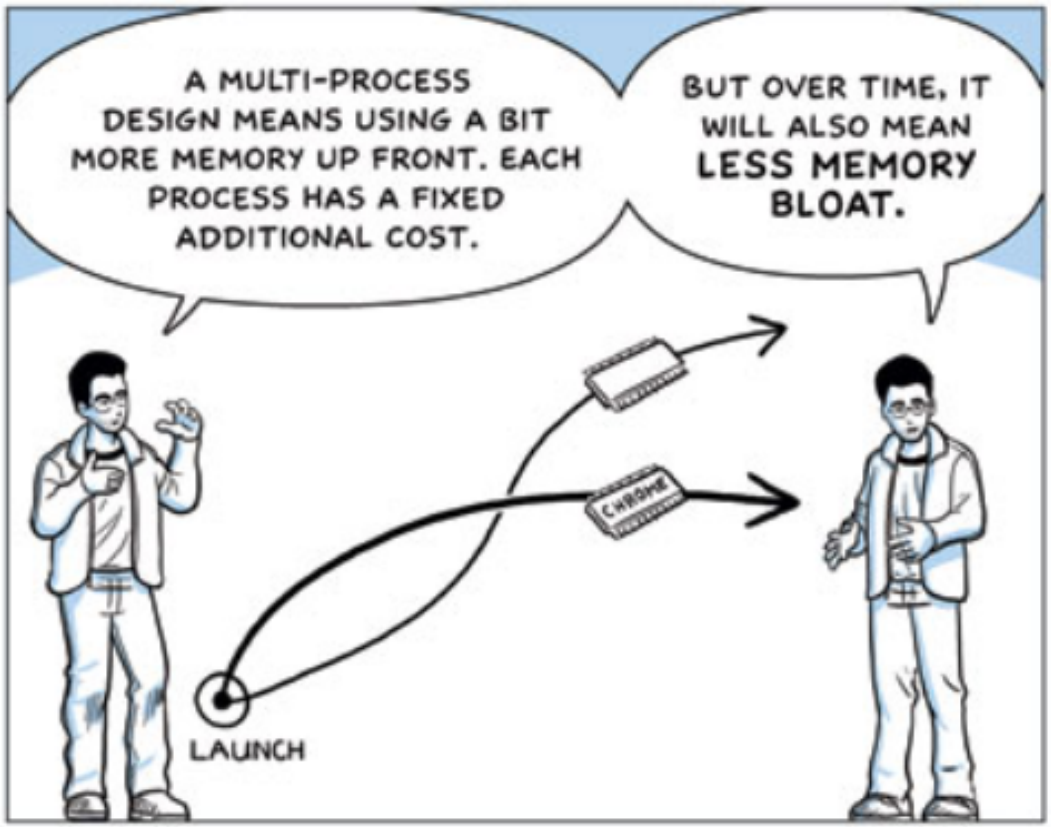

WHEN WE STARTED THIS PROJECT, THE GEARS GUYS WERE SAYING THAT ONE OF THE PROBLEMS WITH BROWSERS IS THAT THEY'RE INHERENTLY SINGLE-THREADED.
BROWSER
HTML
JAVASCRIPT
PLUGINS
HTML

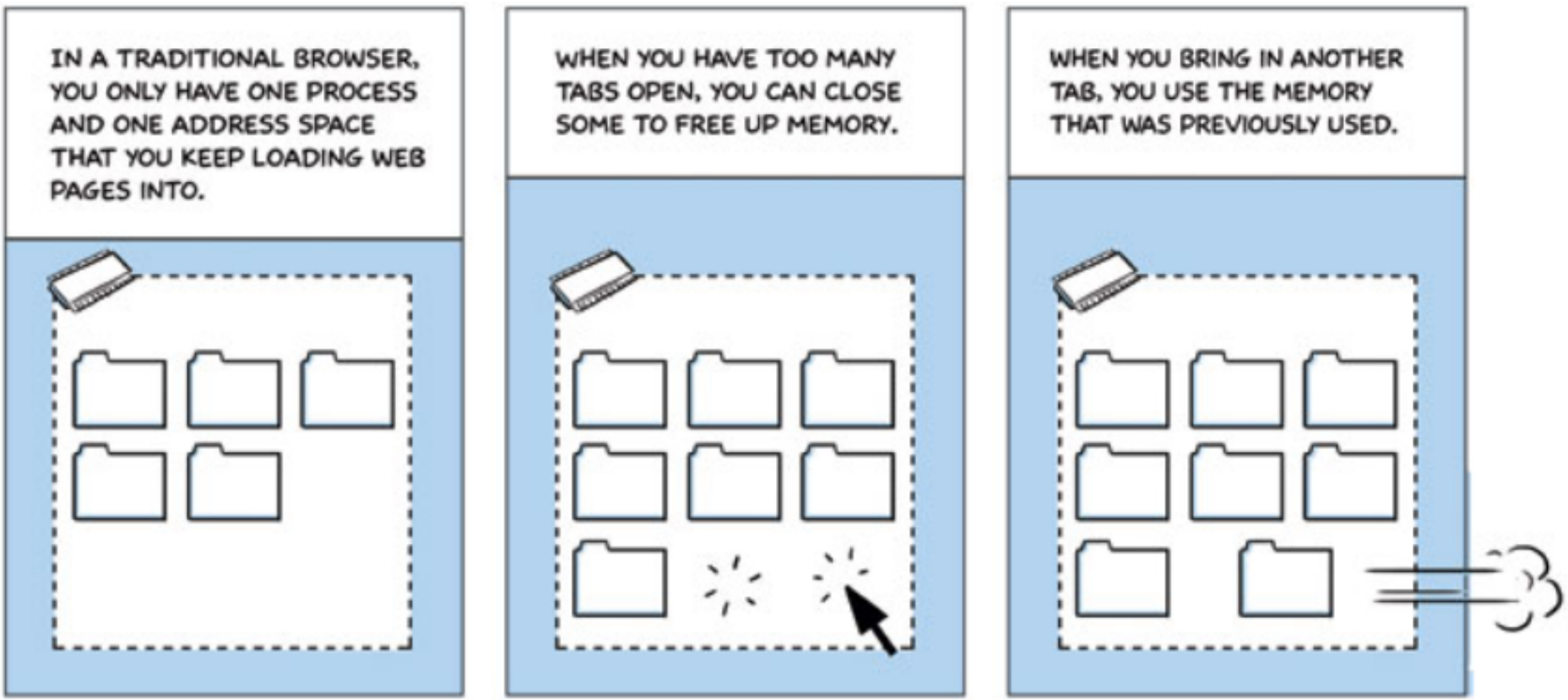

FOR EXAMPLE, ONCE YOU HAVE JAVASCRIPT EXECUTING, IT'S GOING TO KEEP GOING, AND THE BROWSER CAN'T DO ANYTHING ELSE UNTIL JAVASCRIPT RETURNS CONTROL TO THE BROWSER.

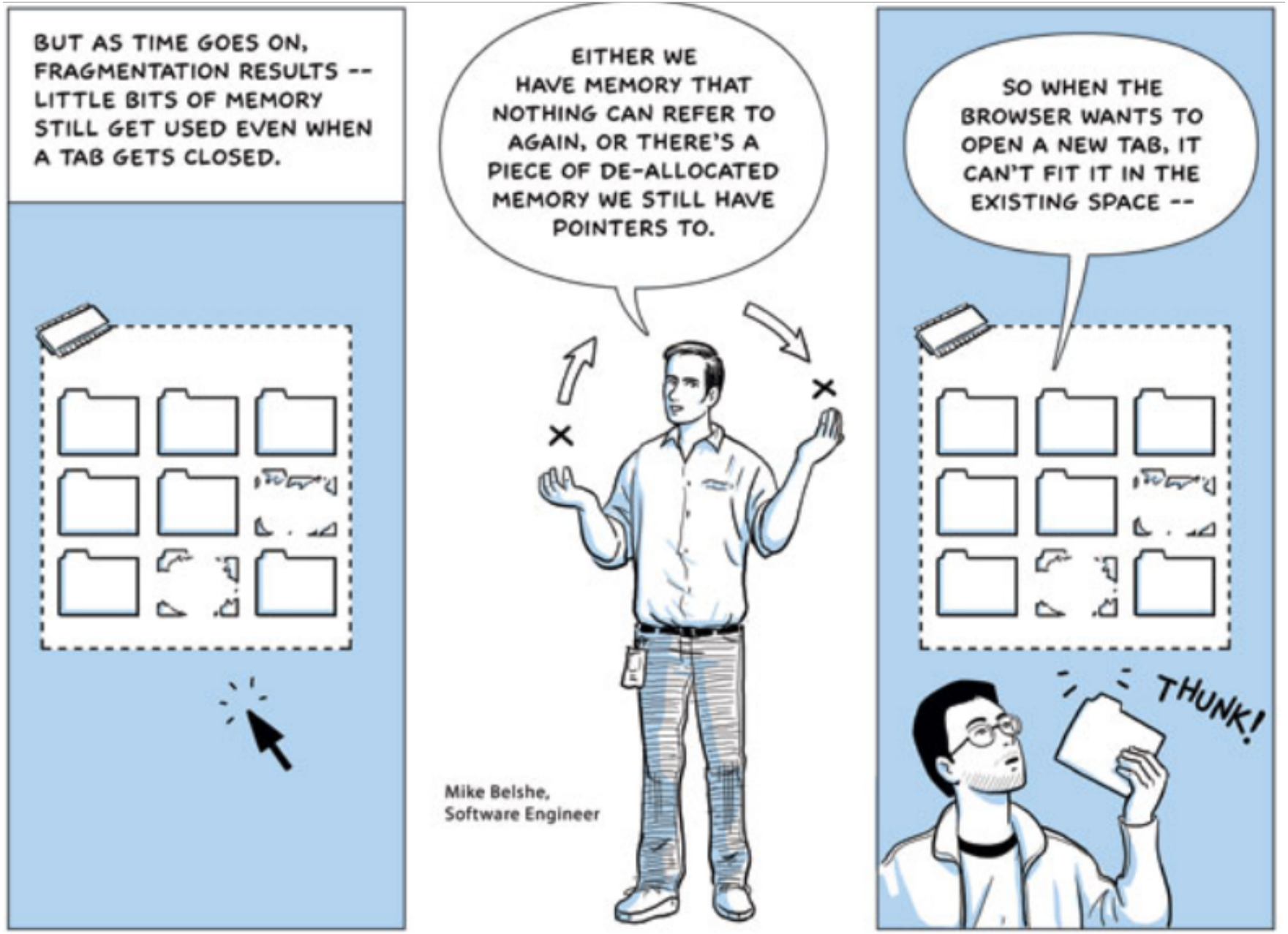

SO DEVELOPERS WRITE APIs THAT ARE ASYNCHRONOUS --
CALL ME AS SOON AS YOU'RE DONE!
BROWSER
SURE THING!
-- AND EVERY NOW AND THEN THE BROWSER LOCKS UP BECAUSE JAVASCRIPT IS HUNG UP ON SOMETHING.
HTTP://WWW.WTF??.COM
BOO-BEEP! ALL CIRCUITS ARE BUSY...

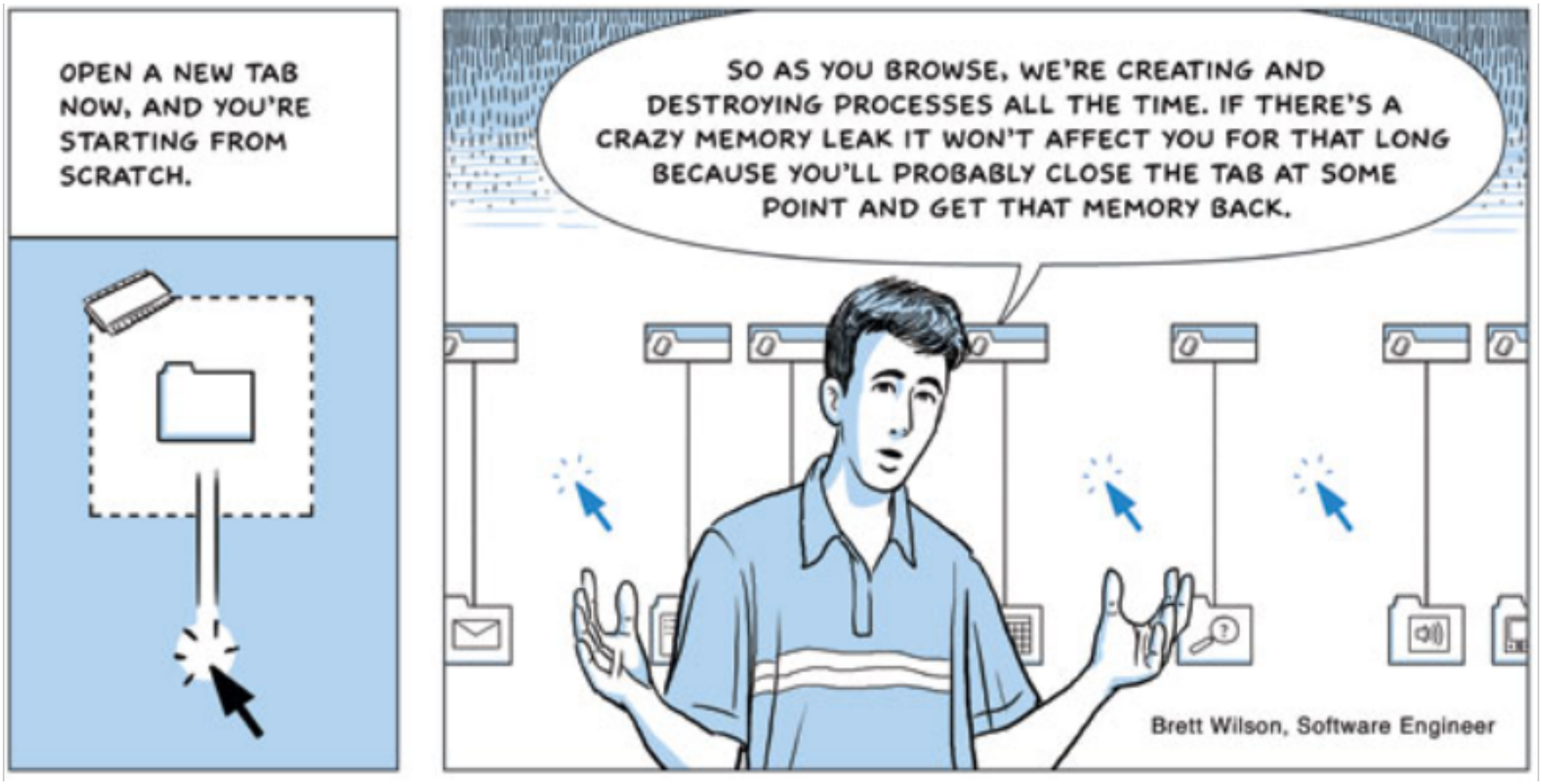

THE GEARS GUYS WERE THINKING ABOUT A MULTI-THREADED BROWSER, BUT THAT LED US TO TALK ABOUT, WELL, INSTEAD OF MULTIPLE THREADS --
BROWSER PROCESS
-- WHAT IF WE HAVE MULTIPLE PROCESSES? EACH HAVING ITS OWN MEMORY AND ITS OWN COPY OF THE GLOBAL DATA STRUCTURES.
PROCESS
PROCESS
PROCESS
PROCESS

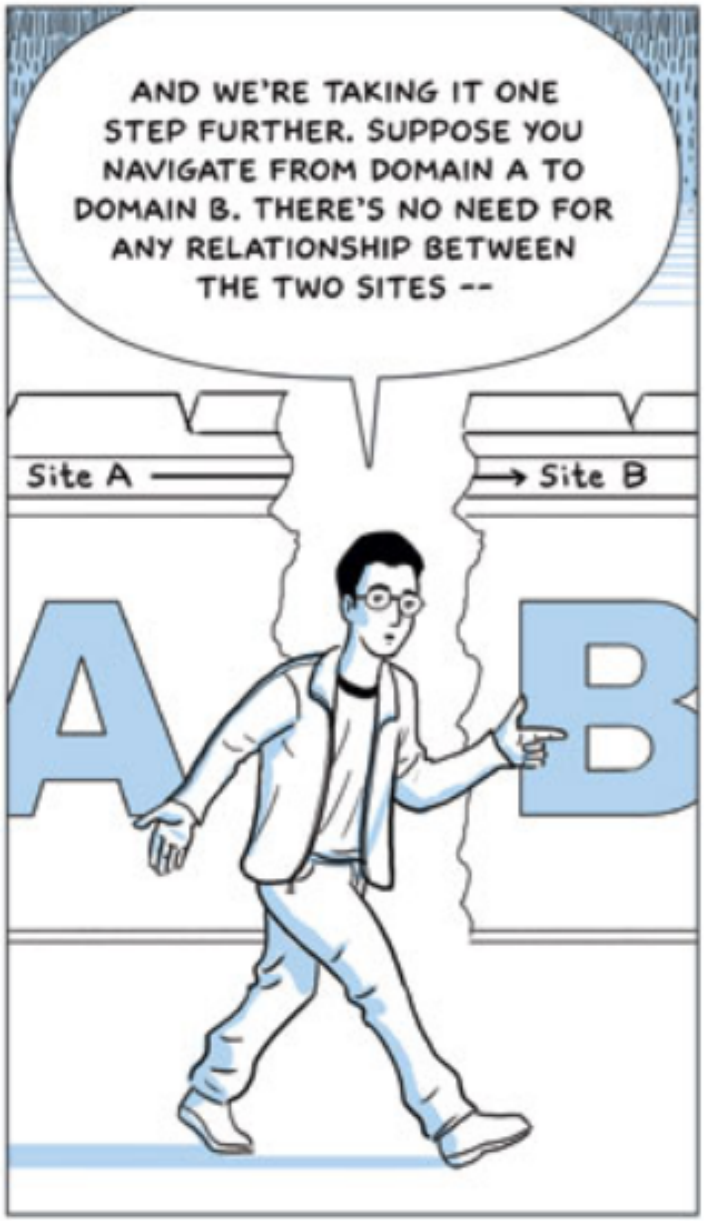

WHEN ONE TAB GOES DOWN YOU GET A 'SAD TAB' BUT IT DOESN'T CRASH THE WHOLE BROWSER.
AND YES, IT REALLY LOOKS LIKE THIS.

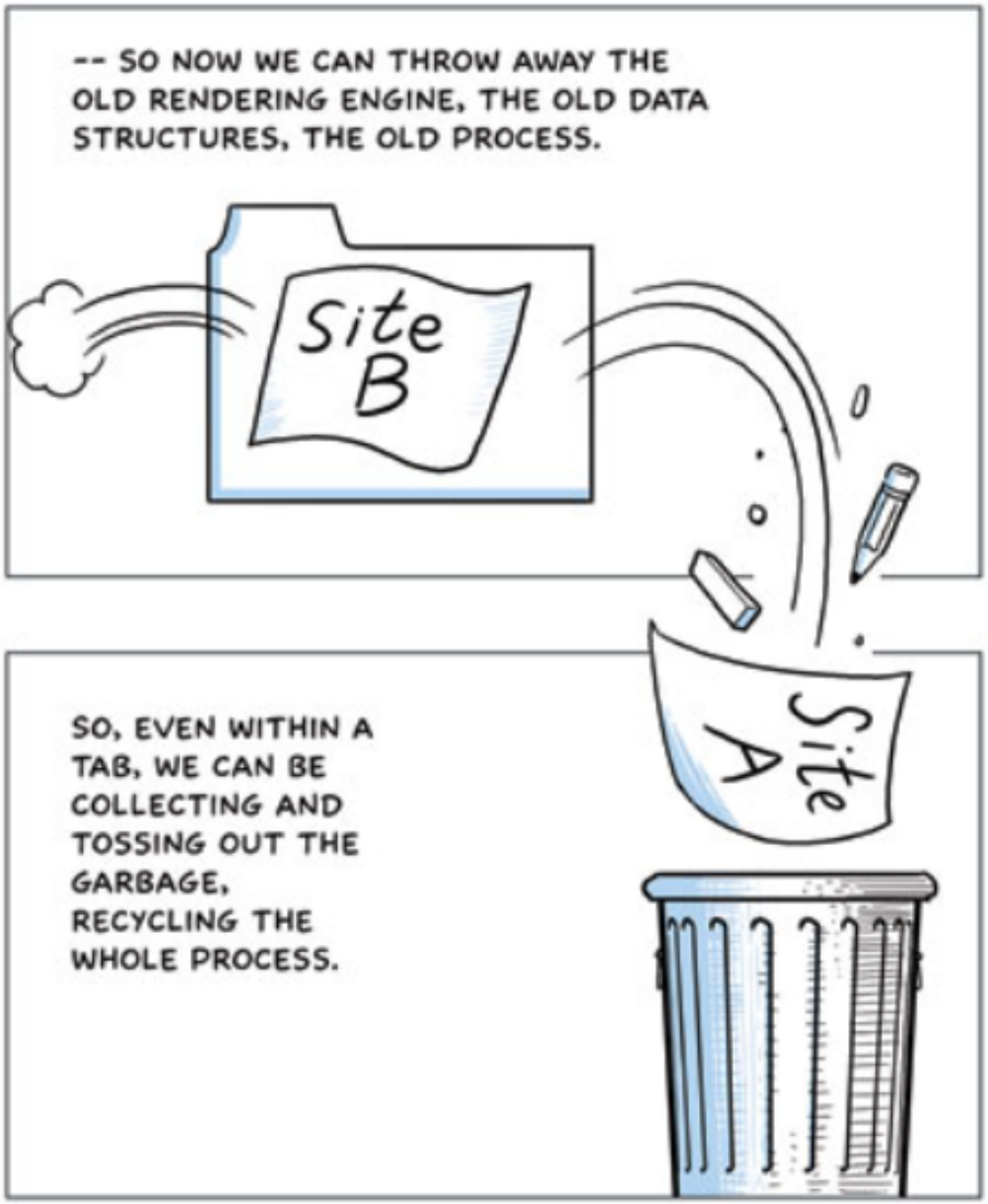

A MULTI-PROCESS DESIGN MEANS USING A BIT MORE MEMORY UP FRONT. EACH PROCESS HAS A FIXED ADDITIONAL COST.
BUT OVER TIME, IT WILL ALSO MEAN LESS MEMORY BLOAT.
CHROME
LAUNCH


IN A TRADITIONAL BROWSER, YOU ONLY HAVE ONE PROCESS AND ONE ADDRESS SPACE THAT YOU KEEP LOADING WEB PAGES INTO.
WHEN YOU HAVE TOO MANY TABS OPEN, YOU CAN CLOSE SOME TO FREE UP MEMORY.
WHEN YOU BRING IN ANOTHER TAB, YOU USE THE MEMORY THAT WAS PREVIOUSLY USED.


BUT AS TIME GOES ON, FRAGMENTATION RESULTS -- LITTLE BITS OF MEMORY STILL GET USED EVEN WHEN A TAB GETS CLOSED.
EITHER WE HAVE MEMORY THAT NOTHING CAN REFER TO AGAIN, OR THERE'S A PIECE OF DE-ALLOCATED MEMORY WE STILL HAVE POINTERS TO.
Mike Belshe, Software Engineer
SO WHEN THE BROWSER WANTS TO OPEN A NEW TAB, IT CAN'T FIT IT IN THE EXISTING SPACE --
THUNK!


OPEN A NEW TAB NOW, AND YOU'RE STARTING FROM SCRATCH.
SO AS YOU BROWSE, WE'RE CREATING AND DESTROYING PROCESSES ALL THE TIME. IF THERE'S A CRAZY MEMORY LEAK IT WON'T AFFECT YOU FOR THAT LONG BECAUSE YOU'LL PROBABLY CLOSE THE TAB AT SOME POINT AND GET THAT MEMORY BACK.
Brett Wilson, Software Engineer


AND WE'RE TAKING IT ONE STEP FURTHER. SUPPOSE YOU NAVIGATE FROM DOMAIN A TO DOMAIN B. THERE'S NO NEED FOR ANY RELATIONSHIP BETWEEN THE TWO SITES --
Site A
Site B
A
B


-- SO NOW WE CAN THROW AWAY THE OLD RENDERING ENGINE, THE OLD DATA STRUCTURES, THE OLD PROCESS.
Site B
SO, EVEN WITHIN A TAB, WE CAN BE COLLECTING AND TOSSING OUT THE GARBAGE, RECYCLING THE WHOLE PROCESS.
Site A

More about Security: Design Principles

Google

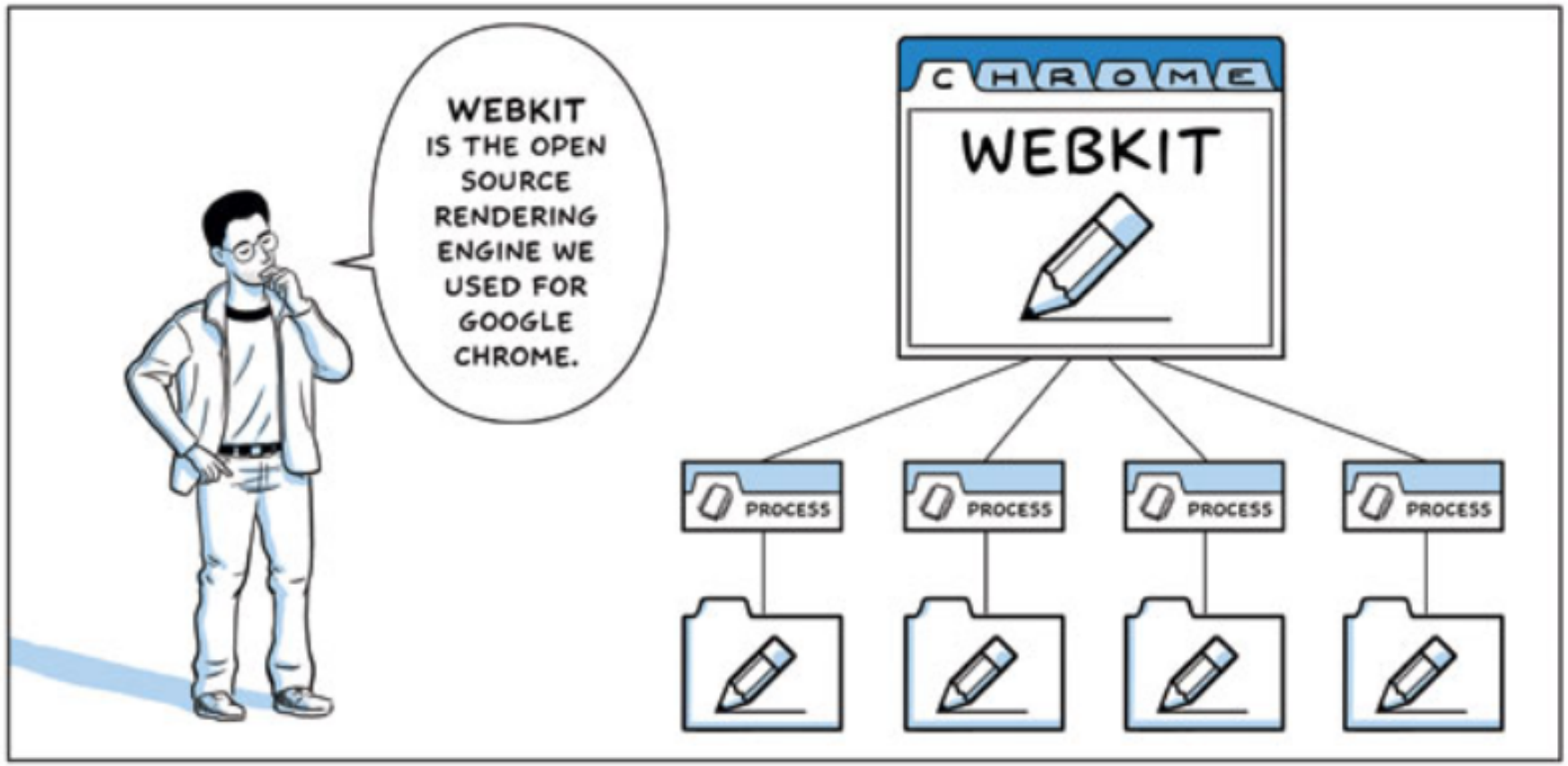
- Do not re-invent the wheel
- Principle of least privilege
- Sandboxed code is malicious code
- Be lightweight
- Emulation doesn’t guarantee security

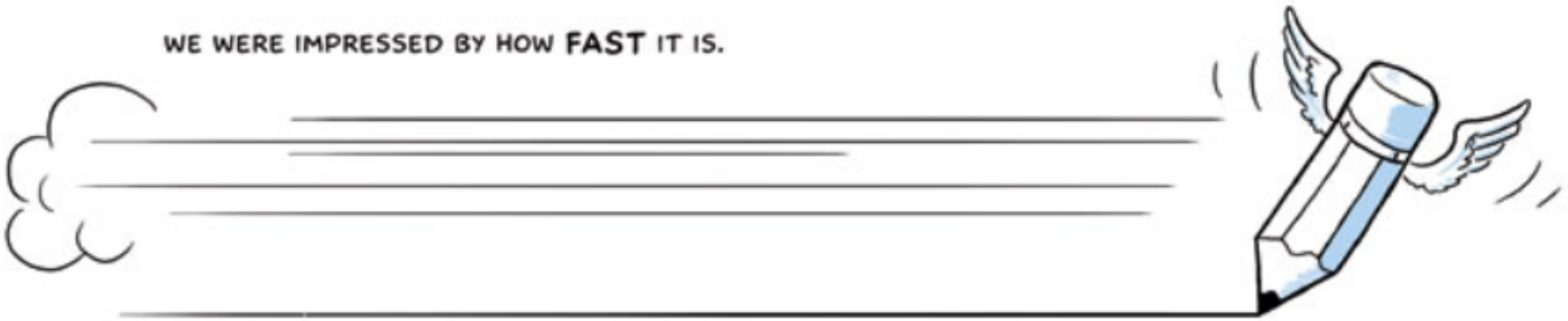
Testing

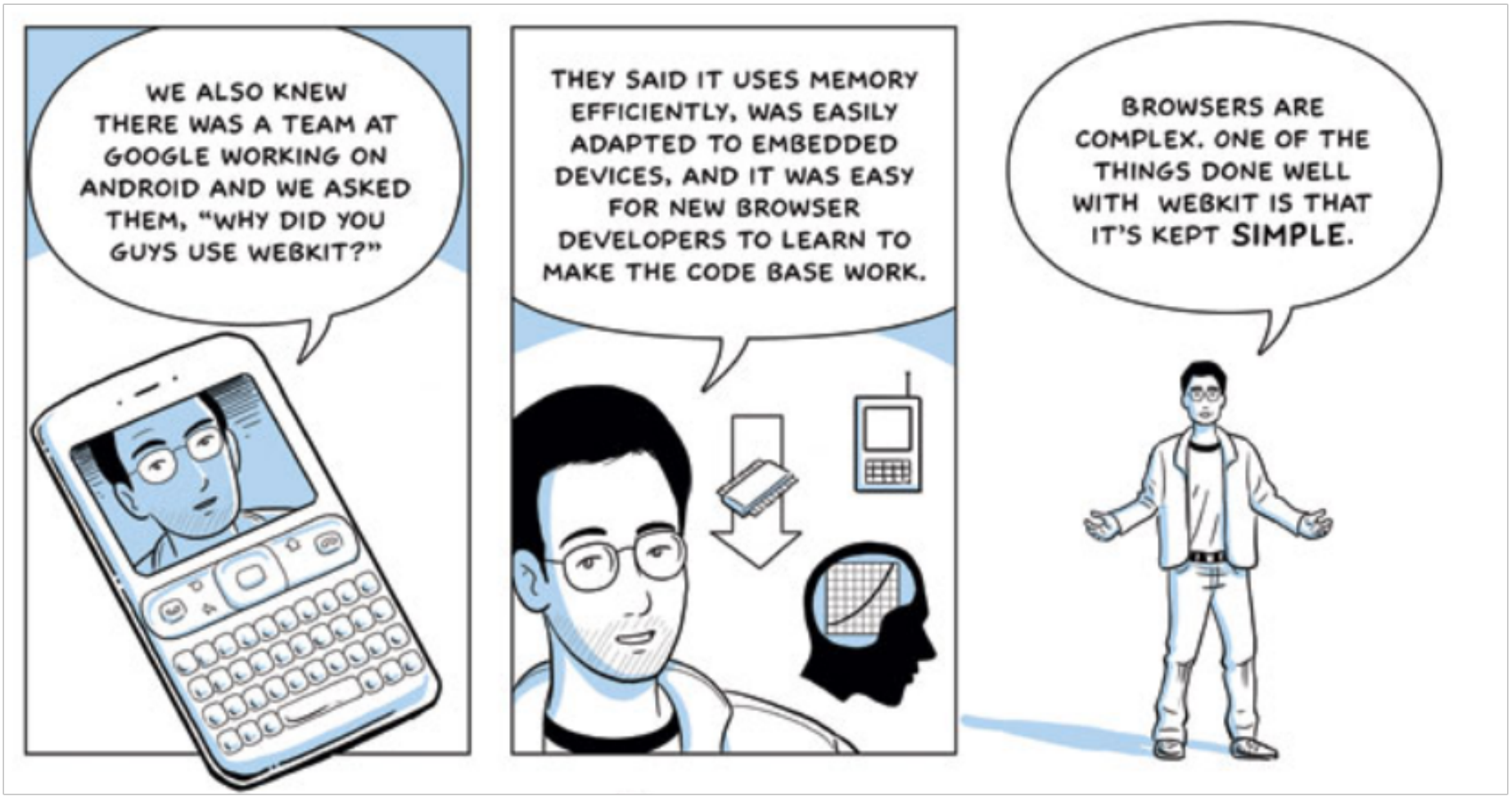
Google

Webkit

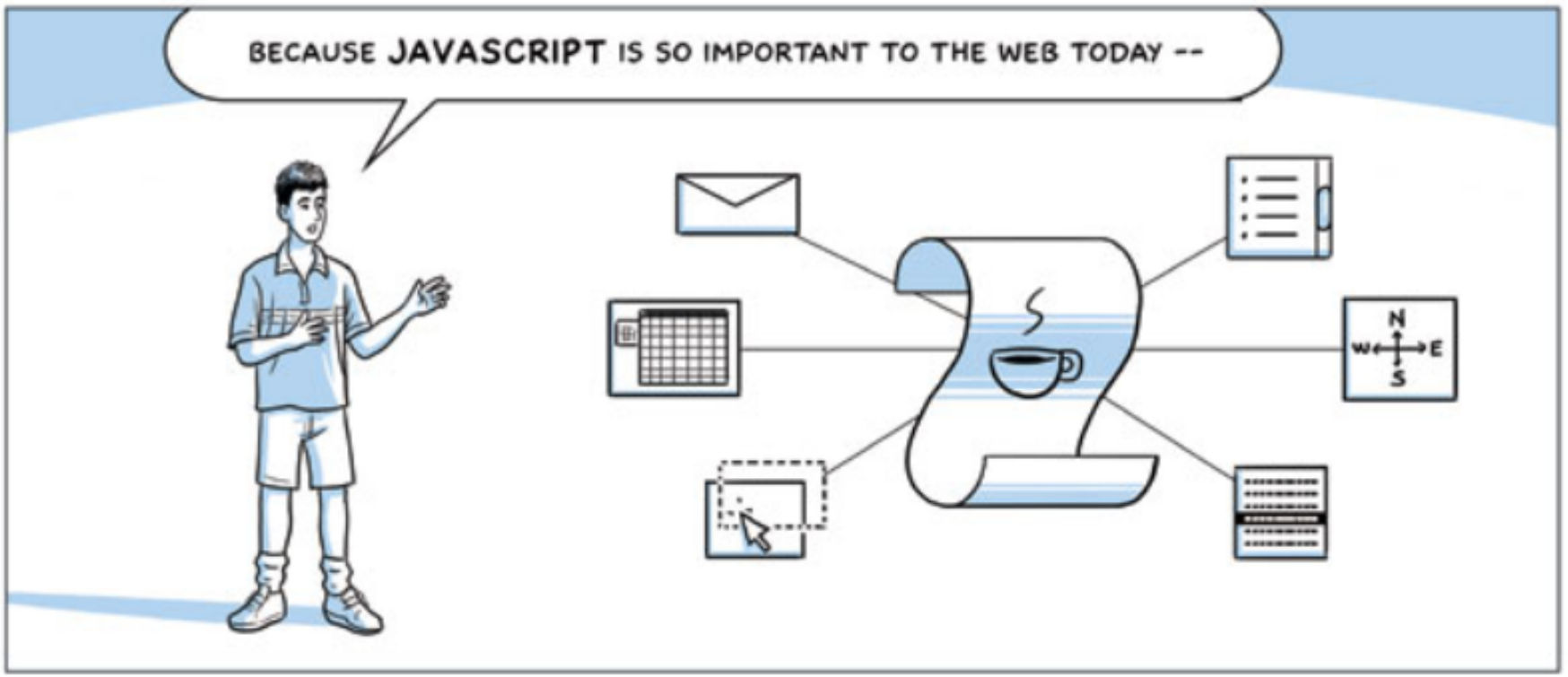
Google

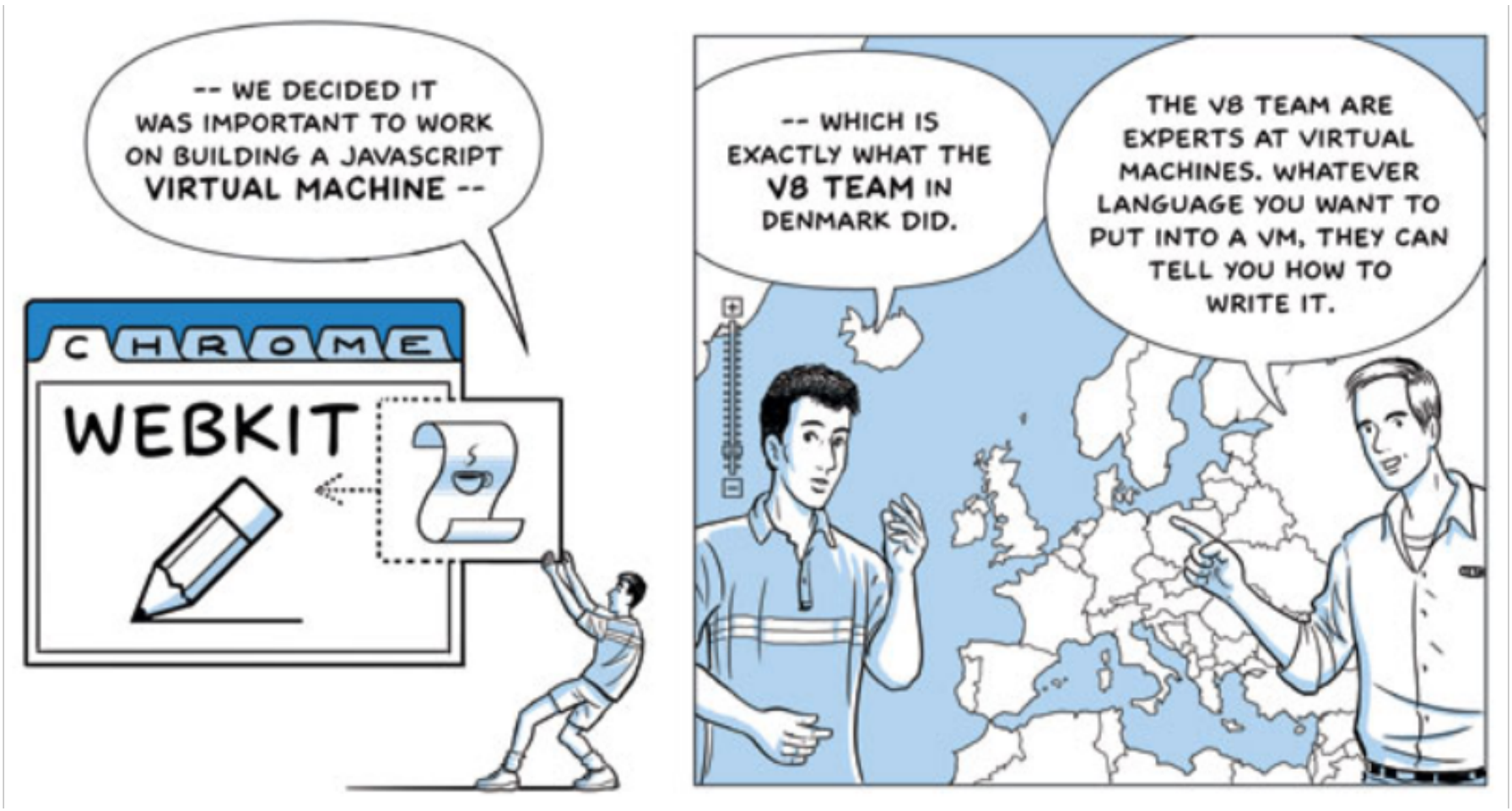





WE ALSO KNEW THERE WAS A TEAM AT GOOGLE WORKING ON ANDROID AND WE ASKED THEM, "WHY DID YOU GUYS USE WEBKIT?"
THEY SAID IT USES MEMORY EFFICIENTLY, WAS EASILY ADAPTED TO EMBEDDED DEVICES, AND IT WAS EASY FOR NEW BROWSER DEVELOPERS TO LEARN TO MAKE THE CODE BASE WORK.
BROWSERS ARE COMPLEX. ONE OF THE THINGS DONE WELL WITH WEBKIT IS THAT IT'S KEPT SIMPLE.

V8 Javascript VM

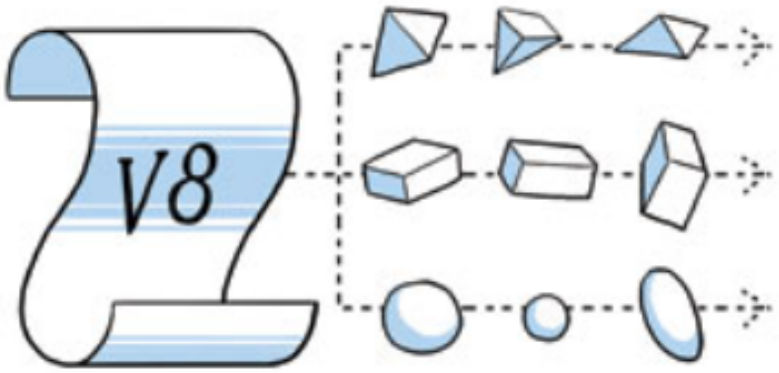




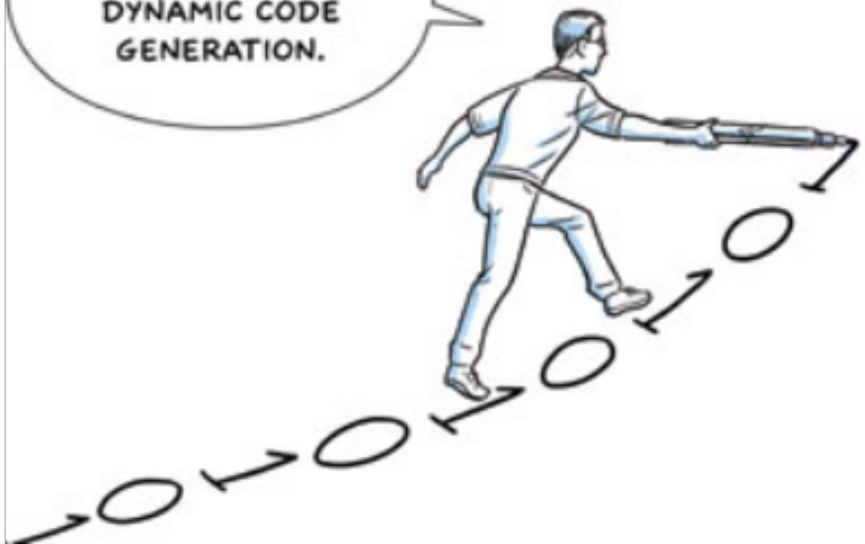
WHEN YOU INTERPRET ONCE AND COMPILE MACHINE CODE, THEN THAT CODE **IS** YOUR REPRESENTATION OF THE JAVASCRIPT SOURCE CODE AND IT DOESN'T NEED TO BE INTERPRETED, IT JUST **RUNS**.

JAVASCRIPT ITSELF IS **CLASSLESS**. YOU CAN CREATE A NEW OBJECT, DYNAMICALLY ADD PROPERTIES TO IT AND GO ON.

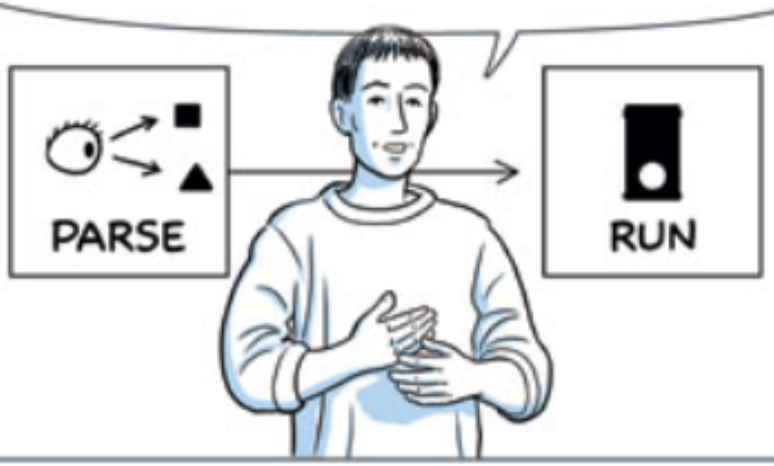
BUT IN **V8**, AS EXECUTION GOES ON, OBJECTS THAT END UP WITH THE SAME PROPERTIES WILL SHARE THE SAME HIDDEN CLASS AND WE CAN START APPLYING DYNAMIC OPTIMIZATIONS BASED ON THAT.

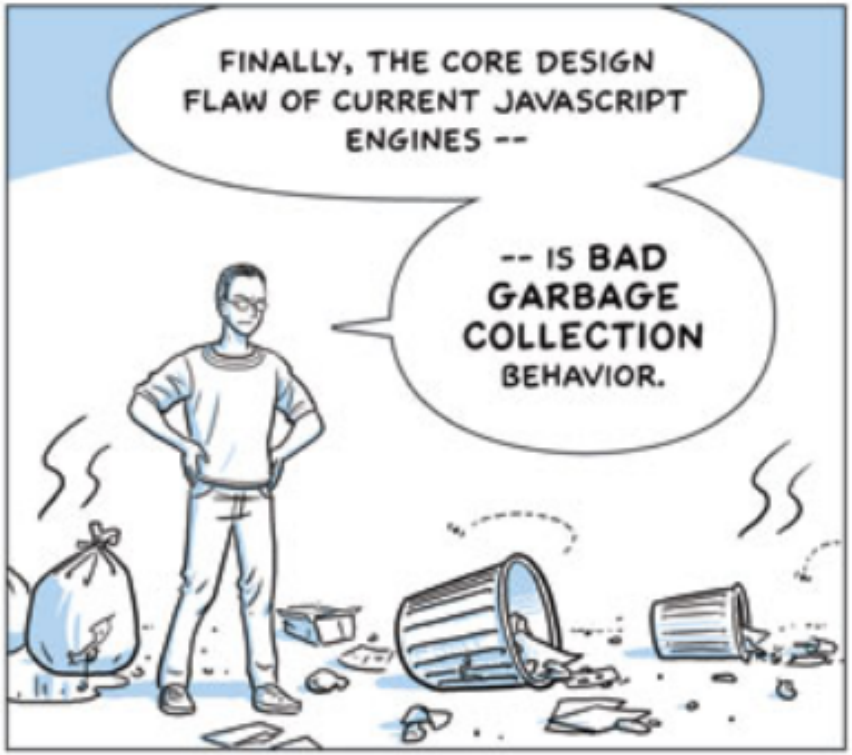


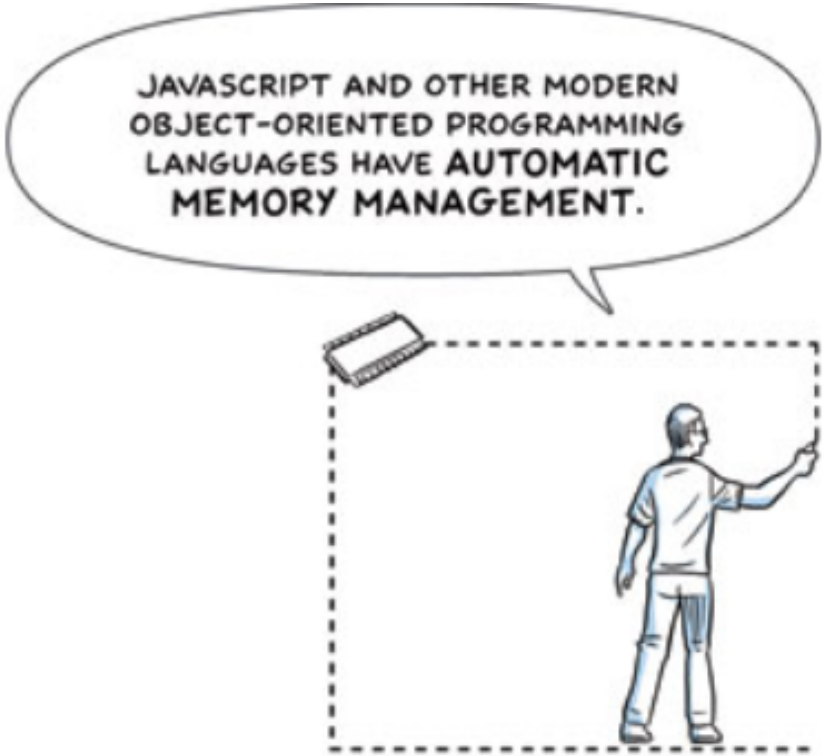
ANOTHER FACTOR IN V8'S SPEED IS DYNAMIC CODE GENERATION.

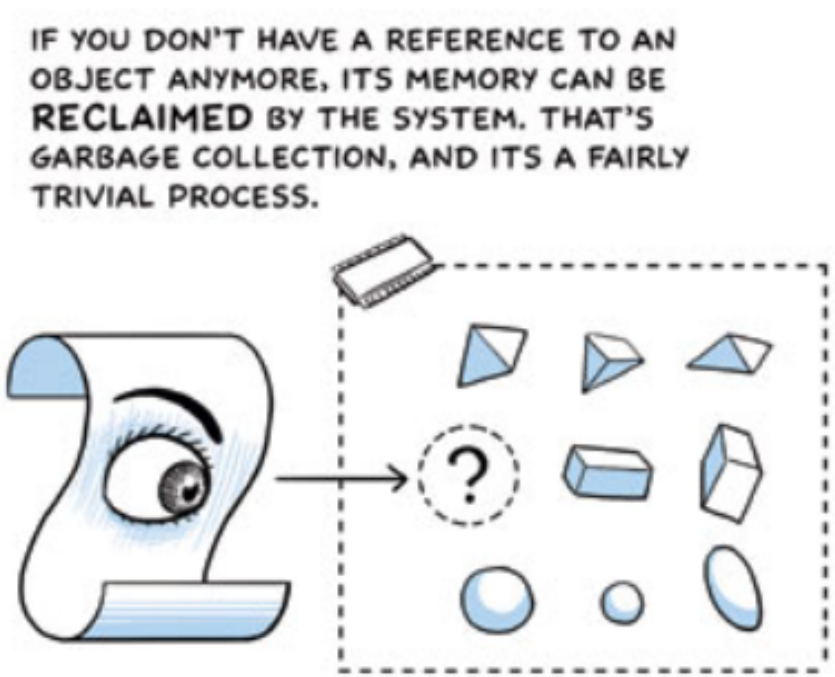


WHEN OTHER JAVASCRIPT ENGINES RUN, THEY LOOK AT THE JAVASCRIPT SOURCE CODE AND GENERATE AN INTERNAL REPRESENTATION OF IT THEY CAN INTERPRET.

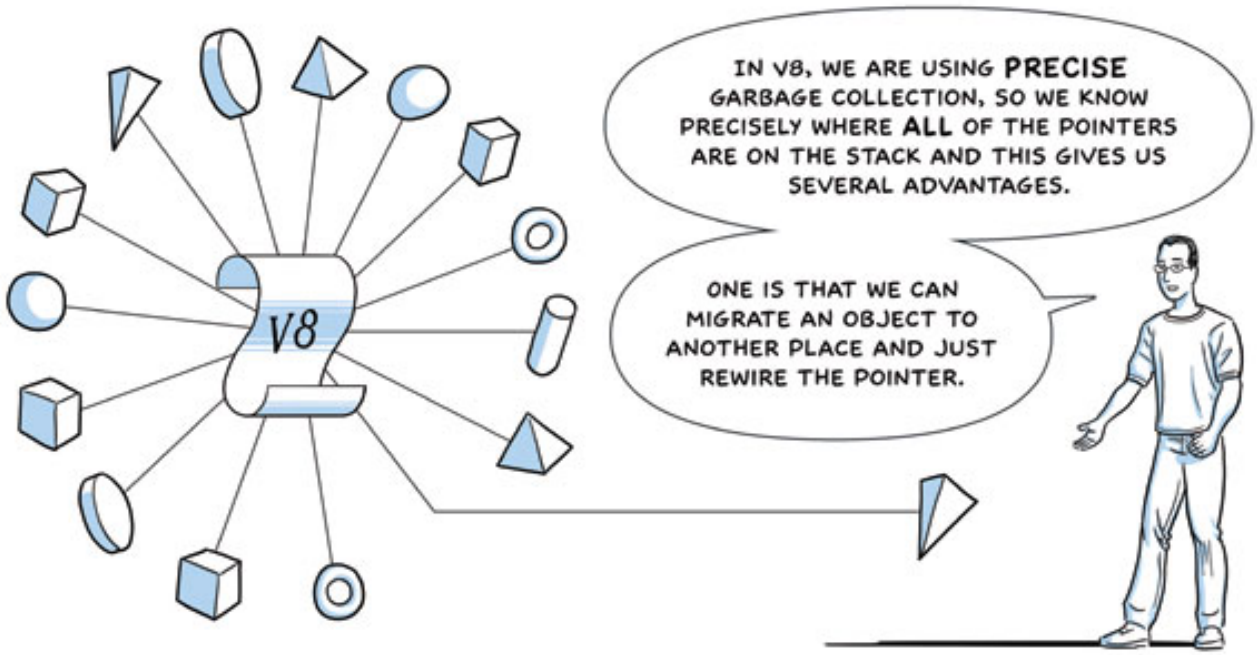


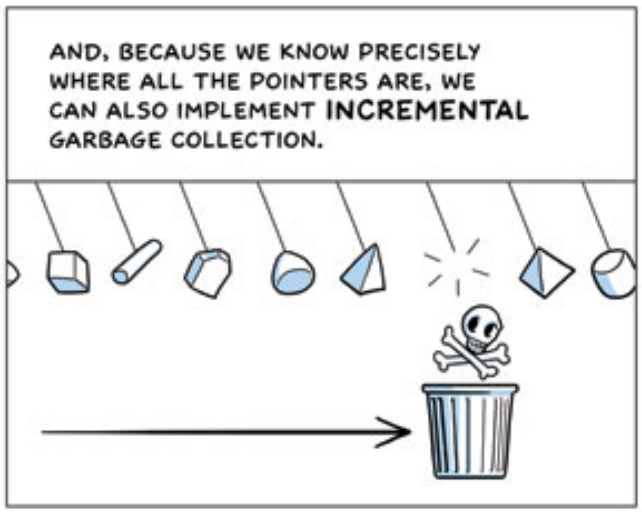

FINALLY, THE CORE DESIGN FLAW OF CURRENT JAVASCRIPT ENGINES --
-- IS BAD GARBAGE COLLECTION BEHAVIOR.


JAVASCRIPT AND OTHER MODERN OBJECT-ORIENTED PROGRAMMING LANGUAGES HAVE AUTOMATIC MEMORY MANAGEMENT.

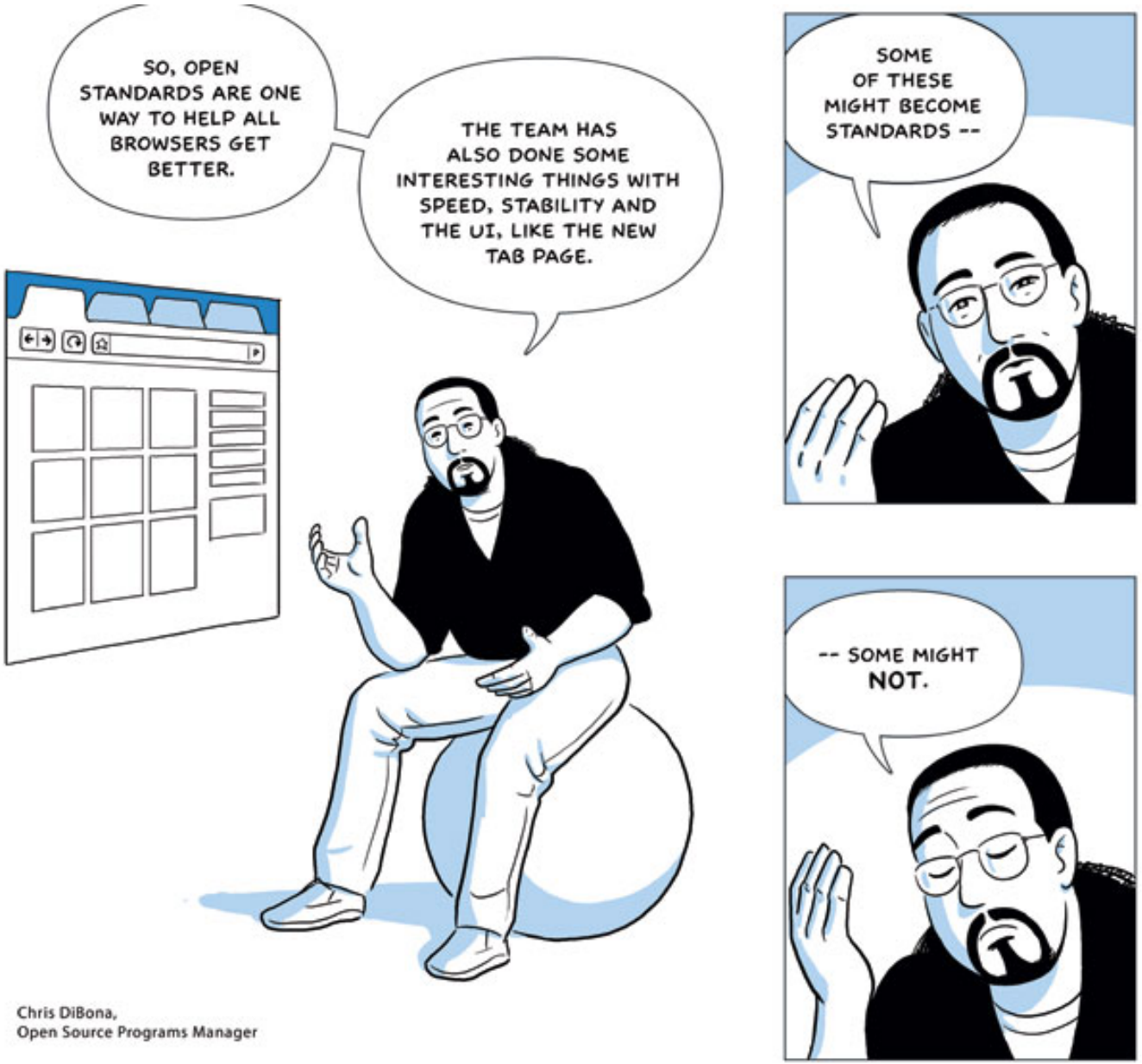

IF YOU DON'T HAVE A REFERENCE TO AN OBJECT ANYMORE, ITS MEMORY CAN BE RECLAIMED BY THE SYSTEM. THAT'S GARBAGE COLLECTION, AND ITS A FAIRLY TRIVIAL PROCESS.
?


BUT IN EXISTING JAVASCRIPT VIRTUAL MACHINES, THEY USE CONSERVATIVE GARBAGE COLLECTION --





MEANING QUICK GARBAGE COLLECTION ROUND-TRIPS THAT ARE CLOSE TO A FEW MILLISECONDS, COMPARED TO PROCESSING ALL 100MB OF DATA WHICH COULD CAUSE SECOND-LONG PAUSES.

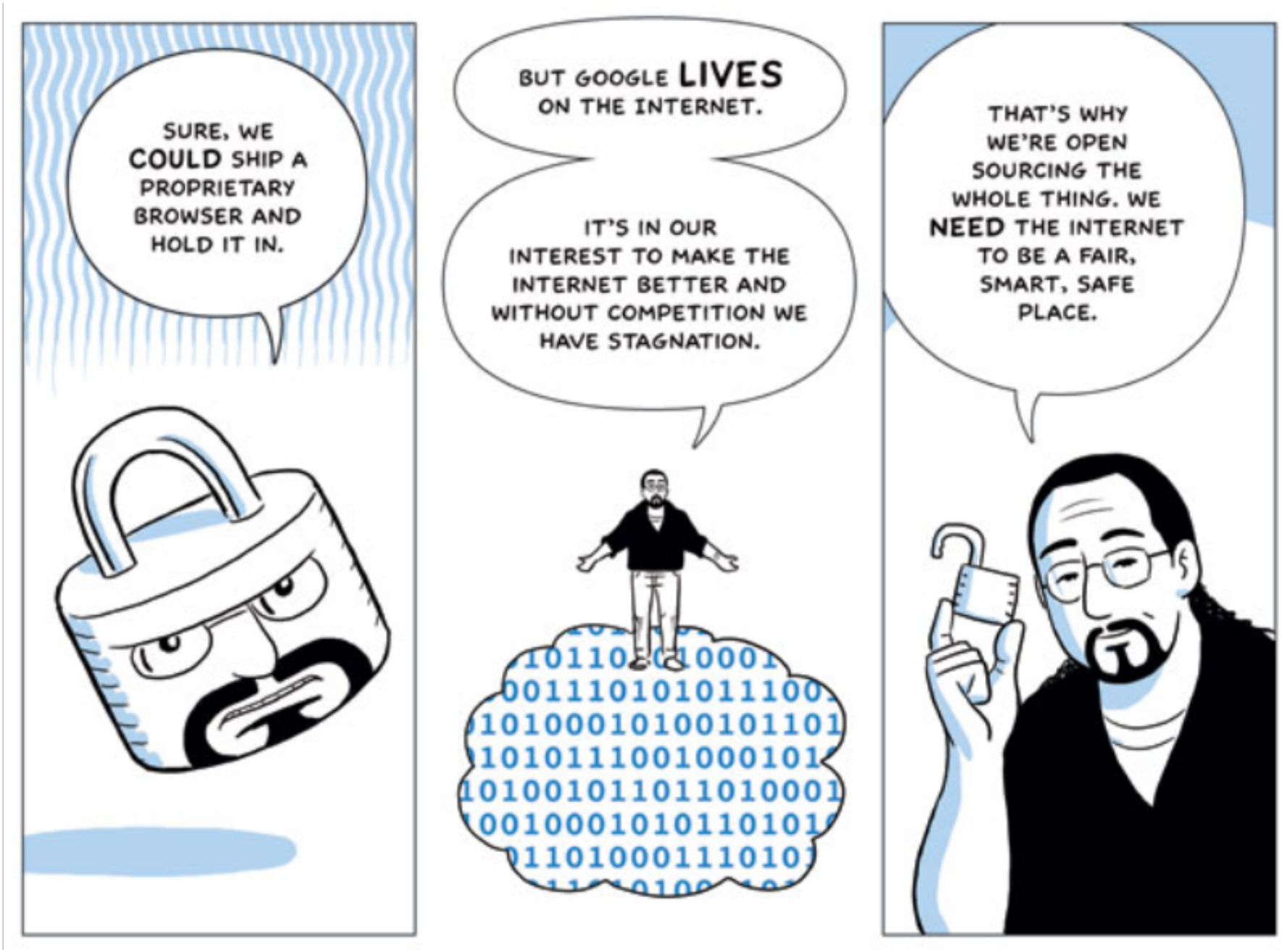


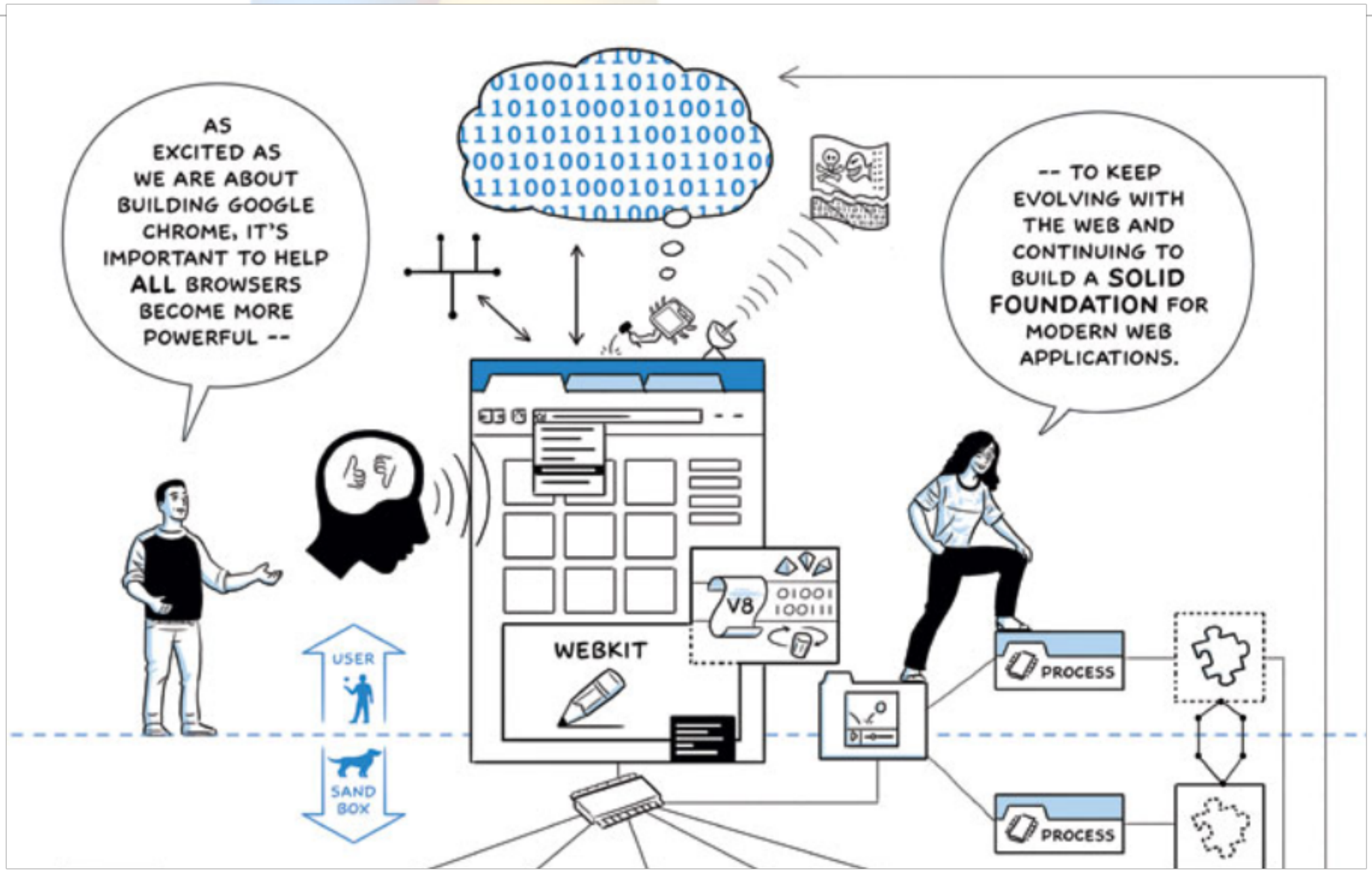
Chris DiBona,
Open Source Programs Manager

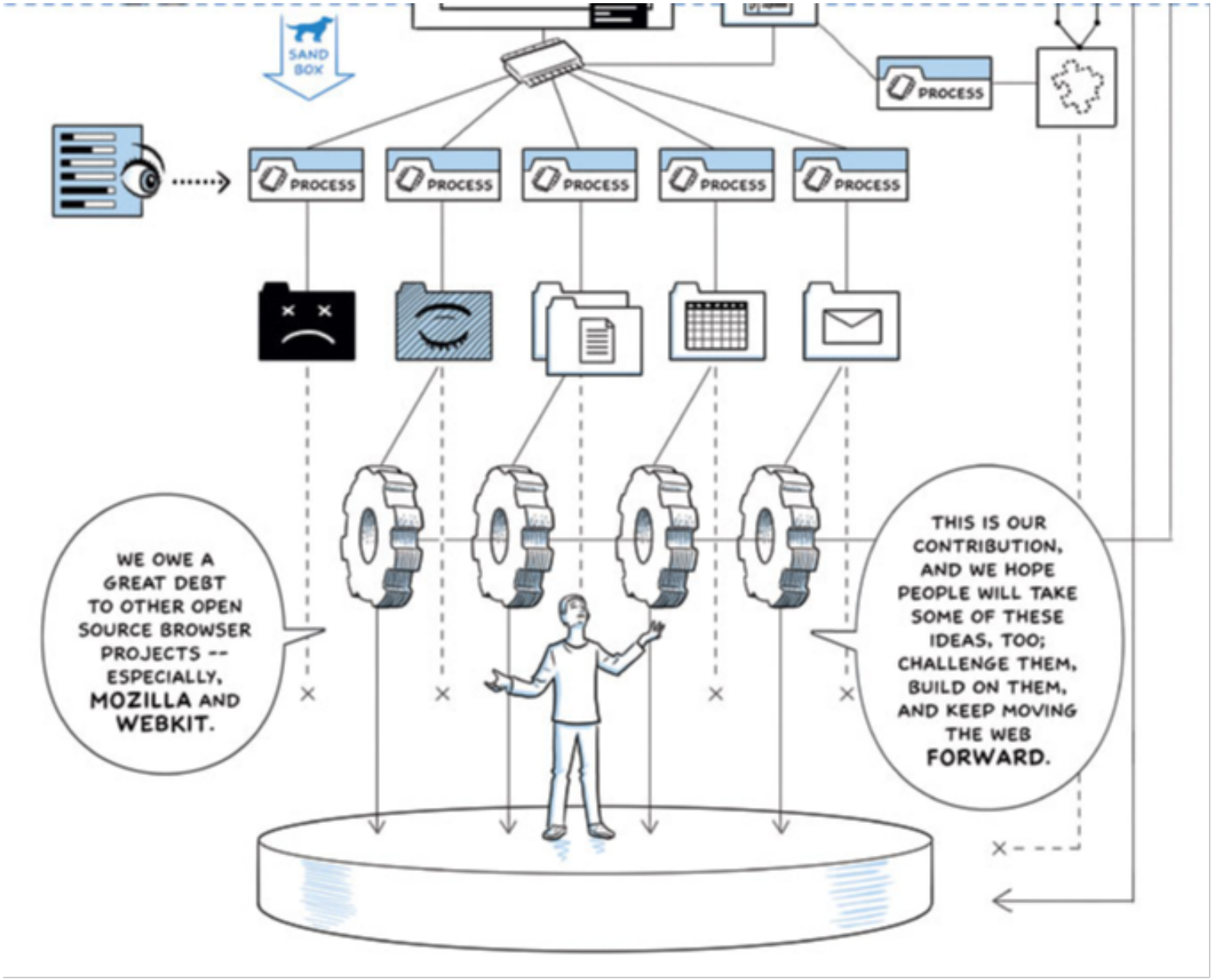

BUT --
-- SINCE IT'S OPEN SOURCE --
-- OTHER BROWSER DEVELOPERS CAN TAKE WHAT THEY WANT OUT OF IT.


THEY DON'T HAVE TO PAY US. THEY DON'T HAVE TO ASK OUR PERMISSION.
THEY DON'T HAVE TO SHARE PATCHES OR REPORT BUGS.*
* THOUGH, IF THEY LIKE, WE HAVE SYSTEMS IN PLACE FOR THAT.


BUT THEY CAN BUILD ON WHAT WE'VE DONE AND BRING THEIR OWN CREATIVITY TO IT.

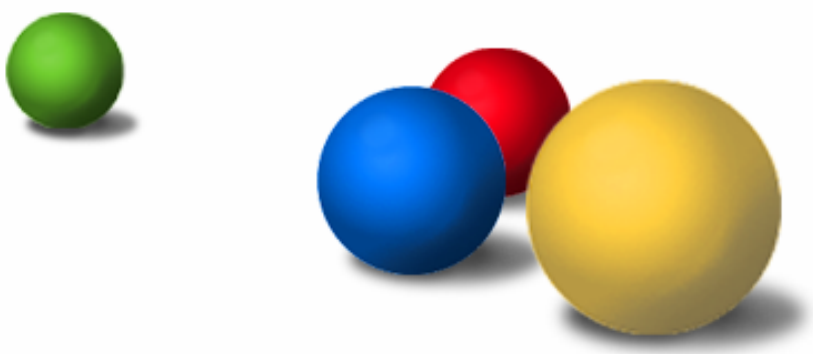

SURE, WE COULD SHIP A PROPRIETARY BROWSER AND HOLD IT IN.
BUT GOOGLE LIVES ON THE INTERNET.
IT'S IN OUR INTEREST TO MAKE THE INTERNET BETTER AND WITHOUT COMPETITION WE HAVE STAGNATION.
THAT'S WHY WE'RE OPEN SOURCING THE WHOLE THING. WE NEED THE INTERNET TO BE A FAIR, SMART, SAFE PLACE.


AS EXCITED AS WE ARE ABOUT BUILDING GOOGLE CHROME, IT'S IMPORTANT TO HELP ALL BROWSERS BECOME MORE POWERFUL --
-- TO KEEP EVOLVING WITH THE WEB AND CONTINUING TO BUILD A SOLID FOUNDATION FOR MODERN WEB APPLICATIONS.
WEBKIT
V8
01001
100111
PROCESS
PROCESS
USER
SAND BOX


SAND BOX
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
WE OWE A GREAT DEBT TO OTHER OPEN SOURCE BROWSER PROJECTS -- ESPECIALLY, MOZILLA AND WEBKIT.
THIS IS OUR CONTRIBUTION, AND WE HOPE PEOPLE WILL TAKE SOME OF THESE IDEAS, TOO; CHALLENGE THEM, BUILD ON THEM, AND KEEP MOVING THE WEB FORWARD.

Q&A

Thanks!



Chrome Tip: Memory

Google

- Memory usage falls into three categories: shared, shareable, and private
- Windows' Task Manager reports different numbers in different versions
- The best way to figure out what Chrome is using is to look at about:memory or use Chrome's own task manager (Shift-Esc)
- Multiple processes means more memory in minimal configurations but less in the long run

Chrome Tip: DNS Prefetch

Google

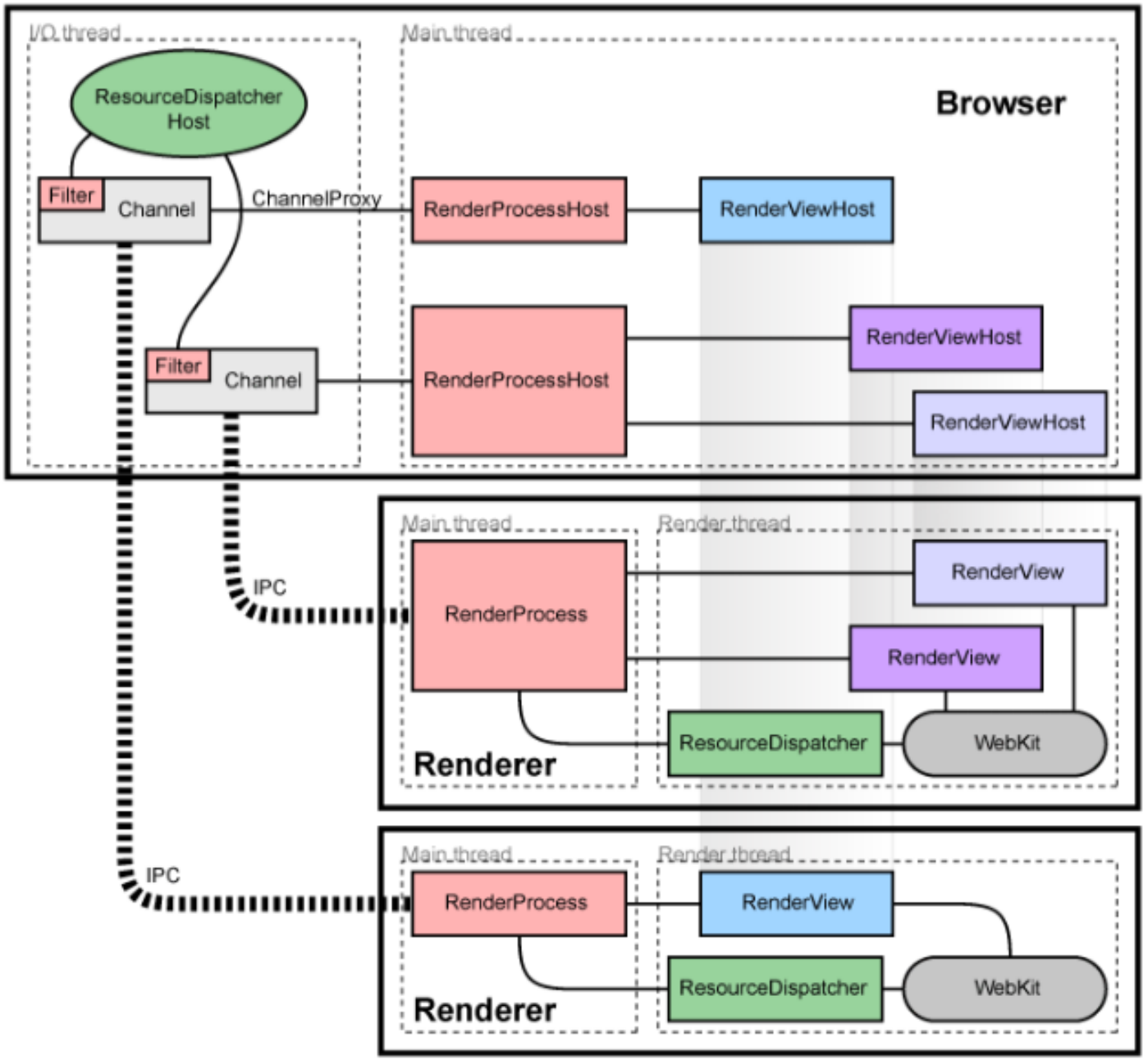
- DNS lookups are a surprising source of potential latency, with lookups that are 250ms or more being commonplace
- Chrome caches DNS lookups and populates the cache from links in the pages displayed to save you time
- about:histograms displays a lot of fun statistics about the workings of the browser, like DNS.PrefetchFoundName for prefetch stats

Chrome Tip: We really aren't evil!

Google

- Enabling chrome to share statistics with Google is a powerful way for Chrome users to work together to gather info
- Crash reports and usage statistics drive development that's good for everyone
- Incognito mode can always be used for specially secure browsing
- We hope you enjoy using and contributing to Chrome!

Multi-Process Architecture

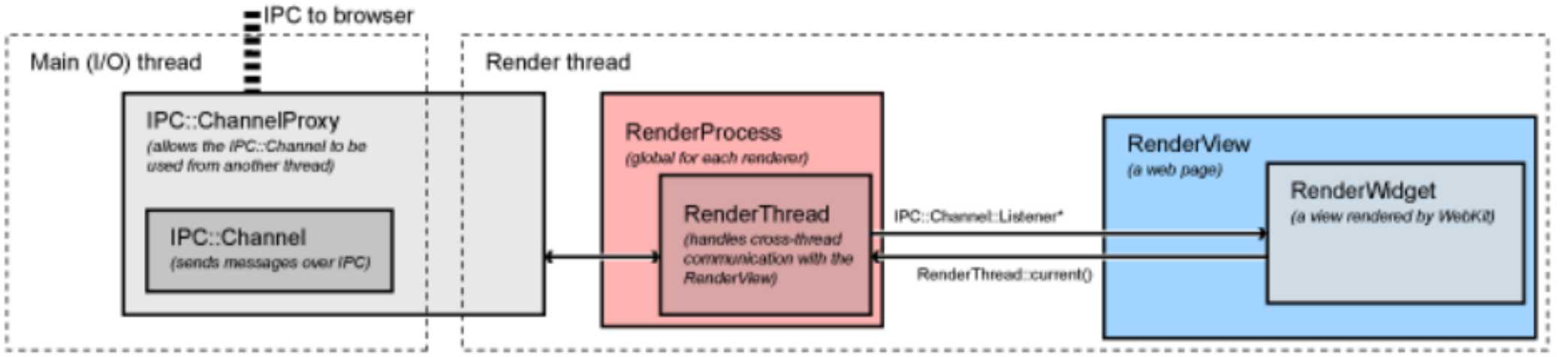


Multi-Process Architecture

Google

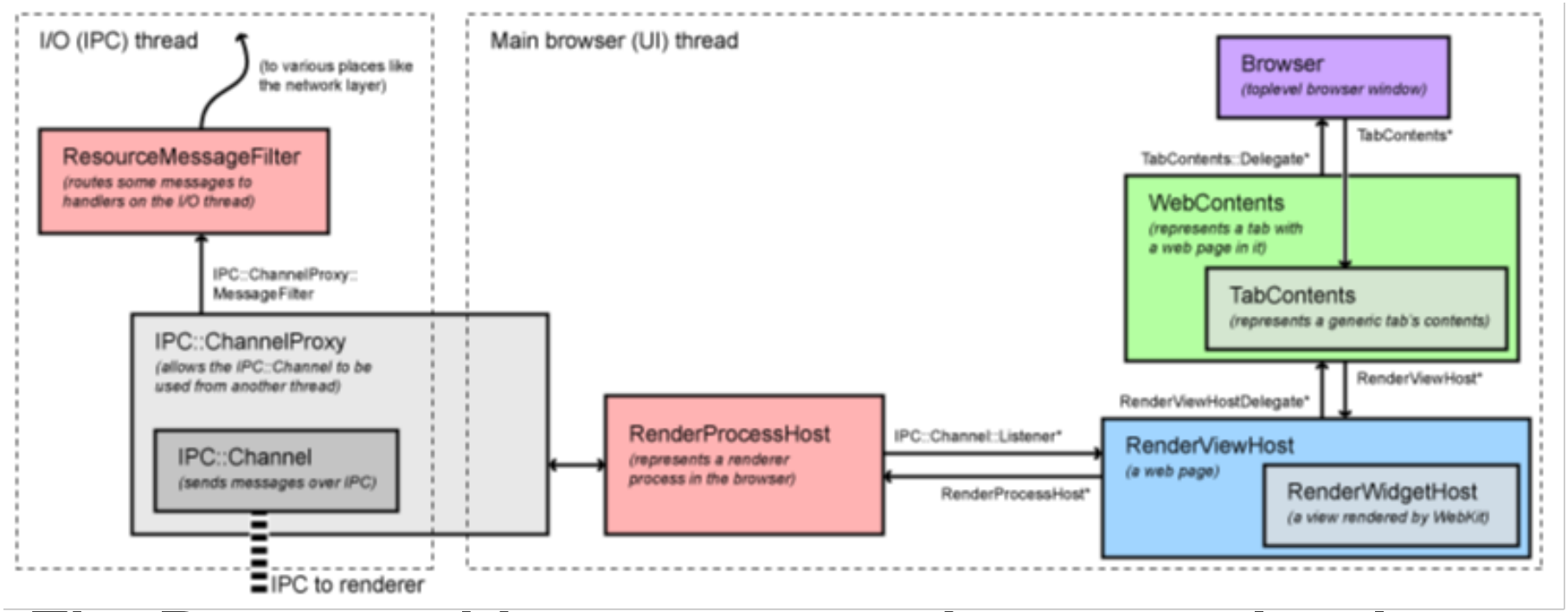
- A single browser process is the master
- Each web site is rendered by a single render process
- Communication between the two is via Chromium's IPC mechanism (named pipes)
- The master process is called a 'broker' and the slave processes are called 'sandboxes'

Anatomy of a Render Process



- The RenderProcess talks to the corresponding RenderHost in the browser. There is exactly one instance per process and it handles all communication to the browser
- The RenderView communicates with the corresponding RenderViewHost via the RenderProcess

Anatomy of the Browser Process



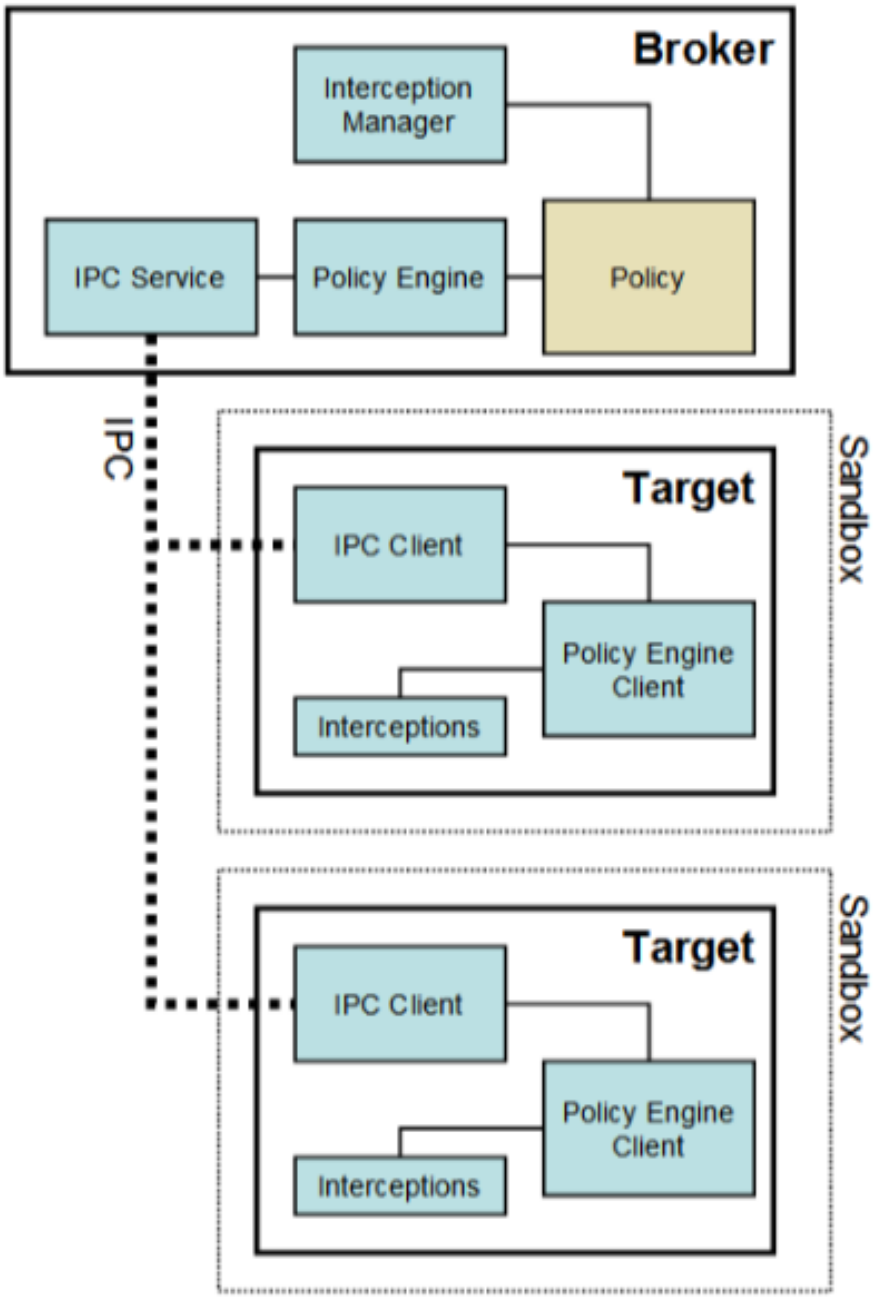
- The Browser object corresponds to a top-level window
- Each RenderProcessHost corresponds to each IPC connection to a render sandbox
- The RenderViewHost encapsulates rendering specific to a frame/DOM in the RenderProcess and handles painting and events

Life of a mouse click

Google

- The Windows message is received on the UI thread of the browser by RenderWidgetHostHWND::OnMouseEvent
- ForwardMouseEventToRenderer packages the input event into a cross-platform WebMouseEvent and sends it to the RenderWidgetHost
- RenderWidgetHost::ForwardInputEvent creates an IPC
- Then the renderer takes control:
- RenderView::OnMessageReceived gets the message and in turn forwards it to RenderWidget::OnHandleInputEvent.
- The event goes to WebWidgetImpl::HandleInputEvent where it is converted to a WebKit PlatformMouseEvent class and passed to the WebCore::Widget class inside WebKit.

Specific naming and isolation for security



The Broker's responsibilities

Google

- Specify the policy for each target process
- Spawn the target processes
- Host the sandbox policy engine service
- Host the sandbox interception manager
- Host the sandbox IPC service (to the target processes)
- Perform the policy-allowed actions on behalf of the target process

The Sandbox's limits

Google

- Sandbox is given a *restricted token*
- Sandbox is in a Windows *job* object
- Sandbox is confined to its own Windows *desktop* object
- Windows Vista+: sandbox is at the lowest integrity level

The Sandbox's Restricted Token

Google

- The restricted token means no access to secured objects
- Does a good job on properly configured(?) Windows systems
- Does not handle access to sockets on XP or access to legacy filesystems (FAT32 on USB Keys for example)

The Sandbox's Job

Google

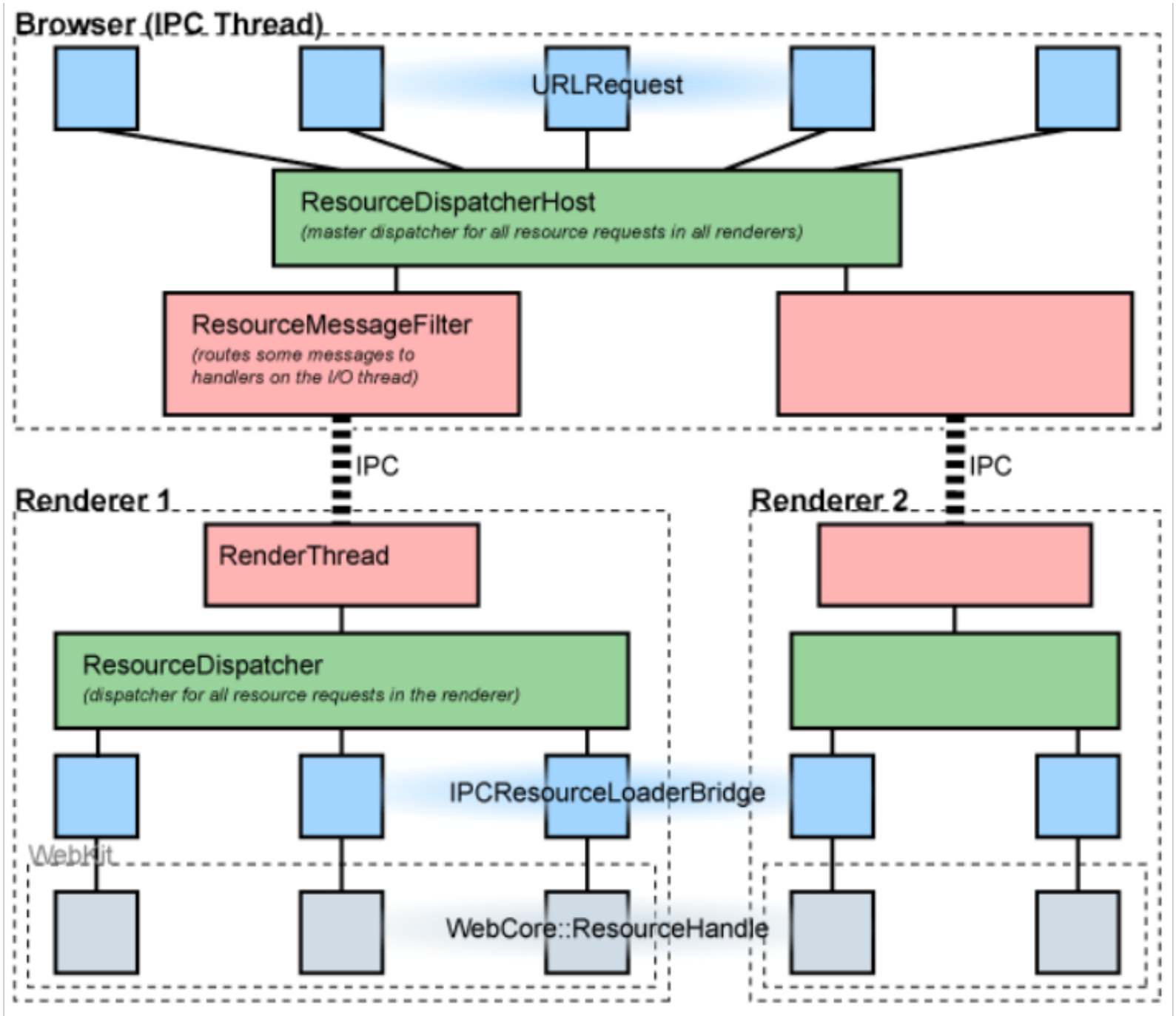
- The Job abstraction allows limiting access to system resources that are otherwise unsecured
- Forbids the creation or switch of desktops, modifying screen resolution, clipboard access, event broadcast, etc.
- Crucial for keeping the sandbox process inside its jail away from other windows

The Sandbox's Desktop

Google

- All windows on the same desktop are vulnerable to each other
- Screen scraping is one threat
- Synthesized events is another
- Keylogging is a third
- Isolating the sandboxes to their own desktop carries a small memory penalty but is otherwise effective

Isolation of Resource Loading



Resource Loading Caveats

Google

- It is assumed that cross-domain restrictions are handled by the renderer
- Handling cross domain rules outside the renderer would introduce a lot of complexity – consider pages that legitimately load resources from many sites versus javascript to do the same
- It is a non-goal of chromium to protect the user from XSS website attacks