



# Designing for Optimal Performance in SharePoint Online

@spmcdonough  
on Twitter  
(for heckling  
purposes)



Sean P. McDonough  
Microsoft MVP  
Bitstream Foundry LLC



# Sponsors

Platinum Sponsors



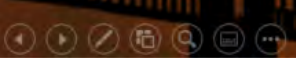
Silver Sponsors



Raffle Sponsors



0365 & SharePoint  
Saturday Cincinnati



# Our Agenda

- SharePoint Online Diagnostics and Tools
- Design and Development Guidance
- Samples and Examples
- Questions and Answers Throughout!
- References



But first ...





An important note

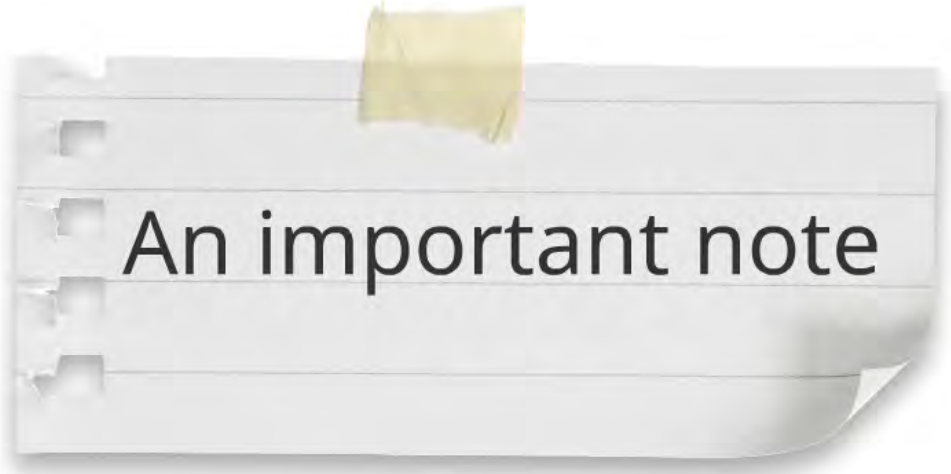


This is  
Microsoft  
(Office)  
365



changing and updating it"

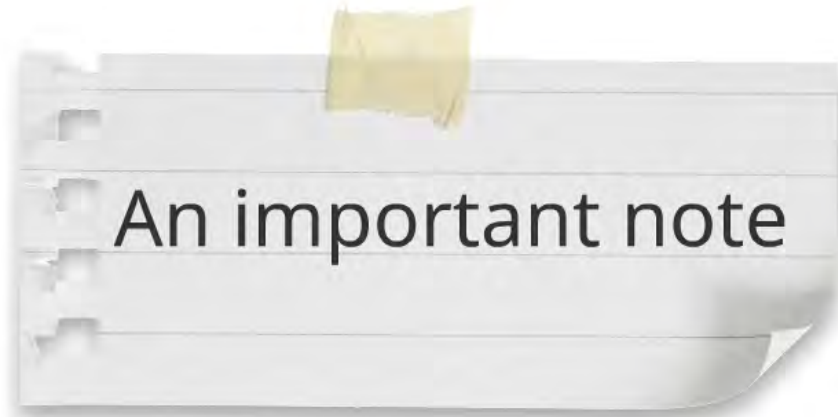
Please don



**Office 365 is an  
"evergreen service"**



meaning "Microsoft is always changing and updating"



**Office 365 is an  
"evergreen service"**

meaning "Microsoft is always changing and updating it"

**What I show you today ...**

- will probably be true tomorrow





# always changing and updating

## What I show you today ...

- will probably be true tomorrow
- has a good chance of being true next week
- might be true in month
- probably worth questioning and re-evaluating in a year



Please don't dig this up in five years and then send me hate mail because I presented something that is no longer accurate due to a SharePoint Online service change.



Please don't dig this up in five years and then send me hate mail because I presented something that is no longer accurate due to a SharePoint Online service change.



**Dear Sean,**

I was reviewing a presentation you put together five years ago, and I found elements that were incorrect. You are a horrible person and you should never touch SharePoint Online again.

Love you lots!  
- an attendee



We don't have time to cover the details of all of the networking, routing, and ways data can get into and out of SharePoint Online in this session ...

but please make a  
note of this:

If you've spent a lot of time in focus

We don't have time to cover the details of all of the networking, routing, and ways data can get into and out of SharePoint Online in this session ...

but please make a  
note of this:

If you've spent a lot of time in focused troubleshooting of SharePoint Online (to little or no effect), maybe you should zoom out and consider the network.



So your users have  
registered complaints  
and *it feels like* you've  
got performance issues

So your users have  
registered complaints  
and *it feels like you've*  
got performance issues



How do you prove it objectively?



Before you do too much damage, put the computers and equipment down and take a deep breath ...



objectively?



Allow me to  
introduce your  
primary diagnostic  
tool



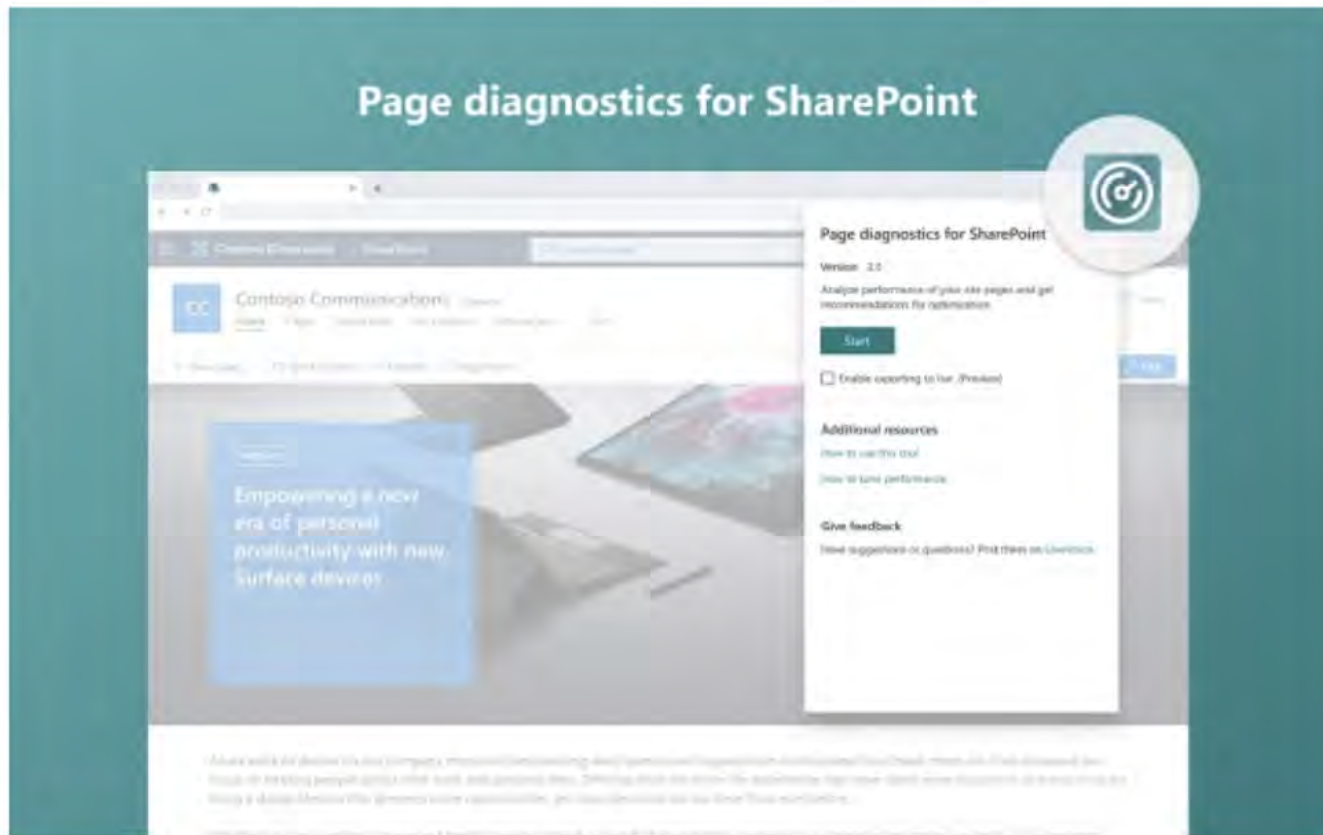
You might have heard of

introduce your  
primary diagnostic  
tool



You might have heard of  
it, but maybe not ...

# Allow me to introduce version 2.0 of the Page Diagnostics for SharePoint browser extension



You may be familiar with



You may be familiar with the initial release of this tool. Scott Stewart announced the availability of it at our SPC performance session in Las Vegas in 2018



0) put a stake in the ground, but we get a



You may be familiar with the initial release of this tool. Scott Stewart announced the availability of it at our SPC performance session in Las Vegas in 2018



v1.0 put a stake in the ground, but we get a much more robust version with version 2.0

- Clients
- Pages
  - Grabbing Performanc...
  - Recycle bin
  - Edit

+ New Send by email Promote Page details

## Grabbing Performance Metrics

Sean P. McDonough  
Owner

A modern page created for the purpose of analyzing performance data and information.

Comment

Comments

Add a comment. Type @ to mention s

Post



Collecting data... take a few minutes

DEMONO

\* potential routing issues (as in "number of hops")

\* slow DNS lookups, proxy authentication, etc.

# Demo Takeaways

HTTP

Response

Headers

waiting on server -  
generally zero or  
near zero

time spent  
processing on  
server (in ms)  
- ideally low

- SPIisLatency
- SPRequestDuration
- X-SharePointHealthScore

0 to 10  
(you want 0)

Don't see the headers?

**Don't panic!**

The Page Diagnostics Tool reports data on classic publishing pages and modern SharePoint pages. Other tools can get you numbers for pages, too.



HTTP  
Response  
Headers





# Roughly approximating for a page ...

## Page diagnostics for SharePoint ...

CorrelationID 27250d9f-f0f5-0000-470e-45ceced8b400  
SPRequestDuration 204ms  
SPIISLatency 0ms  
Page load time 2307ms  
Page URL <https://bitstreamfoundry.sharepoint.com/SitePages/Grabbi...>

Page load time - (SPRequestDuration + SPIISLatency) = "time lost elsewhere"

- \* network latency
- \* potential routing issues (as in "number of hops")
- \* slow DNS lookups, proxy authentication, etc.

# Demo Takeaways

HTTP

Response

waiting on server -

generally zero or

time spent

processing on

So, you've concluded that your pages are slow and you have the data to prove it!

SPIisLatency is low, and maybe your X-SharePointHealthScore is low,

but ...

So, you've concluded that your pages are slow and you have the data to prove it!

SPIisLatency is low, and maybe your X-SharePointHealthScore is low,

but ...

**SPRequestDuration is crazy high (e.g., 9000 ms)!**



Repeat after me ...

"The problem probably isn't

# Repeat after me ...

"The problem probably isn't  
SharePoint Online. It's my site."



Okay, one more time:

Repe

"The pro  
SharePoir

# Repeat after me ...

"The problem probably isn't  
SharePoint Online. It's my site."



So, who's to blame?

*(if you're into that sort of thing)*

In all likelihood

# So, who's to blame?

(if you're into that sort of thing)

In all likelihood:  
blame the  
lousy\* devs.



\*Note: not all devs are lousy devs. Just the ones who create performance problems and knee-jerk into blaming Microsoft and SharePoint Online.

# So, who's to blame?

(if you're into that sort of thing)

In all likelihood:  
blame the  
lousy\* devs.



\*Note: not all devs are lousy devs. Just the ones who create performance problems and knee-jerk into blaming Microsoft and SharePoint Online.

- Compare processing and response times for a SharePoint site or page. ← (may



- In the majority of poor performance scenarios, a combination of UI/UX , client-side code additions, and questionable customization/deployment mechanisms are to blame.
- Microsoft has indicated that the slowest 1% of pages in SPO take more than 5,000ms to load - again, usually due to customizations.

...nes who create performance  
...t and SharePoint Online.

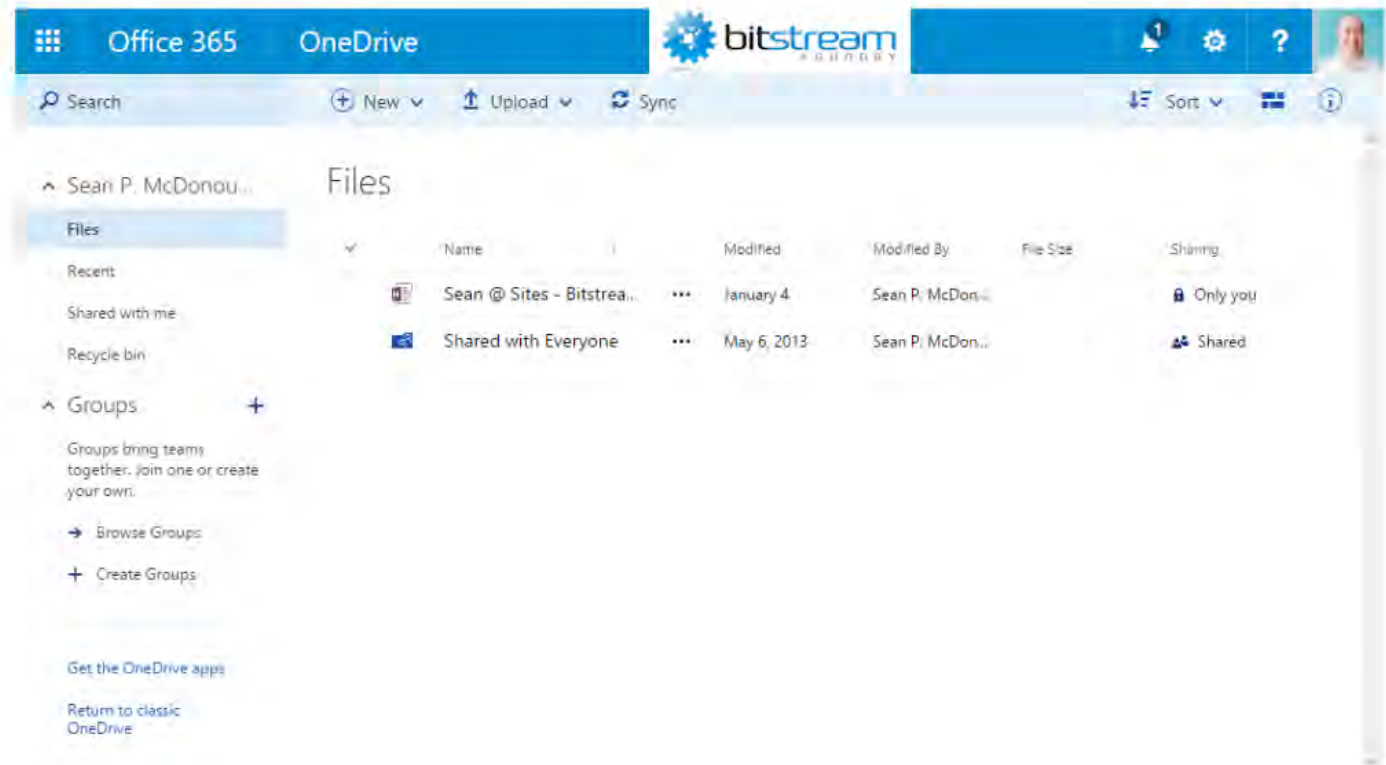
- Compare processing and response times to your problematic SharePoint site or page.  (may not be valid approach much longer ...)

- In the majority of poor performance scenarios, a combination of UI/UX , client-side code additions, and questionable customization/deployment mechanisms are to blame.
- Microsoft has indicated that the slowest 1% of pages in SPO take more than 5,000ms to load - again, usually due to customizations.

Don't believe me?



# Collect the data and validate for yourself!



- Profile your OneDrive for Business page (it's in your MySite).
- Compare processing and response times to your problematic SharePoint site or page. *(may not be valid approach much longer ...)*

- In the majority of poor performance scenarios a

Don't believe me?

"Okay, yeah - my OneDrive for Business page is really fast ... but my SharePoint pages are completely choking."



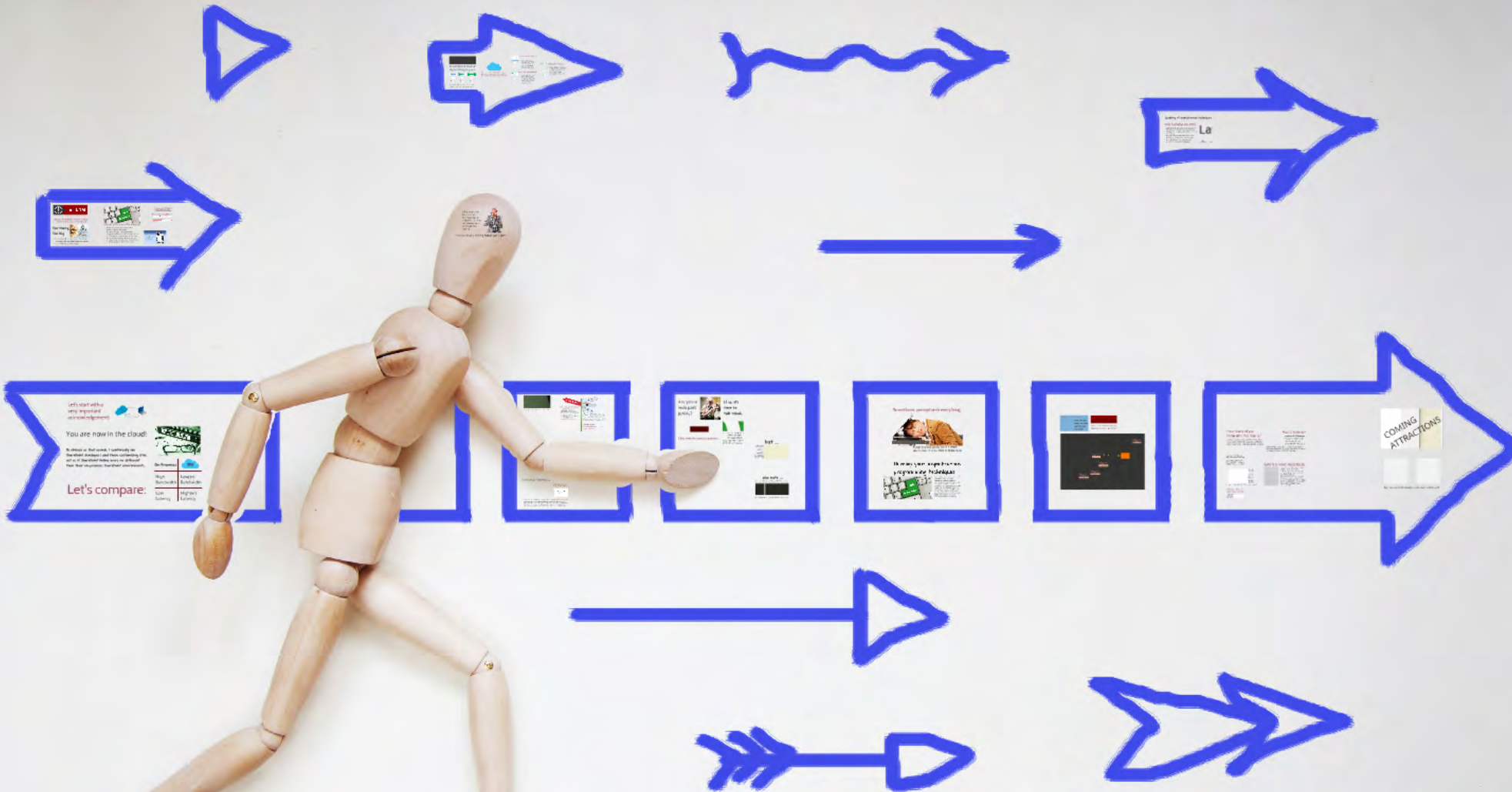
're probably thinking "What can I do

"Okay, yeah - my OneDrive for Business page is really fast ... but my SharePoint pages are completely choking."



You're probably thinking **"What can I do?"**





# Path to Better Performance

Let's start with a  
very important  
acknowledgement:

Let's start with a  
very important  
acknowledgement:



You are now in the cloud!

As obvious as that sounds, I continually see

acknowledgement:

# You are now in the cloud!

As obvious as that sounds, I continually see SharePoint developers and those customizing sites act as if SharePoint Online were no different than their on-premises SharePoint environments.

very important  
acknowledgement:




# You are now in the cloud!

As obvious as that sounds, I continually see SharePoint developers and those customizing sites act as if SharePoint Online were no different than their on-premises SharePoint environments.

## Let's compare:



On-Premises	 SPO
High Bandwidth	Low(er) Bandwidth
Low Latency	High(er) Latency



Failing to acknowledge the "we're in the cloud now" reality leads to a problem I simply call ...



Failing to acknowledge the "we're in the cloud now" reality leads to a problem I simply call ...

Too Many,  
Too Big



- Too many calls are made to the server.

now" reality leads to a problem I simply call ...

# Too Many, Too Big



- Too many calls are made to the server.
- Too many files are referenced on pages.
- The files in-use are too large.





Consider one or more of the following:

- Minify files, especially JavaScript files.

• Resize images to usage sizes

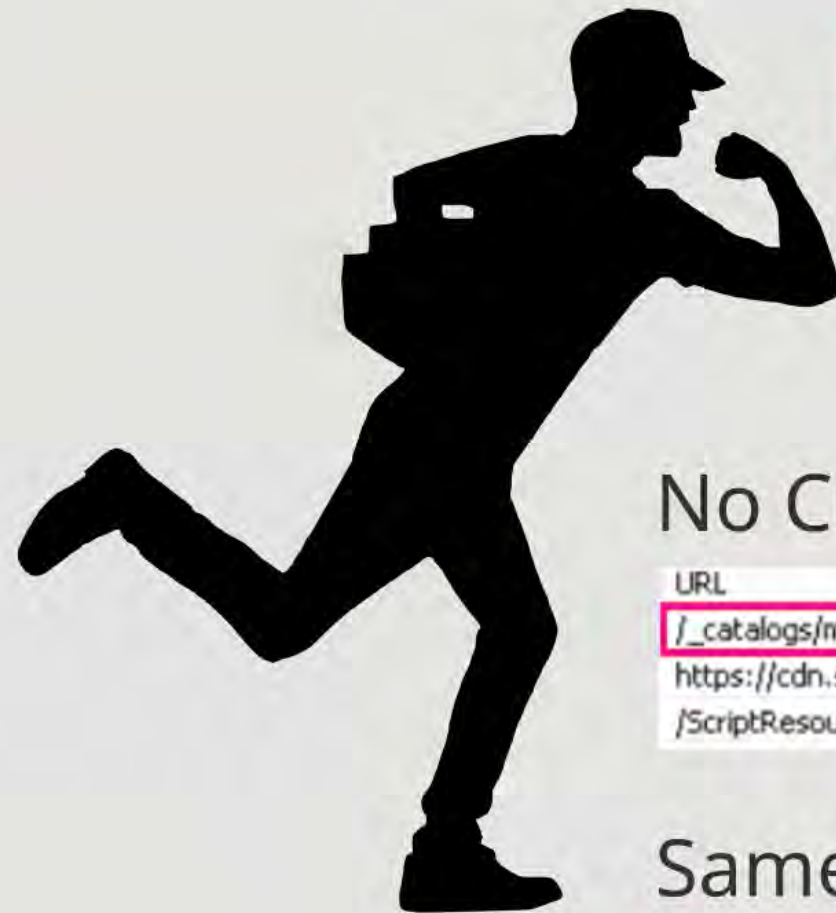


## Consider one or more of the following:

- Minify files, especially JavaScript files.
- Resize images to usage sizes.
- Compress images (more) aggressively.
- Use sprite sheets to reduce the actual number of HTTP requests needed to retrieve images.
- Use SharePoint's Image Rendition service.
- Leverage a toolkit like Font Awesome in place of individual icons and associated files.

And the big  
kahuna ...





# Use a CDN!!!

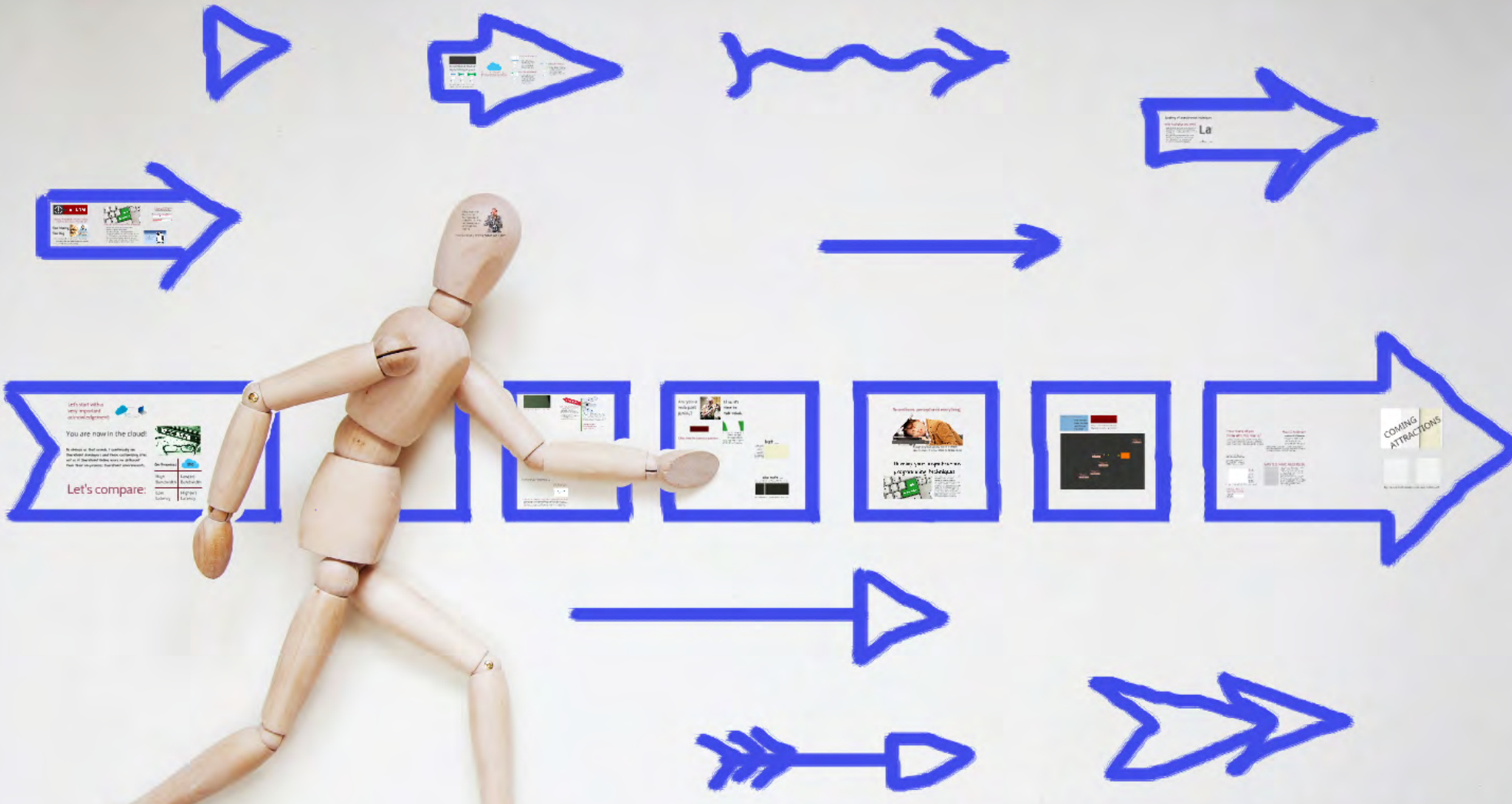
(Content Delivery Network)

No CDN in use (i.e., SPO library direct)

URL	Received	Taken	Initiator
<a href="#">/_catalogs/masterpage/javascript/jquery-2.1.1.min.js</a>	82.98 KB	1.51 s	<script>
<a href="https://cdn.sharepointonline.com/12413/_layouts/15/16">https://cdn.sharepointonline.com/12413/_layouts/15/16</a>	18.98 KB	156 ms	<script>
<a href="#">/ScriptResource.axd?d=M1vNi_a6A2vtkOenP45i9-peGfx</a>	100.80 KB	2.04 s	<script>

Same resource from a CDN

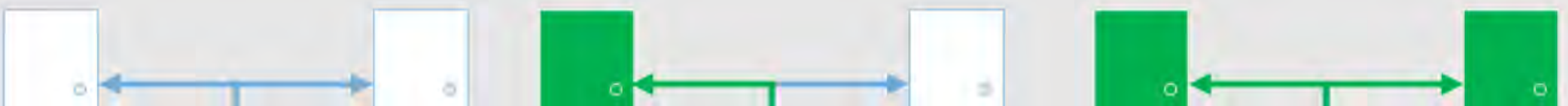
URL	Received	Taken
<a href="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.1.min.js">https://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.1.1.min.js</a>	82.74 KB	469 ms
<a href="#">/WebResource.axd?d=r3Mv/y4JFCBwmUs1-gLXCgVJy4RMAH/qCj2oIh3D5kbMXzSdwm5KlpDx9vM6MKjztZon...</a>	22.33 KB	0.84 s
<a href="#">/_layouts/15/images/spcommon.png?rev=38</a>	20.56 KB	1.15 s



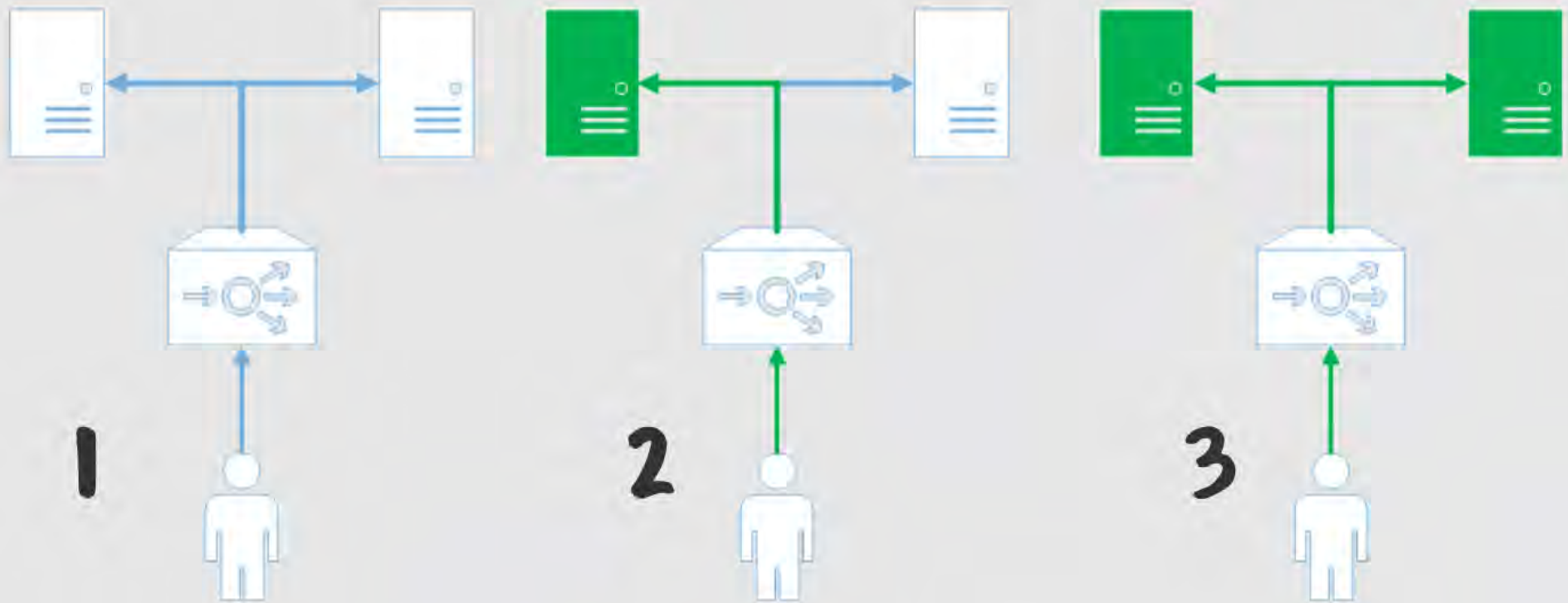
# Path to Better Performance



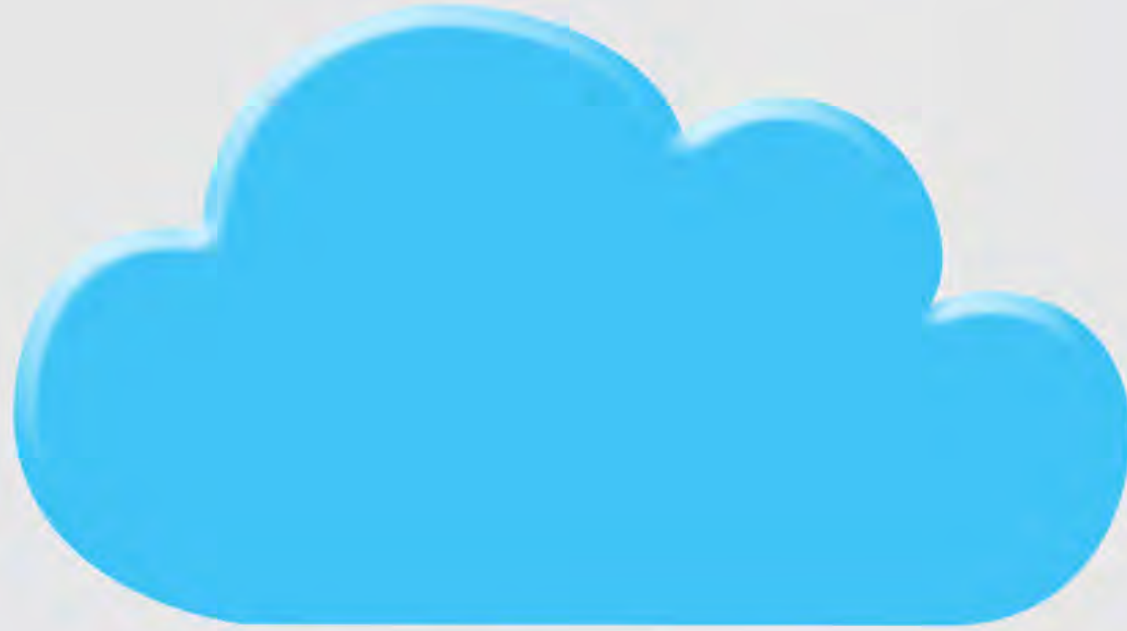
Conventional wisdom  
says caching is good.



# says caching is good.



After just a few requests, the on-premises Object Cache can be "ready for action."



In the cloud, the caching equation (for per-server memory-based caches like the Object Cache) works out a bit differently.\*

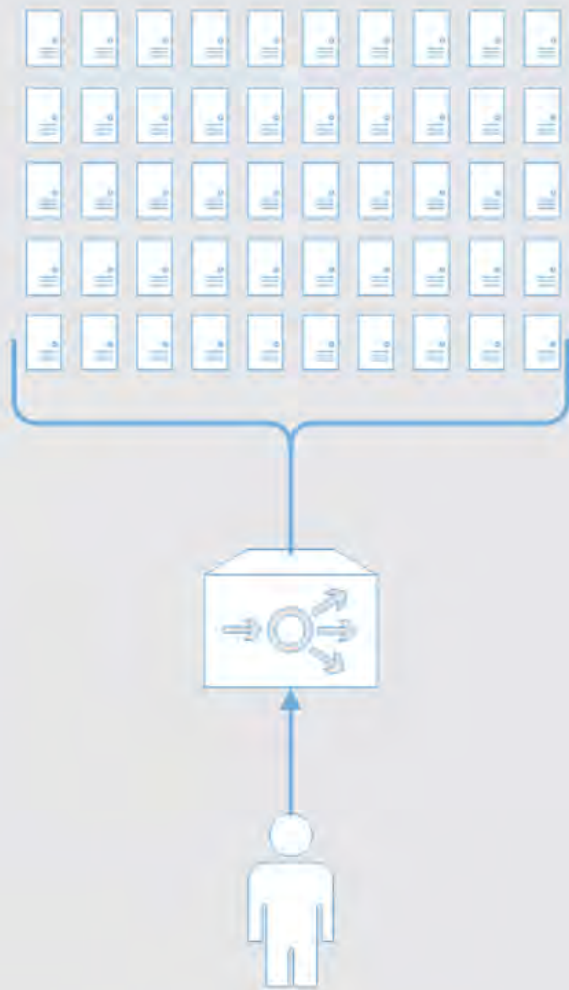






"differently" -  
as in "not at all"





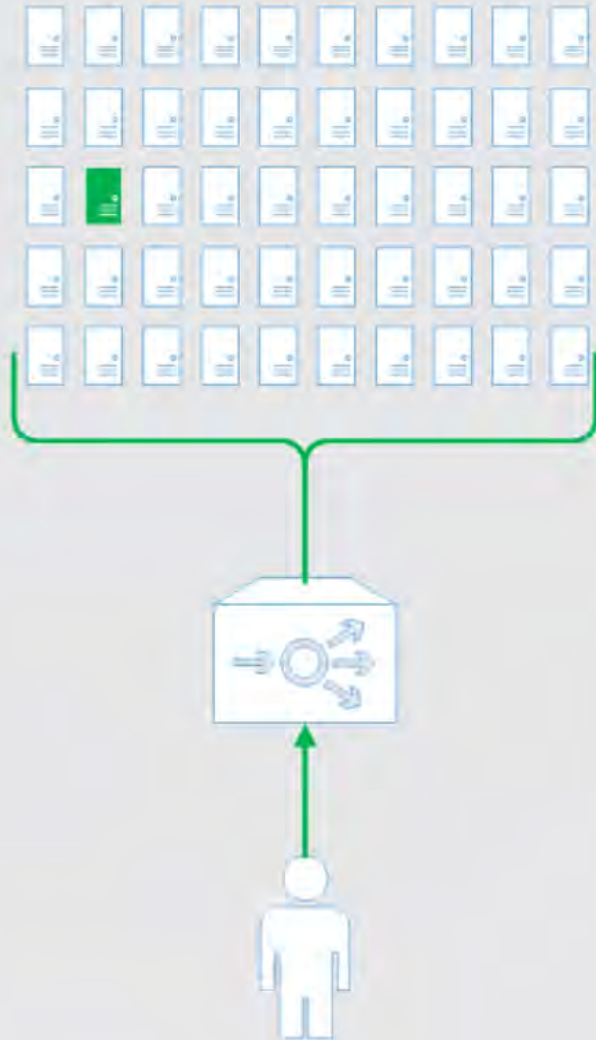
# User's Initial Request

- First thing to note: the number of WFEs tends to be \*much\* higher in the cloud versus on-premises.

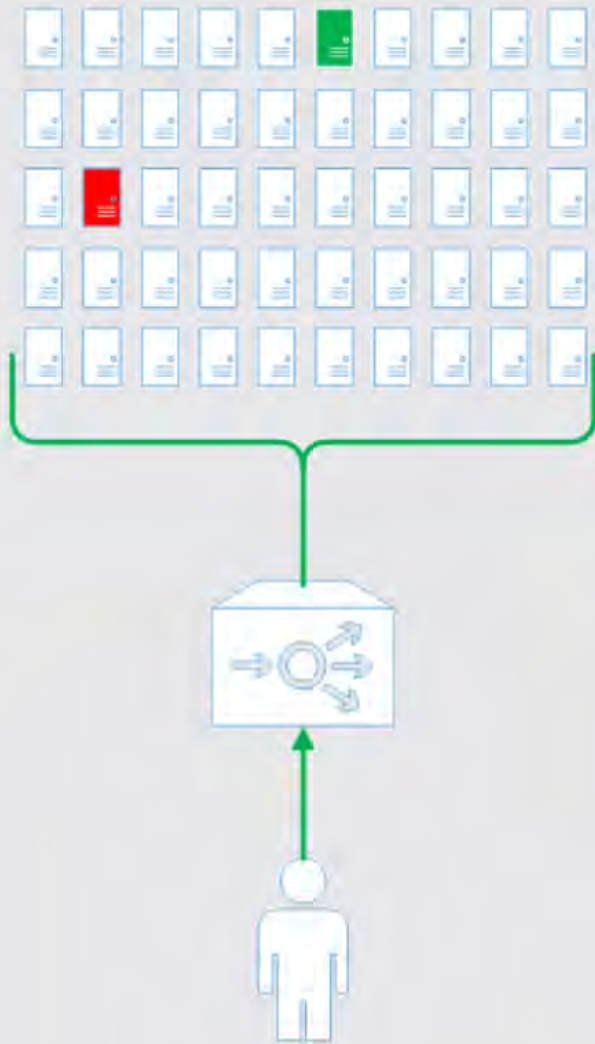


# User's Second Request

# User's Second Request

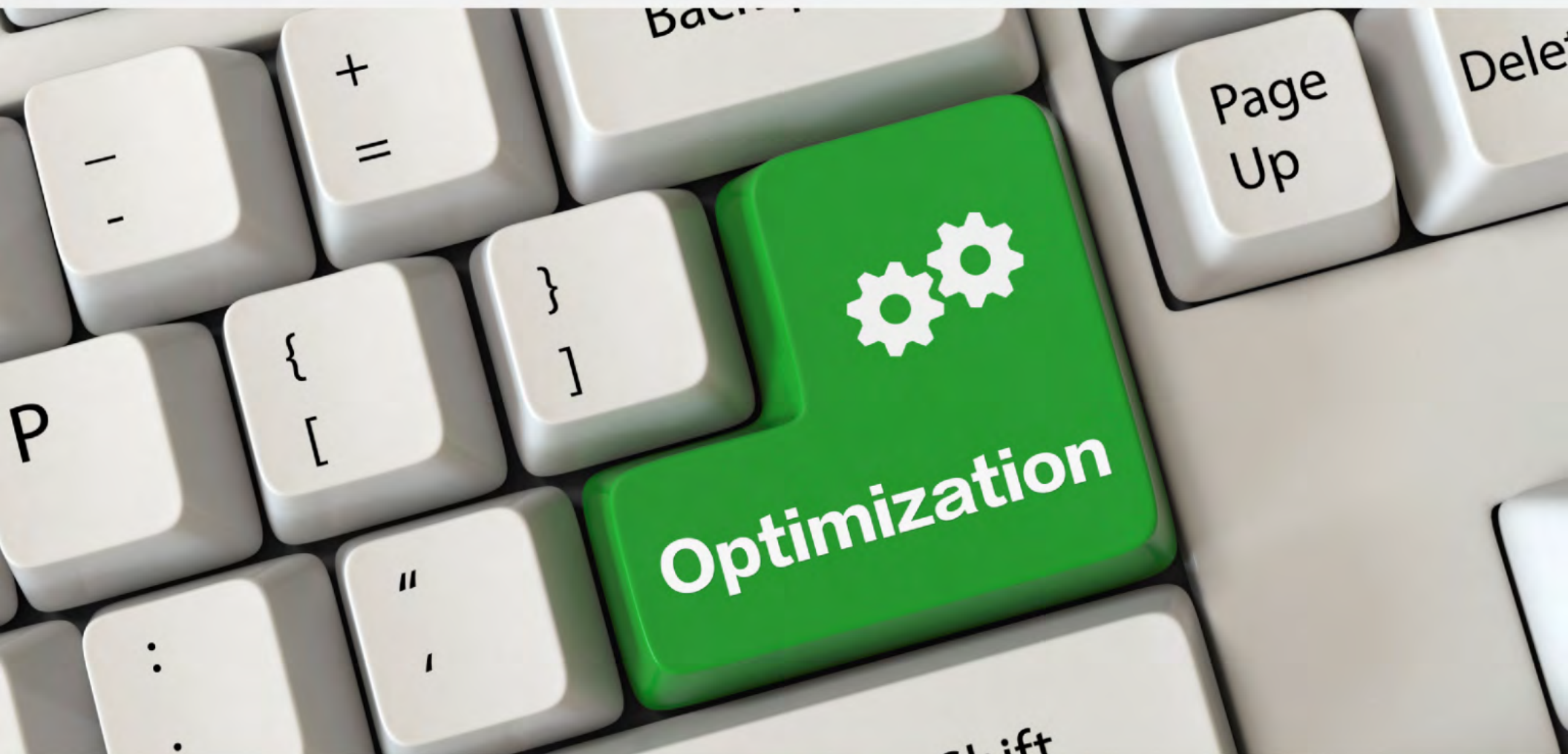


- No affinity is in use, so the chance of a user hitting the same server again is dramatically less than the on-premises scenario.



## Subsequent Requests

- Same reduced chance of hitting the WFE last visited
- Memory pressure causes much more frequent cache ejections versus on-premises.



Two significant adjustments can be made.

\* These sitemaps are then stored in the Object

Navigation style  
has a huge impact  
on performance.

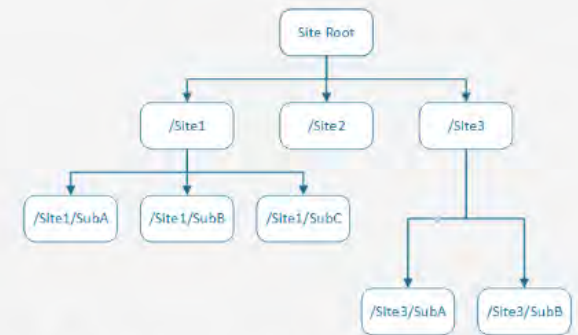
CHOOSE

CHOICE

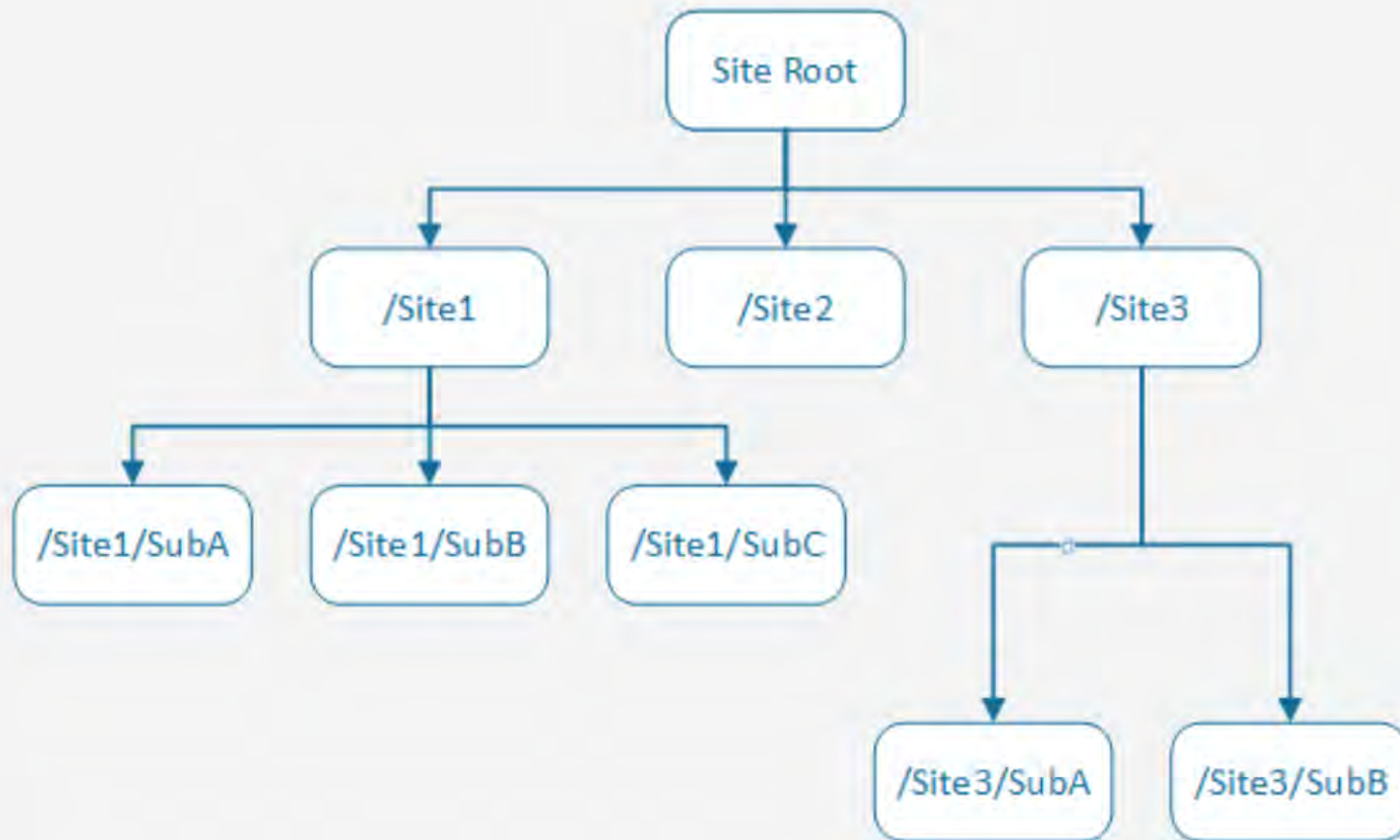
Using structural navigation is the default (but poor) choice for complex site hierarchies in the cloud.

- \* building each site node generates roughly 8 SQL Server round trips
- \* These sitemaps are then stored in the Object Cache on WFEs

Navigation style  
has a huge impact  
on performance.



8 site nodes/~64 SQL calls



8 site nodes/~64 SQL calls





## Better Options for Navigation

- Managed Navigation (i.e., using a term set to drive navigational structures) can significantly improve page performance.  
*note: the SharePoint Server Publishing Infrastructure site collection Feature must be enabled to use Managed Navigation*
- Search-driven navigation leverages SharePoint's Search index and the process of client-side navigational rendering to dramatically speed things up.  
*note: implementation is non-trivial and less customizable*

Using structural navigation is the default (but poor) choice for complex site hierarchies in the cloud.

- \* building each site node generates roughly 8 SQL Server round trips
- \* These sitemaps are then stored in the Object Cache on WFEs

## Navigation style



As was pointed-out in the navigational scenario,  
Search can be used to boost performance significantly.

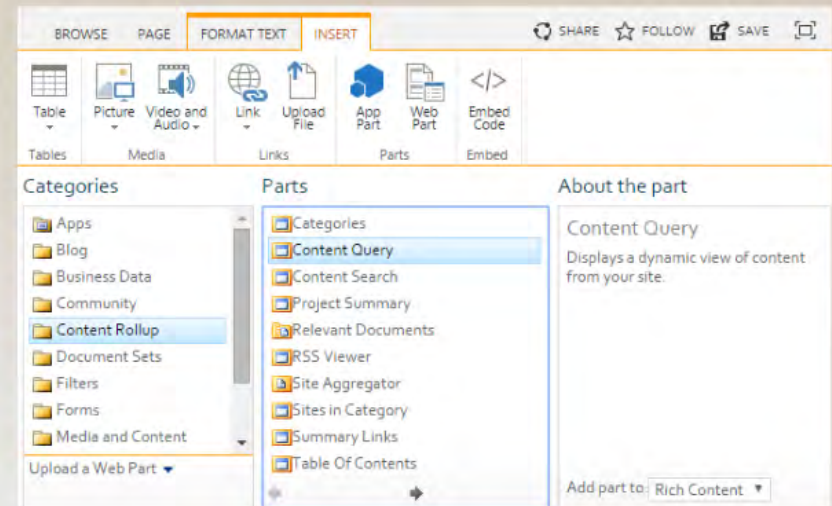


In the cloud, the CQWP can cause some signifi

ut in the navigational scenario,  
ed to boost performance significantly.



Do you like the Content  
Query Web Part (CQWP)?

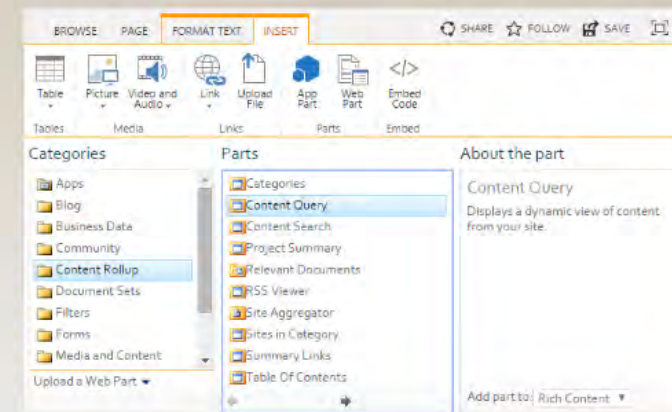


the CQWP can cause some significant performance hits.

As was pointed-out in the navigational scenario,  
Search can be used to boost performance significantly.



Do you like the Content  
Query Web Part (CQWP)?



**In the cloud, the CQWP can cause some significant performance hits.**

- The CQWP performs expensive cross-list and cross-site queries at run-time.
- The CQWP relies on the Object Cache to store results for acceptable performance.
- The Content Search Web Part (CSWP) provides options that are similar to the CQWP (and in a number of ways, more powerful) and uses Search so it's FAST!

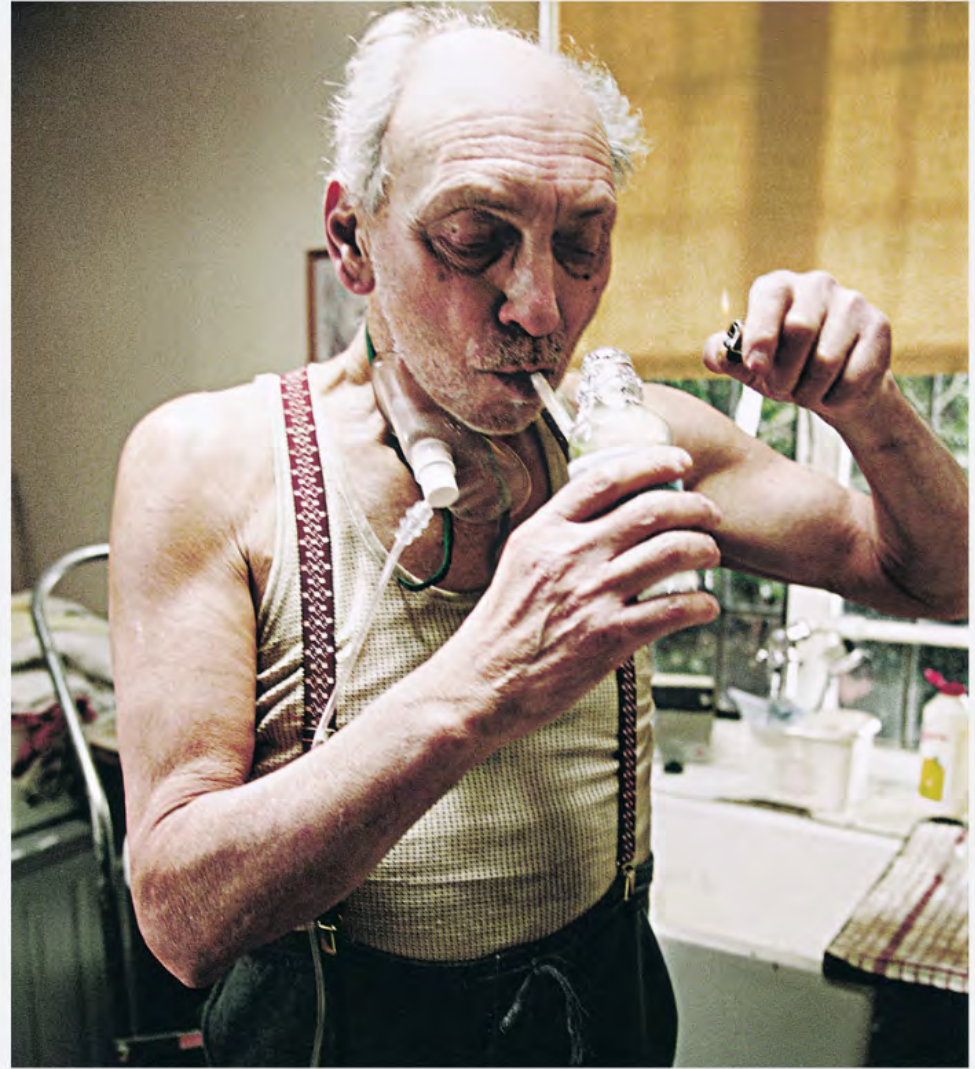


# Path to Better Performance



Okay, time for a serious question ...

Are you a  
web part  
junkie?

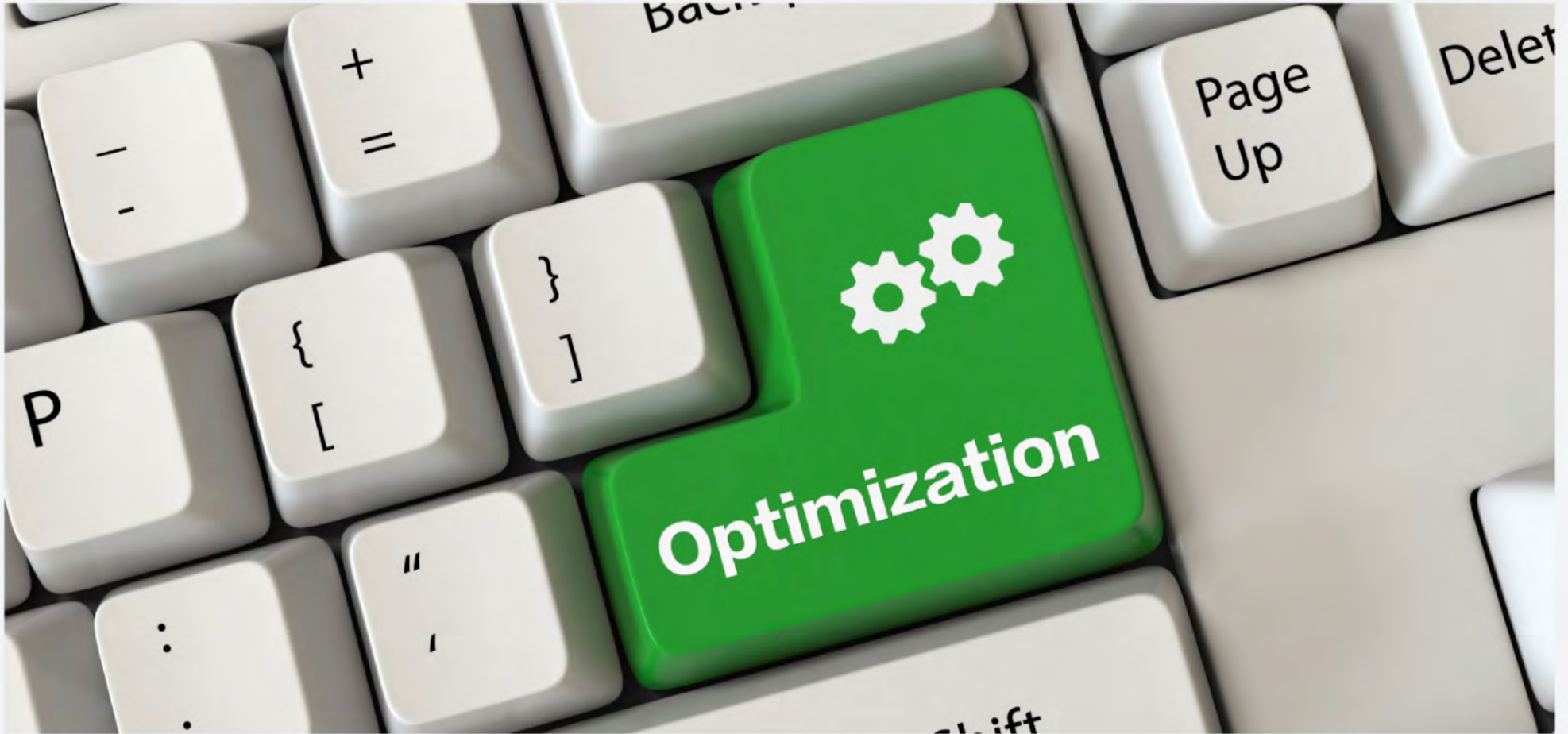




If so, it's  
time to  
talk rehab.



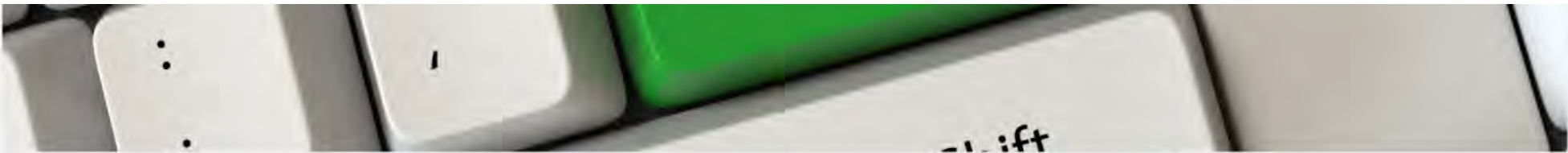




What's the alternative to freebasing web parts?



**There's no single**



What's the alternative to freebasing web parts?



**There's no single  
(or simple) answer.**

Generally speaking, consider leveraging client-side code (JavaScript) and asynchronous techniques - both of which we'll discuss soon.

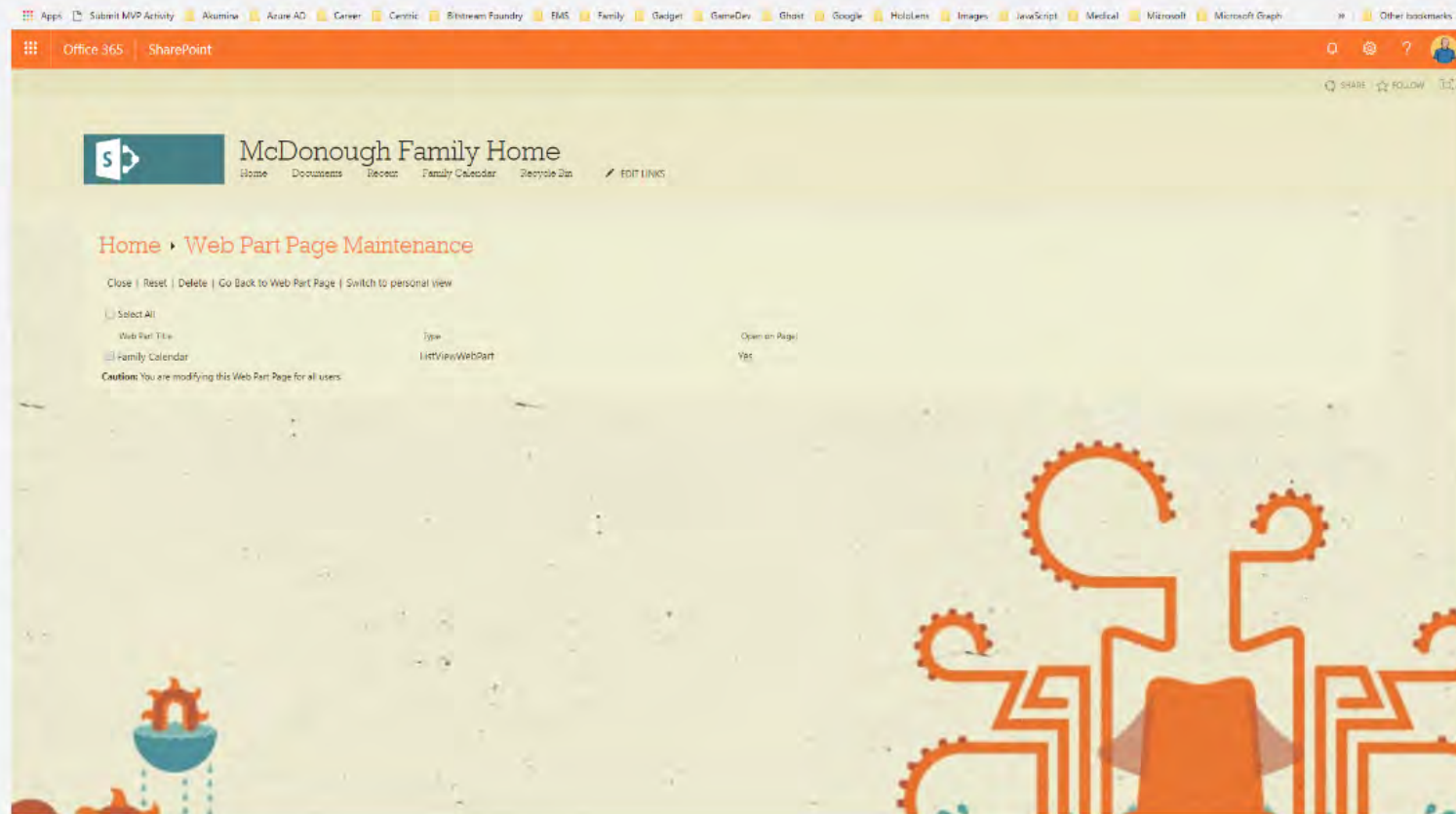
but ...

Be sure to put  
problem pages

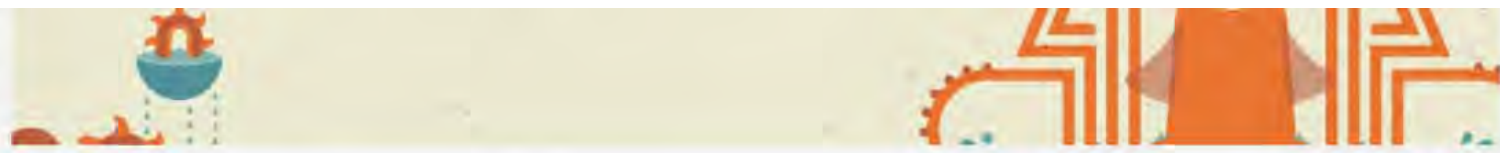


# but ...

Be sure to put  
problem pages  
in web part  
maintenance  
view with  
**?contents=1**  
to find web parts  
which are closed  
but not deleted!



which are closed  
but not deleted!



**also note ...**

When I say "web part," I'm talking about traditional (server-side) web parts. Everything is different, and all bets are off with SPFx/client-side web parts.

# also note ...

When I say "web part," I'm talking about traditional (server-side) web parts. Everything is different, and all bets are off with SPFx/client-side web parts.

```
HELLOWORD-WEBPART
├── .vscode
├── config
├── dist
├── lib
├── node_modules
├── src
├── temp
├── .editorconfig
├── .gitignore
├── .yo-rc.json
├── gulpfile.js
├── package-lock.json
├── package.json
├── README.md
├── tsconfig.json
└── tslint.json
```

```
yo @microsoft/sharepoint

Welcome to the
SharePoint Client-side
Solution Generator

Let's create a new SharePoint solution.
? What is your solution name? helloworld-webpart
? Which baseline packages do you want to target for your component(s)? SharePoint Online only (latest)
? Where do you want to place the files? Use the current folder
? Do you want to allow the tenant admin the choice of being able to deploy the solution to all sites immediately
ny feature deployment or adding apps in sites? No
? Which type of client-side component to create? WebPart
? What is your Web part name? HelloWorld
? What is your Web part description? HelloWorld description
? Which framework would you like to use? (Use arrow keys)
> No JavaScript framework
  React
  Knockout
```

And a friendly reminder: use a CDN with those SPFx web parts!

# also note ...

When I say "web part," I'm talking about traditional (server-side) web parts. Everything is different, and all bets are off with SPFx/client-side web parts.

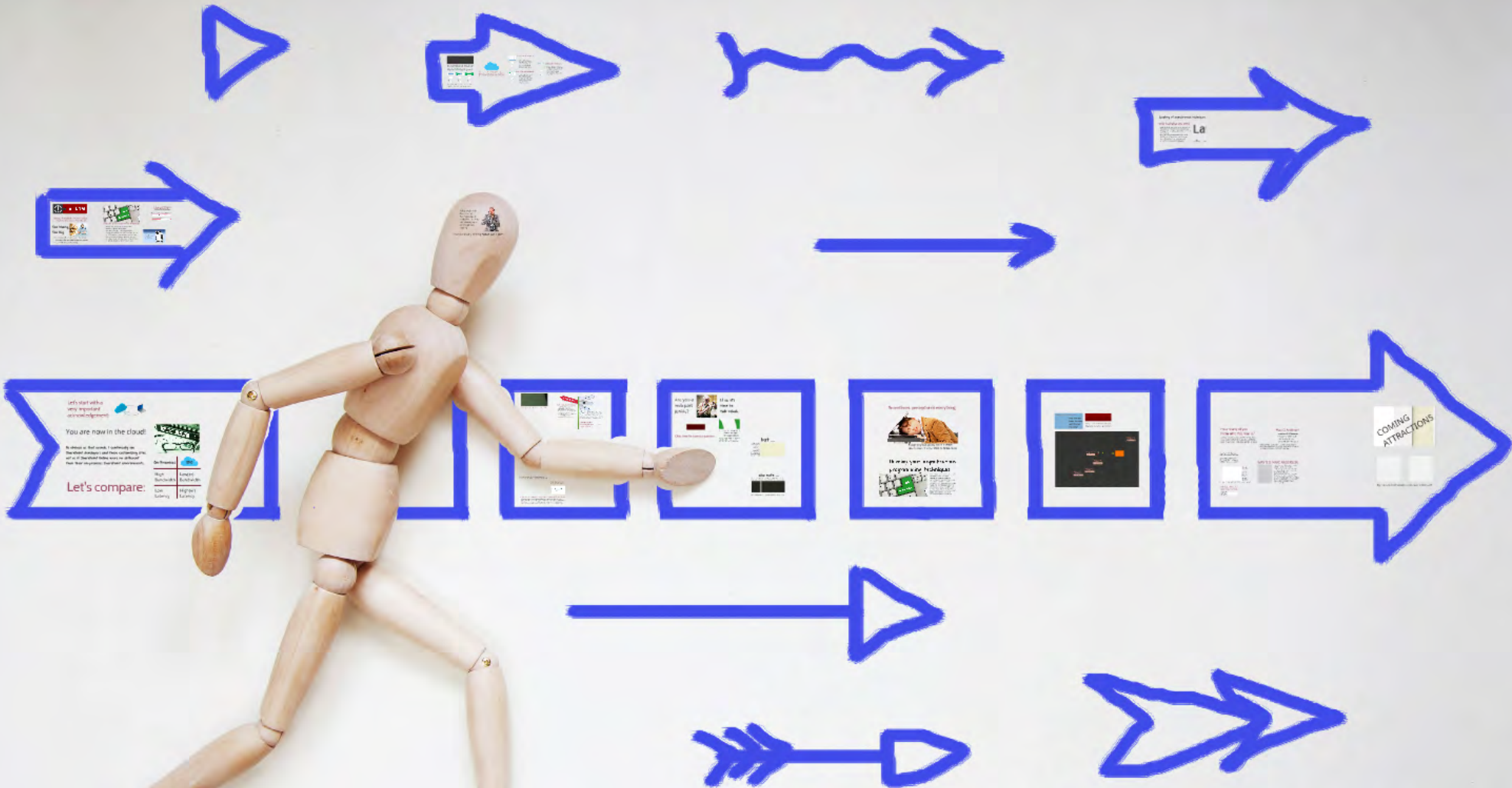
```
HELLOWORD-WEBPART
├── .vscode
├── config
├── dist
├── lib
├── node_modules
├── src
├── temp
├── .editorconfig
├── .gitignore
├── .yo-rc.json
├── gulpfile.js
├── package-lock.json
├── package.json
├── README.md
├── tsconfig.json
└── tslint.json
```

```
yo @microsoft/sharepoint

Welcome to the
SharePoint Client-side
Solution Generator

Let's create a new SharePoint solution.
? What is your solution name? helloworld-webpart
? Which baseline packages do you want to target for your component(s)? SharePoint Online only (latest)
? Where do you want to place the files? Use the current folder
? Do you want to allow the tenant admin the choice of being able to deploy the solution to all sites immediately by feature deployment or adding apps in sites? No
? Which type of client-side component to create? WebPart
? What is your Web part name? HelloWorld
? What is your Web part description? HelloWorld description
? Which framework would you like to use? (Use arrow keys)
> No JavaScript framework
  React
  Knockout
```

And a friendly reminder: use a CDN with those SPFx web parts!

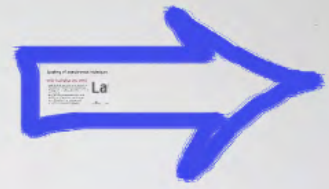
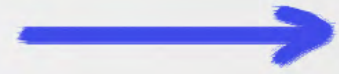


# Path to Better Performance





# Path to Better Performance



Sometimes, perception is everything.



**ALARM**

A page may load quickly, but if it FEELS slow to users, it is the SAME AS BEING SLOW.

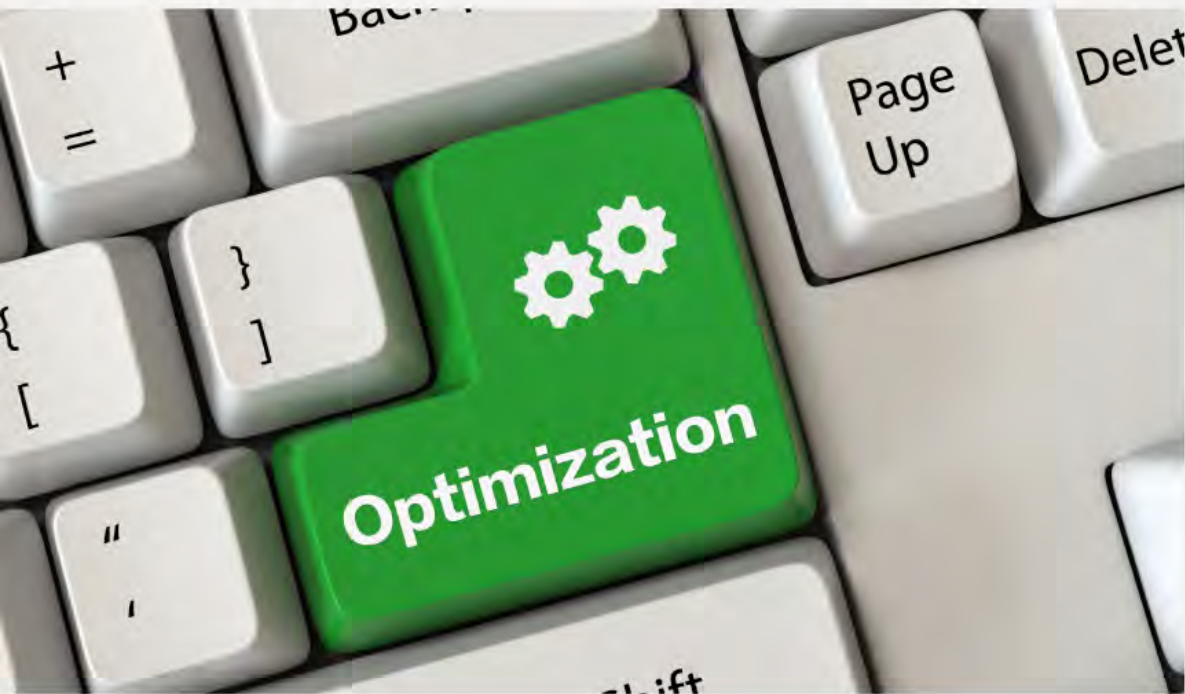
# Sometimes, perception is everything.



**ALARM**

A page may load quickly, but if it **FEELS** slow to users, it is the **SAME AS BEING SLOW.**

# Develop your asynchronous programming techniques



- You can't block a browser's main thread of execution, so leveraging async development patterns is essential.
- Async programming is made much easier in jQuery using promises. Promises approximate a synchronous programming model under asynchronous conditions.
- Certain web parts (e.g., the CSWP) also allow you to set their (a)sync behavior.
- Good use of async techniques make pages **appear** to load faster ... and as we discussed, perception is everything.

# Speaking of asynchronous techniques:

## Only load what you need.

- Instead of fetching everything at once within the context of the initial page load, retrieve the page with only the payload that's needed immediately.
- (Lazy) load images and other items "below the fold" only if users start scrolling down and will see them (e.g., Facebook and LinkedIn's "forever-scrolling" pages).





# Path to Better Performance

Ask yourself this question:



How well do I  
know the code  
and libraries  
I'm using?

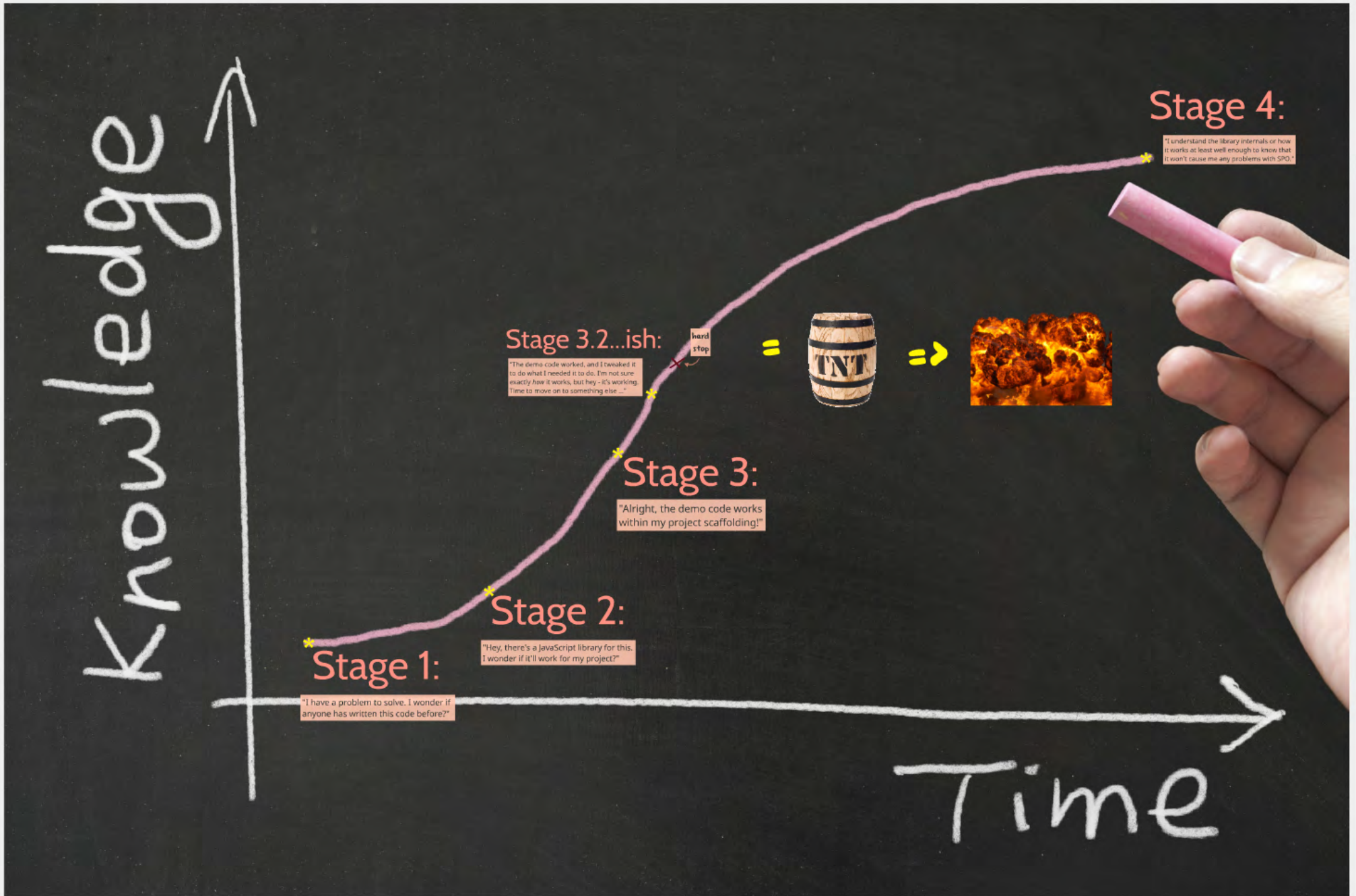


Reason I ask: when troubleshooting performance issues, I commonly encounter a pattern that can be illustrated with the following diagram of stages:



I'm using?

be illustrated with the following diagram of stages:



"Alright,  
within m



\*  
Stage 1:

"I have a problem to solve. I wonder if anyone has written this code before?"

\*  
Stage 2:  
"Hey, there's a JavaScript library for this. I wonder if it'll work for my project?"



# Stage 3:

"Alright, the demo code works within my project scaffolding!"

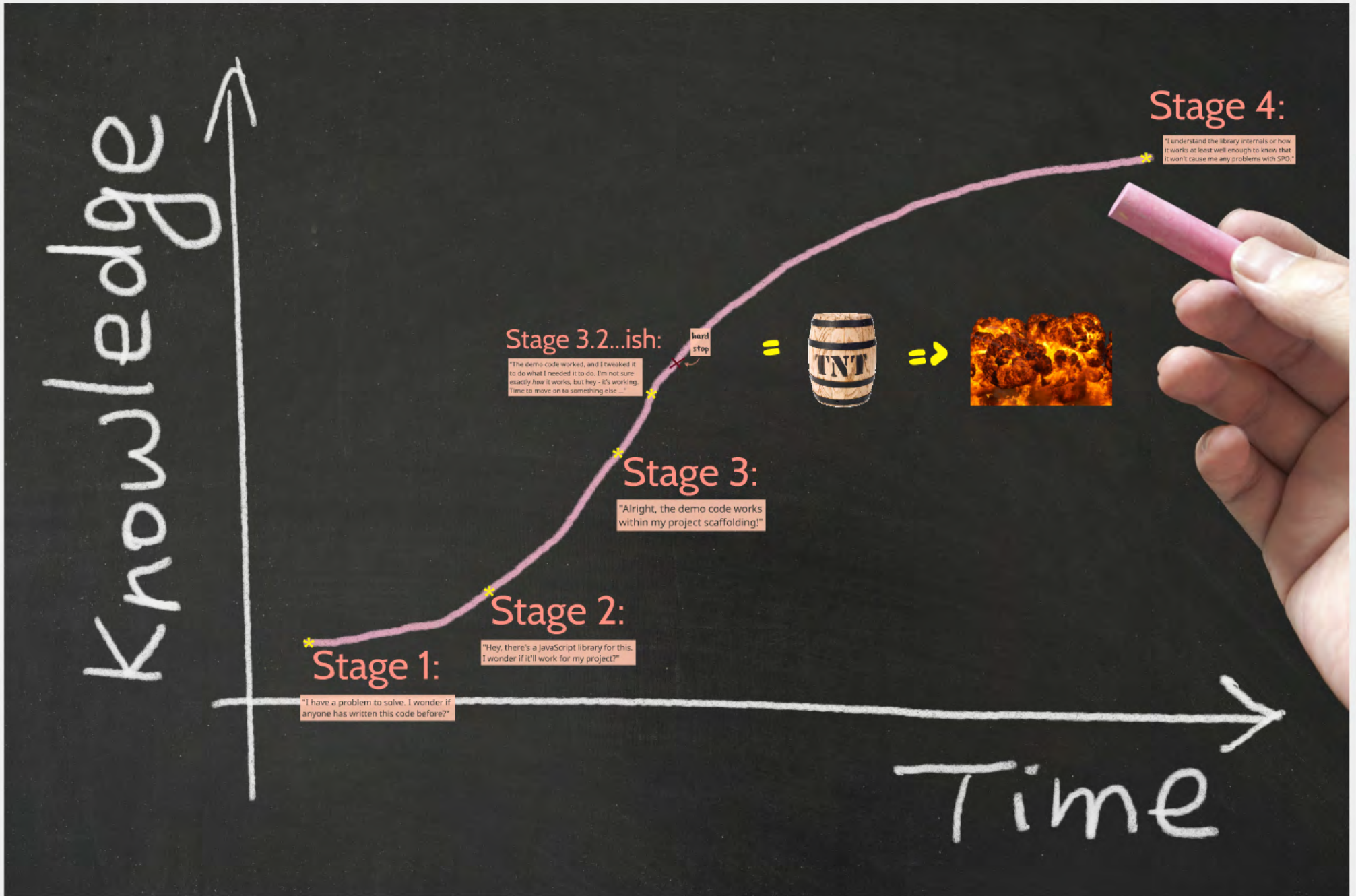
# Stage 4:

"I understand the library internals or how it works at least well enough to know that it won't cause me any problems with SPO."



I'm using?

be illustrated with the following diagram of stages:



# Stage 3.2...ish:

"The demo code worked, and I tweaked it to do what I needed it to do. I'm not sure exactly *how* it works, but hey - it's working. Time to move on to something else ..."

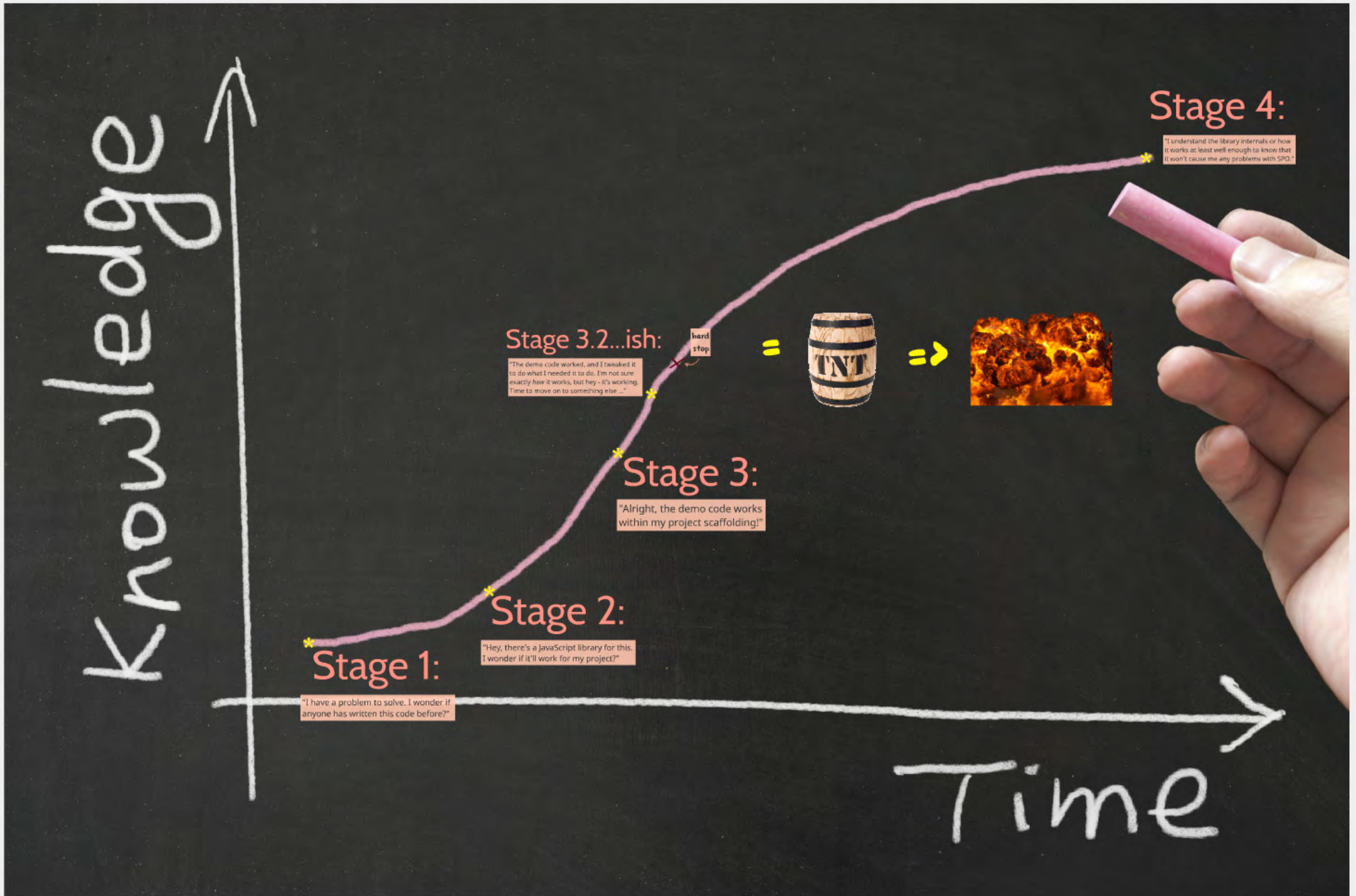
hard  
stop



# Stage 3:

I'm using?

be illustrated with the following diagram of stages:



example

How many of you





example

## How many of you know who this man is?

Chances are at least one or two of you have used code that he has created to get things done in your client-side development projects ...



Before CSOM/JSOM and REST  
Marc's library simplified access  
for developers everywhere. It



# Marc D. Anderson

- **creator of SPServices**
- master of client-side development and associated techniques

Before CSOM/JSOM and REST APIs - and before WCF SVC endpoints - Marc's library simplified access to the older ASMX web service endpoints for developers everywhere. It's still used heavily today.

So, getting back to "know your code/libraries" and how they work ...

As Marc will tell you, SPServices works just

So, getting back to "know your code/libraries" and how they work ...

As Marc will tell you, SPServices works just fine with SharePoint Online. But even Marc will tell you that you probably shouldn't use all of SPServices' methods when accessing SPO.

```
1 //Pre-populate all "Contact" fields with current user
2 var thisUserName = $().SPServices.SPGetCurrentUser({
3     fieldName: "Title",
4     debug: false
5 });
6 $().SPServices.SPFindPeoplePicker({
7     peoplePickerDisplayName: "Contact",
8     valueToSet: thisUserName,
9     checkNames: true
10 });
11 $().SPServices.SPFindPeoplePicker({
12     peoplePickerDisplayName: "Author/Contact",
13     valueToSet: thisUserName,
14     checkNames: true
15 });
16 $().SPServices.SPFindPeoplePicker({
17     peoplePickerDisplayName: "Organizer/Contact",
18     valueToSet: thisUserName,
19     checkNames: true
20 });
```

Consider  
this code.

It works just  
fine and does  
exactly what  
the comment  
indicates.

**But it has a  
big problem.**

Anyone ever used the SPServices.SPGetCurrentUser() method

```
1 //Pre-populate all "Contact" fields with current user
2 var thisUserName = $().SPServices.SPGetCurrentUser({
3     fieldName: "Title",
4     debug: false
5 });
6 $().SPServices.SPFindPeoplePicker({
7     peoplePickerDisplayName: "Contact",
8     valueToSet: thisUserName,
9     checkNames: true
10 });
11 $().SPServices.SPFindPeoplePicker({
12     peoplePickerDisplayName: "Author/Contact",
13     valueToSet: thisUserName,
14     checkNames: true
15 });
16 $().SPServices.SPFindPeoplePicker({
17     peoplePickerDisplayName: "Organizer/Contact",
18     valueToSet: thisUserName,
19     checkNames: true
20 });
```

Consider  
this code.

It works just  
fine and does  
exactly what  
the comment  
indicates.

**But it has a  
big problem.**

Has anyone ever used the SPServices.SPGetCurrentUser() method?

Switching over to REST-based calls

# WANTED: MARC ANDERSON



- Under the hood, `SPGetCurrentUser()` is generating an additional call to `/_layouts/userdisp.aspx` to “scrape” the contents of the page that is returned.
- If you (innocently) use `SPGetCurrentUser()` in your JavaScript files (especially multiple times in the context of a single page), you're creating all sorts of additional load on SPO and delaying the final results of your executing scripts.

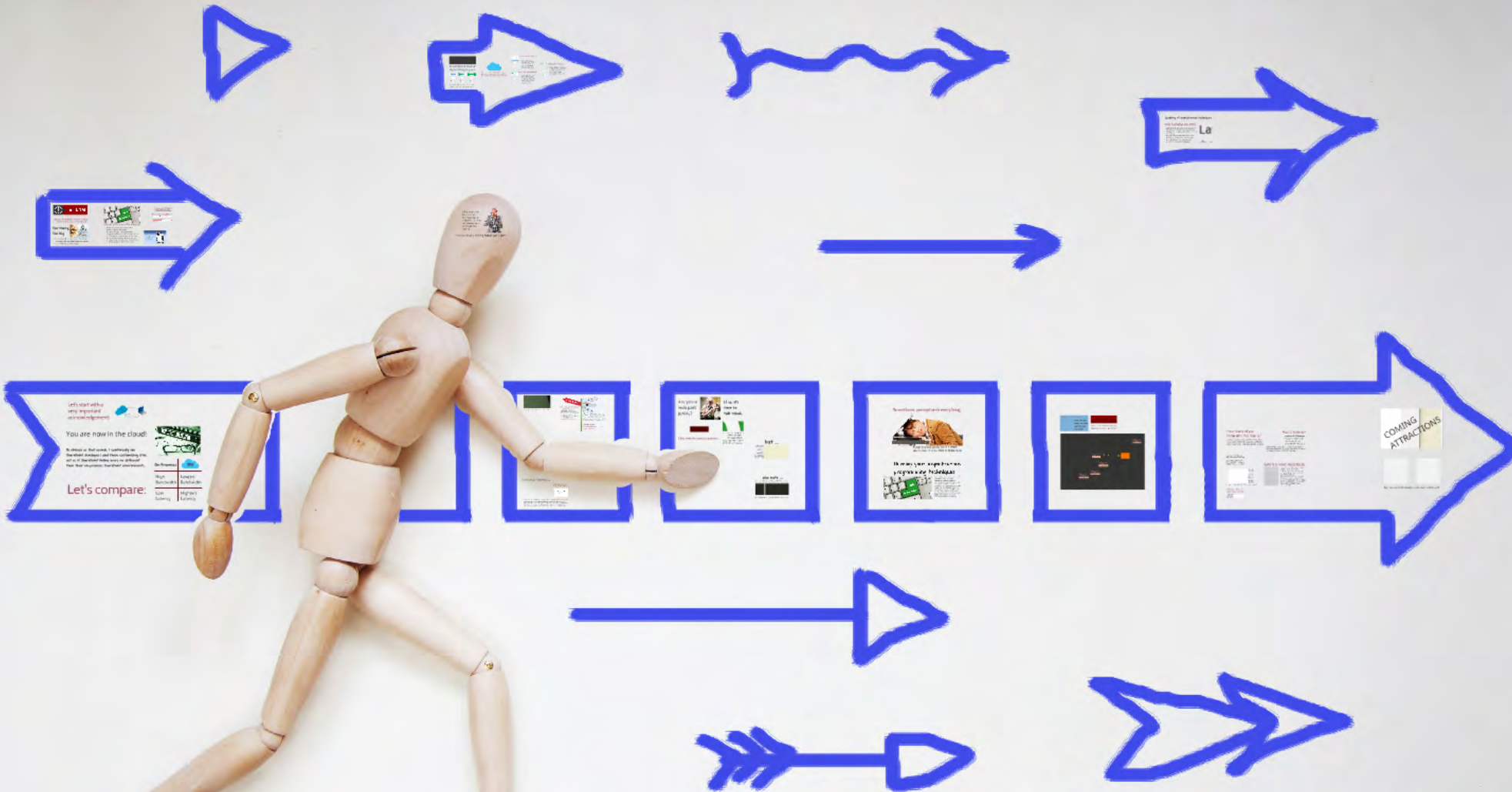
# Switching over to REST-based calls to get current user information can dramatically reduce execution time.

We had a script where `SPGetCurrentUser()` was being called several times. The results from swapping-in REST-based calls for the `SPGetCurrentUser()` calls:

I also performed some basic tests to capture the speed differences. I performed each test 10 times and here are the results:

- \* Without the fix or browser caching - avg. 14.47 seconds
- \* With the fix without browser caching - avg. 7.17 seconds
- \* With the fix and browser caching - avg. 5.84 seconds





# Path to Better Performance

# The Quick Summary



- Don't treat SPO like your on-premises SharePoint farm. The two operate differently.
- Server-based caching isn't your friend (generally speaking) in SPO.
- Your browser can be your best friend when trying to troubleshoot SPO performance issues.
- Know the code you implement - or at least profile it before release.

# References

SPTechCon Austin 2016

<http://www.sptechcon.com/>

"Technical diagrams for SharePoint Server"

<https://docs.microsoft.com/en-us/SharePoint/technical-reference/technical-diagrams>

"Page Diagnostics for SharePoint"

<https://chrome.google.com/webstore/detail/page-diagnostics-for-shar/inahogkhlkbbkjkaleonemeijihmfagi>

# References

“Azure Global Infrastructure”

<https://azure.microsoft.com/en-us/global-infrastructure/regions/>

“Deploying Office 365”

[http://video.ch9.ms/sessions/project/2014/PC215\\_Medford.pptx](http://video.ch9.ms/sessions/project/2014/PC215_Medford.pptx)

“Network and Migration Planning for Office 365”

<https://docs.microsoft.com/en-us/office365/enterprise/network-and-migration-planning>

# References

Fiddler web debugging proxy

<http://www.telerik.com/fiddler>

“FAQ – Certificates in Fiddler”

<http://www.telerik.com/blogs/faq---certificates-in-fiddler>

“Analyzing your webpage’s network traffic”

[https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/samples/dn255004\(v=vs.85\)](https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/samples/dn255004(v=vs.85))

# References

"Diagnosing performance issues with SharePoint Online"

<https://docs.microsoft.com/en-us/office365/enterprise/diagnosing-performance-issues-with-sharepoint-online>

"Image optimization for SharePoint Online classic publishing sites"

<https://docs.microsoft.com/en-us/office365/enterprise/image-optimization-for-sharepoint-online>

# References

Font Awesome

<https://fontawesome.github.io/Font-Awesome/>

“Use the Office 365 content delivery network with SharePoint Online”

<https://docs.microsoft.com/en-us/office365/enterprise/use-office-365-cdn-with-spo>

“Navigation options for SharePoint Online”

<https://docs.microsoft.com/en-us/office365/enterprise/navigation-options-for-sharepoint-online>

# References

“Using Content Search Web Part instead of Content Query Web Part to improve performance in SharePoint Online”

<https://docs.microsoft.com/en-us/office365/enterprise/using-content-search-web-part-instead-of-content-query-web-part-to-improve-perfo>

“Using jQuery Promises & Deferred”

<http://blog.blackninjaadojo.com/javascript/2019/02/27/using-jquery-promises-and-deferreds.html>



# References

“Lazy Load Plugin for jQuery”

[https://github.com/tuupola/jquery\\_lazyload](https://github.com/tuupola/jquery_lazyload)

“SPServices Example: UserProfileService.GetUserProfileByName”

<http://sympmarc.com/2012/02/15/spservices-example-userprofileservice-getuserprofilebyname/>

“get current user info using jquery, REST and csom”

<http://sharepoint.stackexchange.com/questions/124909/get-current-user-info-using-jquery-rest-and-csom>

# References

"Users, groups and the roles REST API reference"

[https://docs.microsoft.com/en-us/previous-versions/office/developer/sharepoint-rest-reference/dn531432\(v=office.15\)](https://docs.microsoft.com/en-us/previous-versions/office/developer/sharepoint-rest-reference/dn531432(v=office.15))

"Caching, You Ain't No Friend Of Mine"

<http://sharepointinterface.com/2016/01/31/caching-you-aint-no-friend-of-mine/>

# References

“Deprecation of Custom Code in Sandboxed Solutions”

<https://blogs.msdn.microsoft.com/sharepointdev/2014/01/14/deprecation-of-custom-code-in-sandboxed-solutions/>

“Using the Object Cache with SharePoint Online”

<https://docs.microsoft.com/en-us/office365/enterprise/using-the-object-cache-with-sharepoint-online>

# References

"Tune SharePoint Online performance"

<https://aka.ms/spoperformance>

"Use the Page Diagnostics tool for SharePoint Online"

<https://docs.microsoft.com/en-us/office365/enterprise/page-diagnostics-for-spo>

"Microsoft 365 Engineering Webcast Series: Optimizing Classic pages in SharePoint Online"

<https://microsoft365.eventbuilder.com/event/3675>

Thank you



# Sean P. McDonough

SharePoint & Office 365

Gearhead, Tinkerer, and

Microsoft MVP

**Email:** [sean@bitstreamfoundry.com](mailto:sean@bitstreamfoundry.com)

**Twitter:** [@spmcdonough](https://twitter.com/spmcdonough)

**Blog:** <http://SharePointInterface.com>

**About:** <http://about.me/spmcdonough>