

A Marketer's Guide to

BROWSER PUSH NOTIFICATION



- 00 Introduction
- 01 Anatomy of a Browser Push Message
- 02 What makes it a preferable choice among marketers
- 03 Asking for Permission-to-Push
- 04 Right way to ask for permission from user
- 05 What makes a Browser Push campaign successful
- 06 Use-cases
- 07 Limitation

Introduction

- A brief history
- Effectiveness

A BRIEF HISTORY

Browser push was first **introduced by Chrome in March 2015**. It enthused web publishers, as push messaging was a critical mobile feature that was missing from the web.

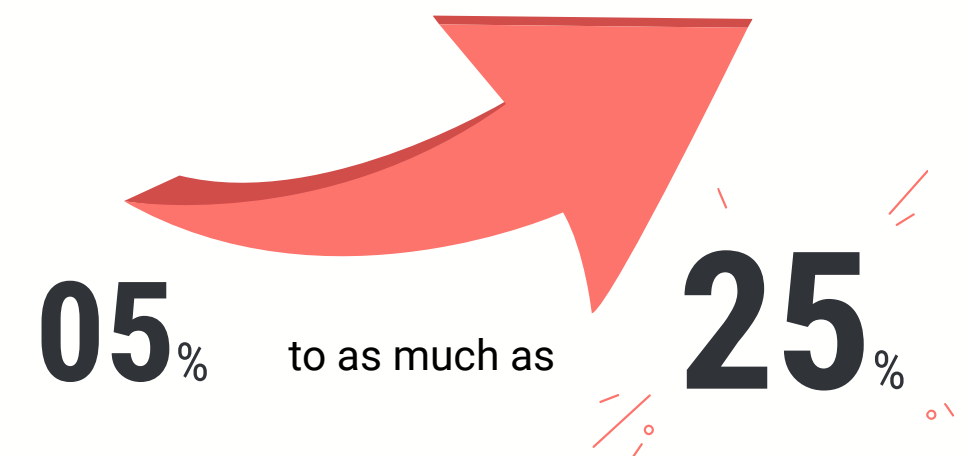
Another reason why browser push, also called web push, gained momentum is because

- a) it gave marketers the opportunity to establish a dedicated opt-in audience and
- b) control the information feed the way they want to, like they can with email, mobile push, text etc.

Upon its release, it witnessed an enormous adoption by the publishers, just as its mobile counterpart, especially from the digital news sector.



How effective has web push been on the conversion side?


















That's the **CTR** that our clients are currently reporting for web push campaigns. Once we have a reliable data bank for web push, we will conduct and publish a study on its macro influence on conversion.

NOT ALL PLATFORMS SUPPORT BROWSER PUSH

Browser push is still at its nascent stage. Following Chrome, **Firefox introduced it in early 2016**. However, not every platform has embraced this development and incorporated browser push, the most prominent of them being iOS.

Yes, at the time of writing this book, iOS still doesn't support browser push. As of July 2017, this is the table listing the browsers which support push notification.

| OS |  Windows |  Android |  iOS |  Mac OS |  Ubuntu |
|--------------|---|---|---|---|---|
| Supported by |   Chrome 42+, Firefox 44+, |  Chrome 42+ |  No browser support |    Chrome 42+, Firefox 44+, Safari Mavericks |   Chrome 42+, Firefox 44+, |



Anatomy of a Web Push Message

- Elements of a browser push (Chrome vs Safari)

ELEMENTS OF BROWSER PUSH

A typical browser push looks like this:

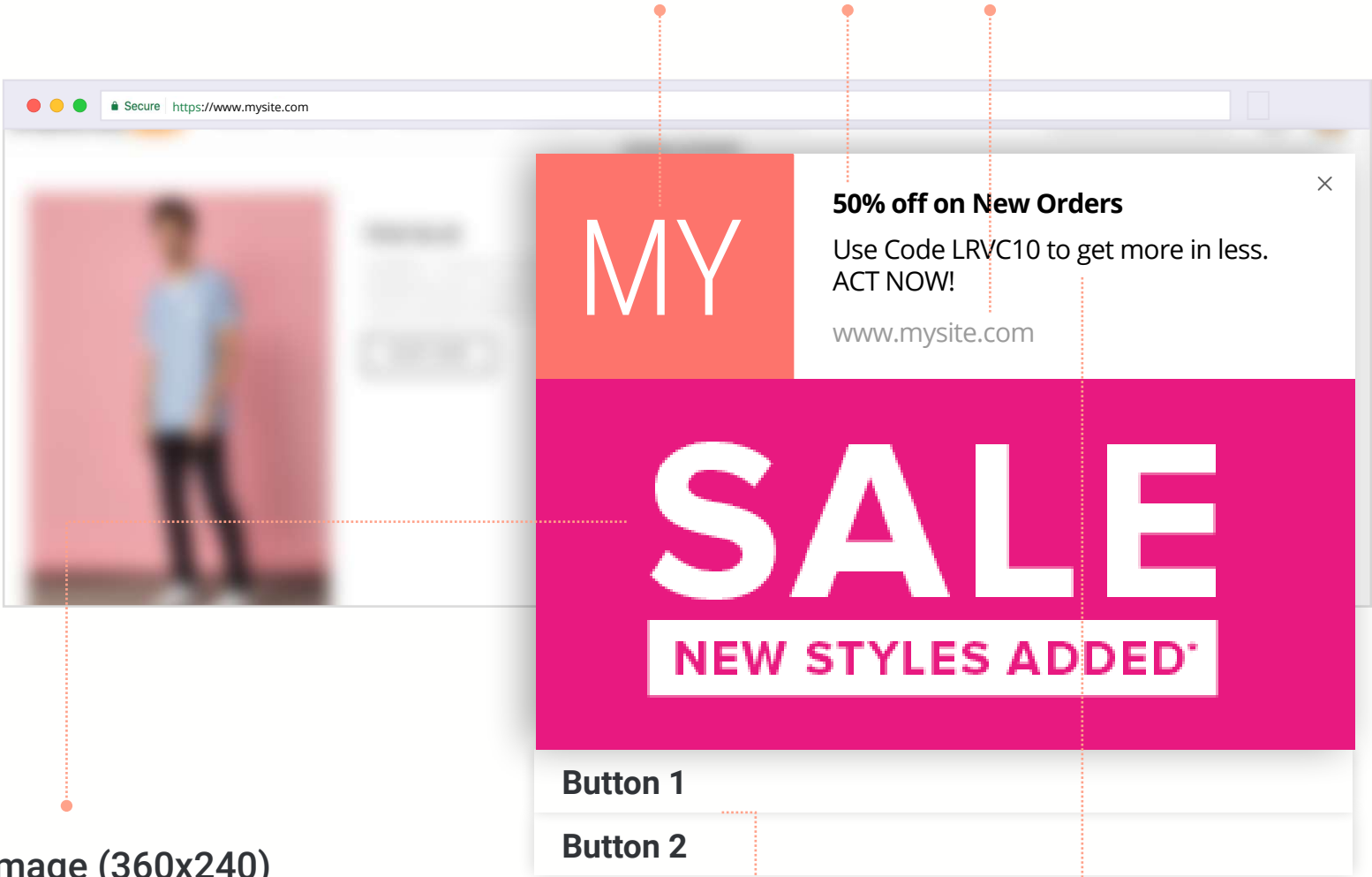


Image (360x240)

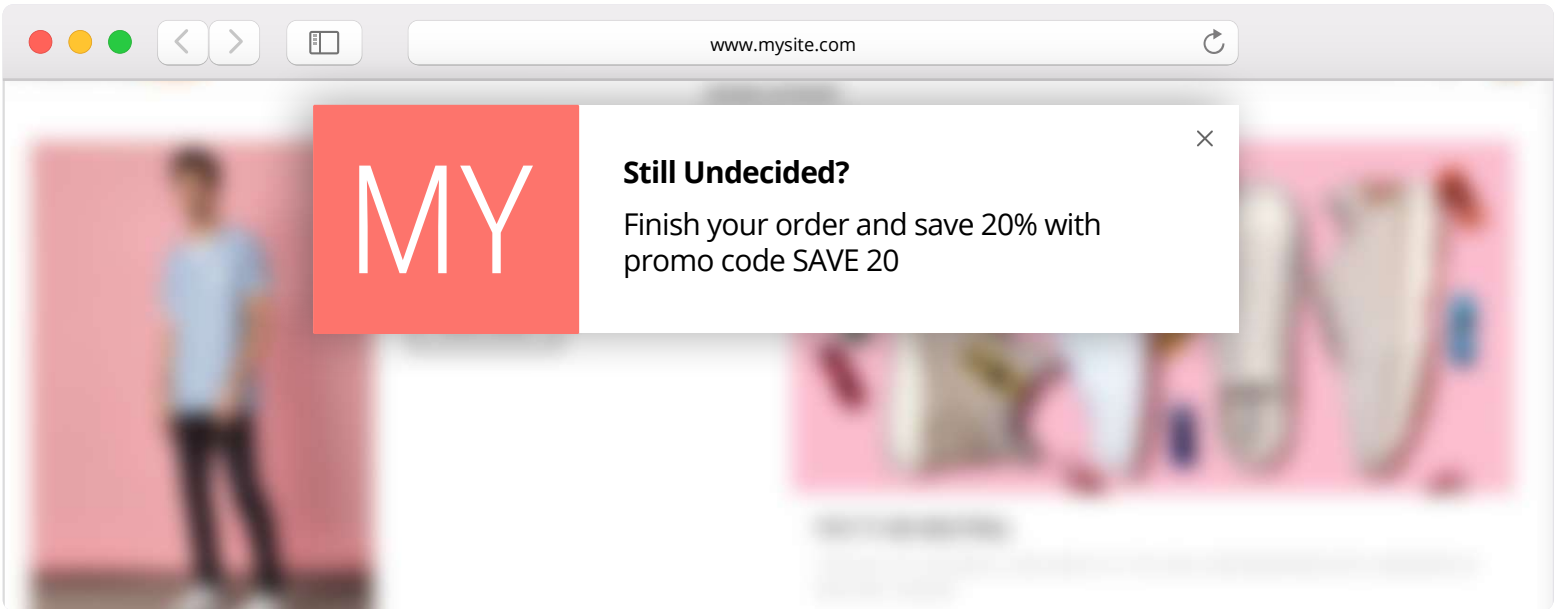
This is the latest release in Chrome 56,

Buttons

Add upto two buttons. Available only in Chrome 48+

Description

You can incorporate variables in the message.



A Safari push notification doesn't show the url of the sender nor does it support images or multiple CTAs. Firefox push is is similar to that of Safari's except that it shows the sender's url like Chrome.

What makes browser push a preferred choice among marketers?

- Easy to setup
- Kills the need of having an app just to send push notification
- Assured message delivery

What makes browser push a preferred choice among marketers?

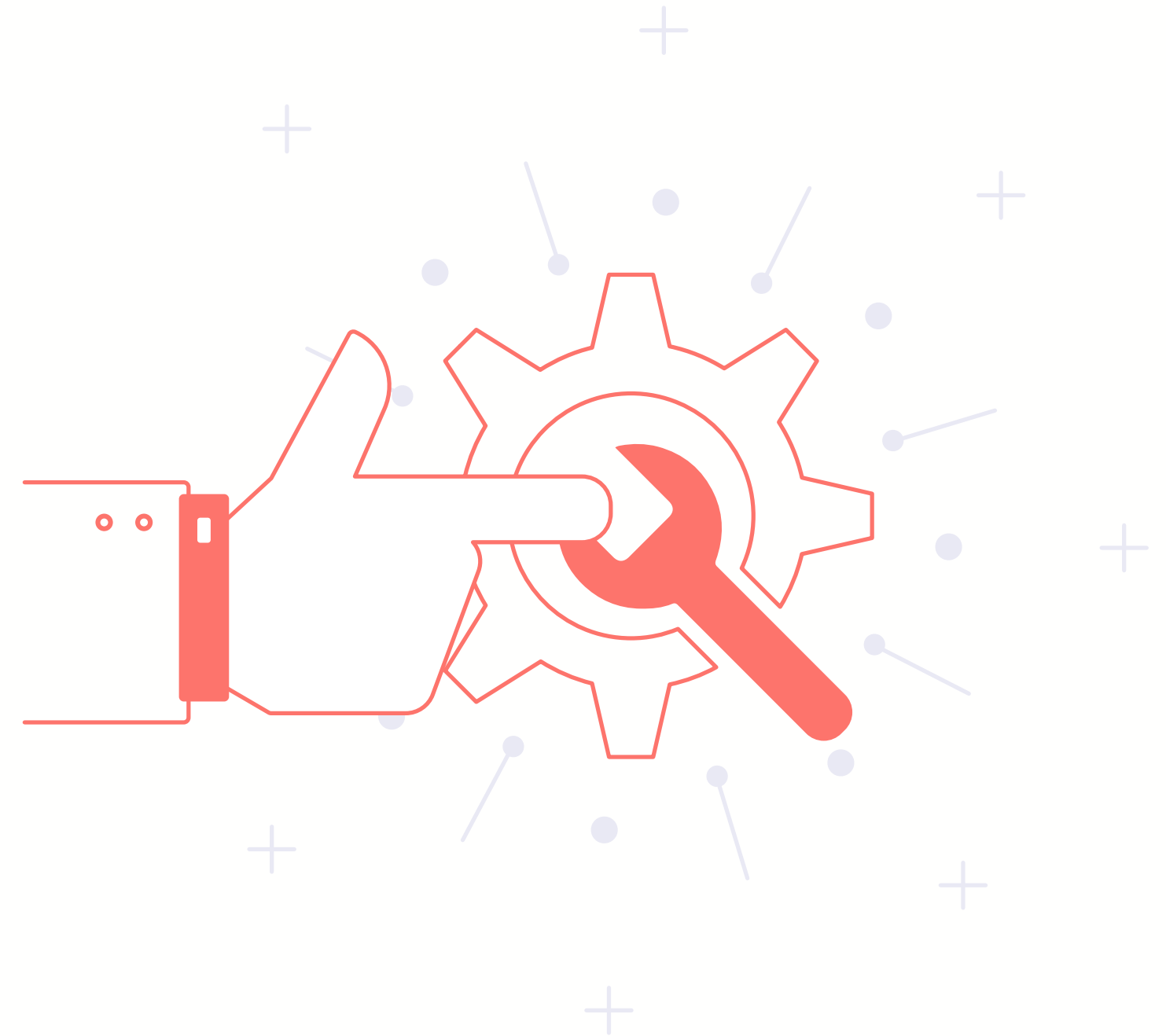
BENEFITS OF BROWSER PUSH

1. Easy to setup

You could opt for a browser push service provider, which sets up your account in a jiffy (really), or you choose the hard way of manually implementing it in your web-app. And even if you go the hard way, your browser push notifications **will be up and running in less than a day.**

2. Assured message delivery

Browser push doesn't run the risk of ending up in the spam folder. Neither can it be blocked by **ad blockers**. It shows right up on your browser, even when the website isn't open. On mobile it shows up in the notification tray, just like mobile push notification, even when the browser is not running.



What makes browser push a preferred choice among marketers?

BENEFITS OF BROWSER PUSH

3. Kills the need of having an app just to send push notification

Since their inception, push notifications have given enormous ROI to publishers. For instance, the New York Times claimed that push messages drive almost **60% of its traffic**, which is a massive number if you consider that it's just one channel.

This makes launching an app to send notifications a lucrative proposition for publishers. However, not every company has the need or resources to develop a mobile app. Browser push serves a suitable alternative to these entities.



Asking for permission-to-push from user

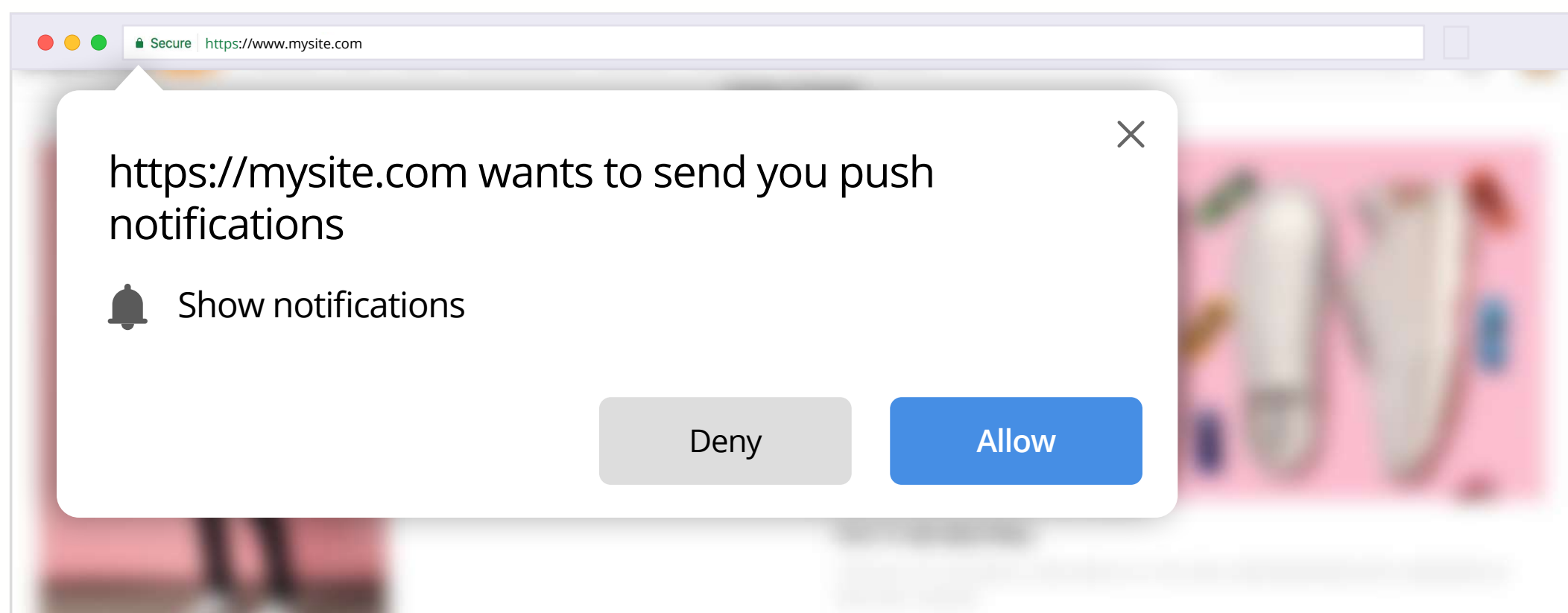
- Browser push permission flow
- Drawback

Asking for permission-to-push from user

PUSH NOTIFICATION PERMISSION FLOW

In order to be able to push web notifications to the user,

a publisher has to take explicit permission from the user, via a system dialog box like shown below:



Upon receiving the consent of the user, the publisher would be able to push notifications to the user's browser.

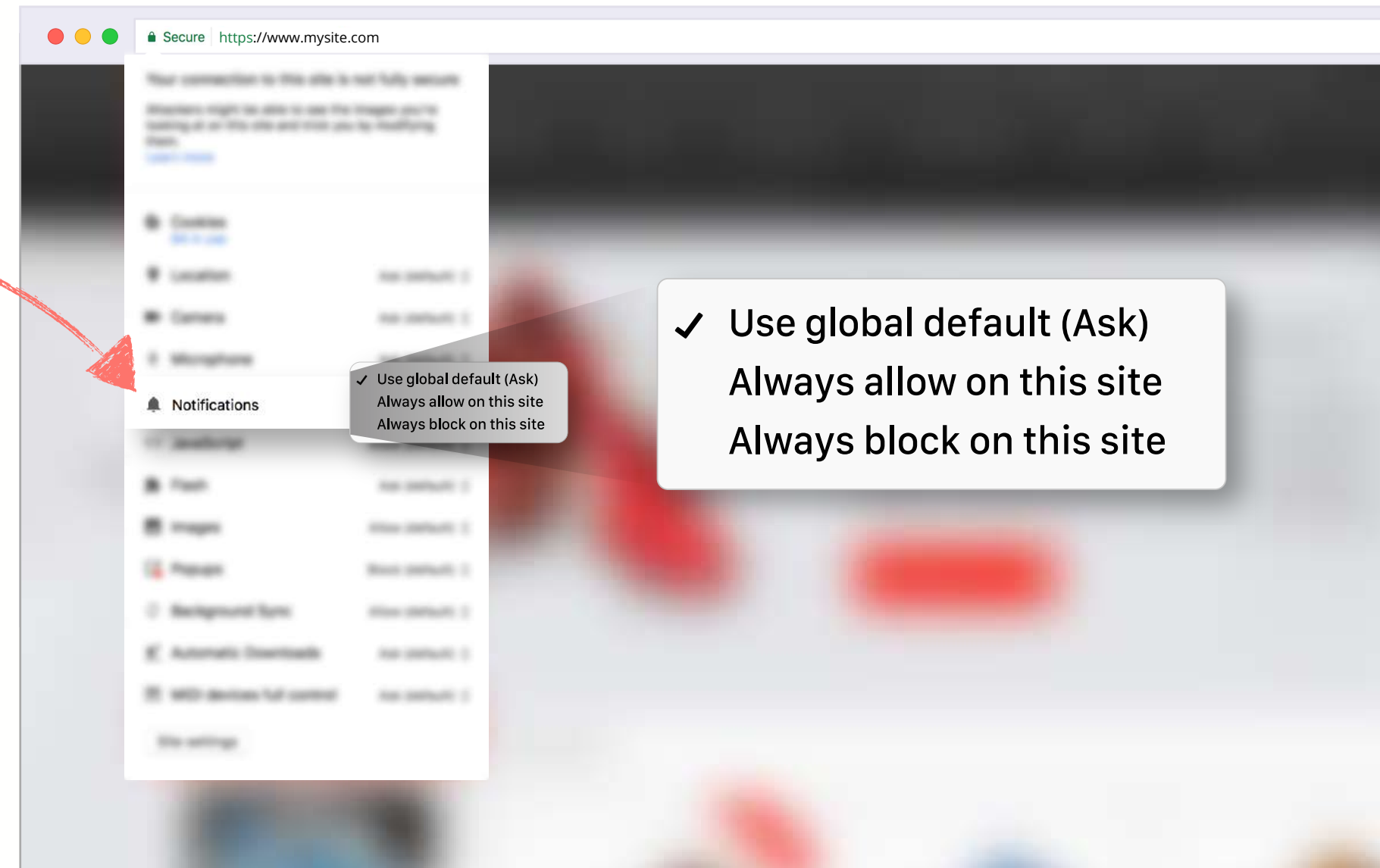
Asking for permission-to-push from user

DRAWBACK

Now, amidst the immense potential that browser push presents to marketers, it also comes with a challenging drawback.

Push API gives the users only three options to engage with the prompt:

- **Default** – When the user ignores the prompt by pressing ‘escape’ or by canceling the notification
- **Granted** – When the user clicks on ‘Allow’
- **Denied** – When the user clicks on ‘Block’

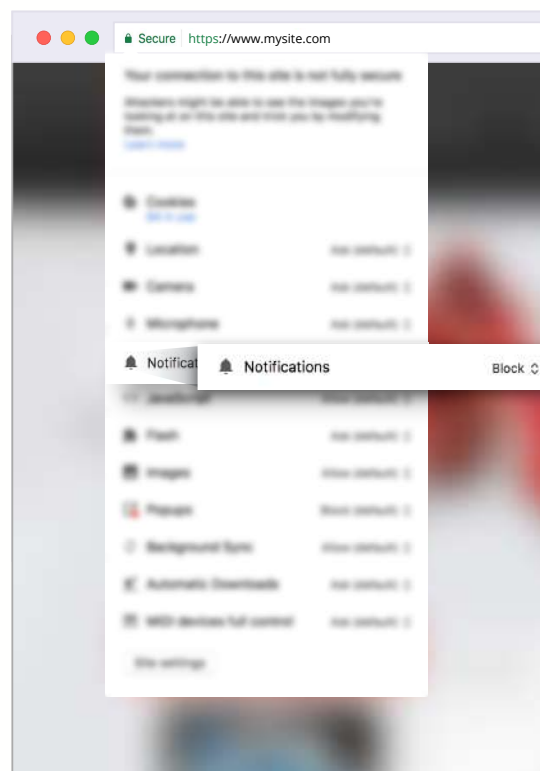


Asking for permission-to-push from user

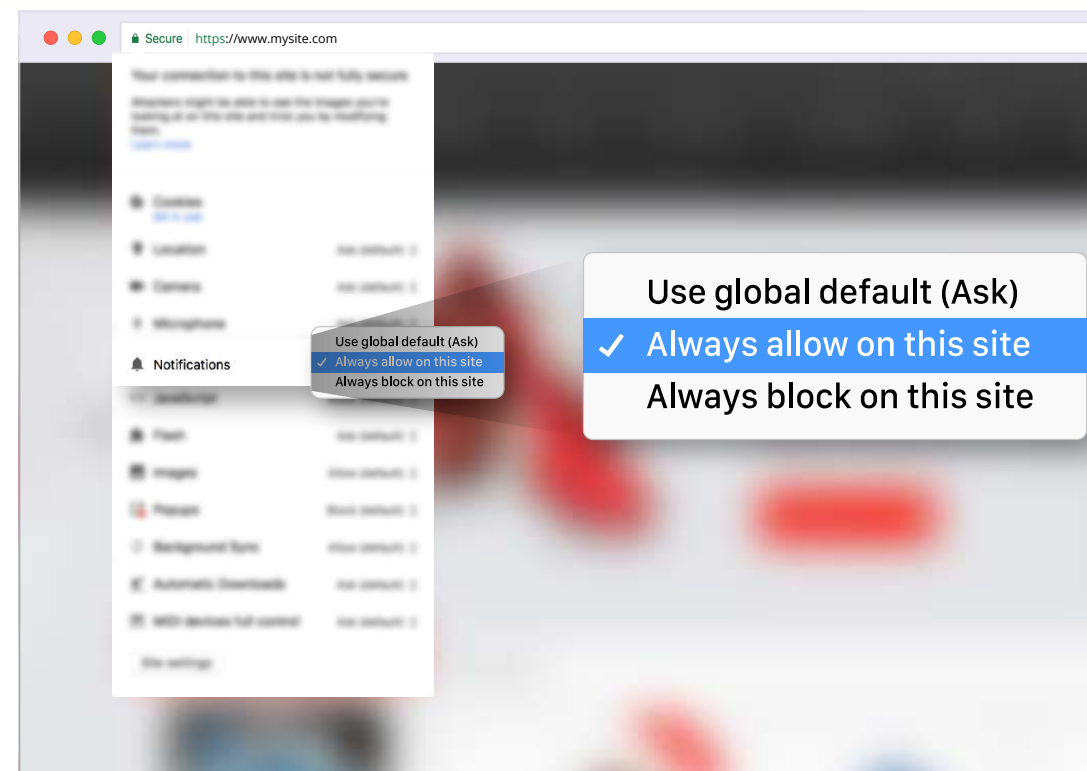
DRAWBACK

However, the disadvantage is that once the user clicks on **'Deny' in the opt-in prompt**, the browser API doesn't allow you to launch the opt-in prompt again.

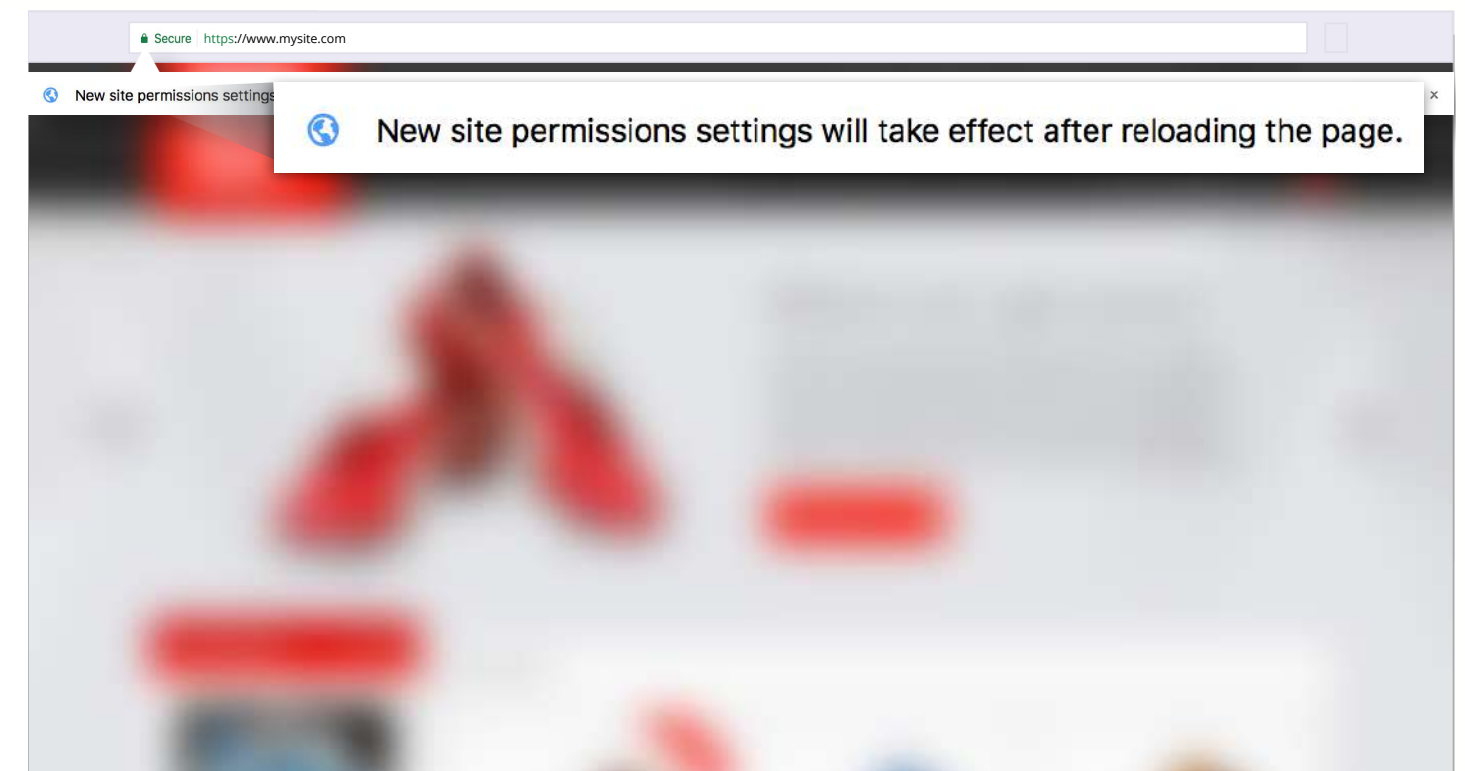
Now, for the user to reverse his decision, he has to go through a complex series of steps, almost 3, as shown below to manually allow the notification. So, technically a user clicking on 'deny' entails the loss of the web push channel forever, as a means to engage with him.



1. Click on 'Site Information'



2. Go to "Notifications" and change from "Always block" to "Always show"



3. 'Reload' the page

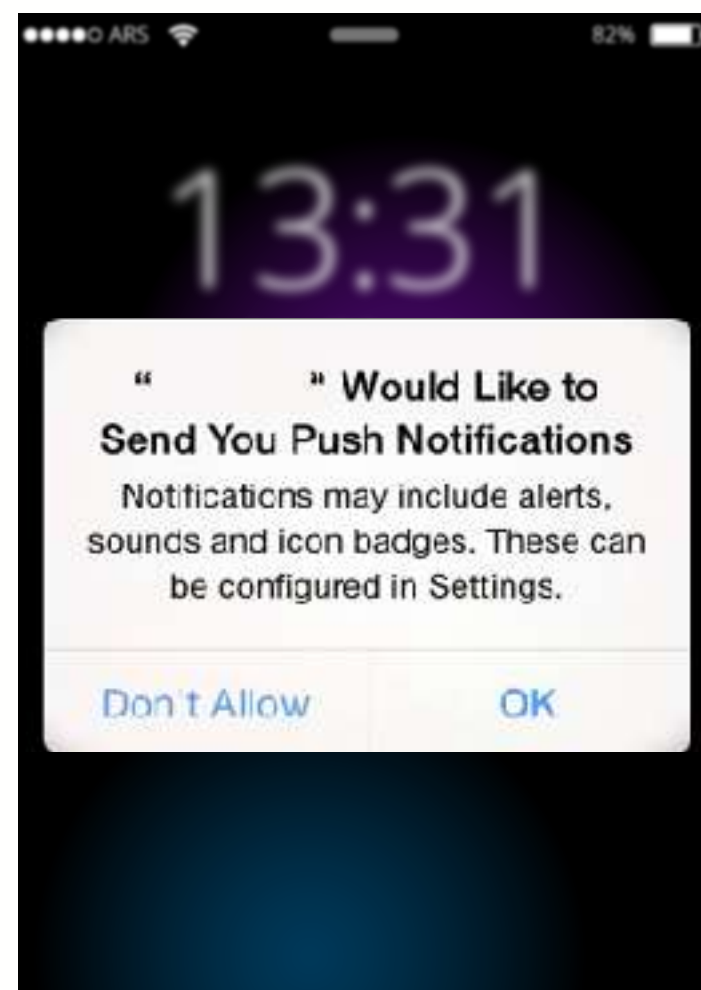
Right way to ask for permission-to-push from user

- Introduce a pre-permission popup
- Trigger the system prompt via click on a button or widget
- Workaround for non-SSL secured websites (HTTP)

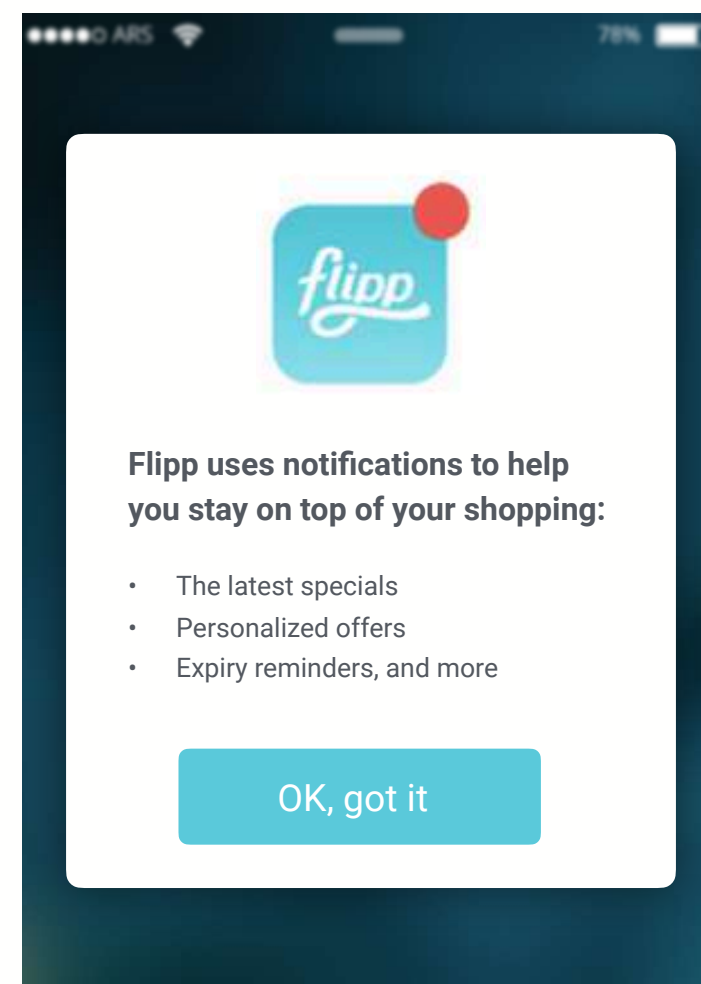
Right way to ask for permission-to-push from user

INTRODUCE A PRE-PERMISSION POPUP

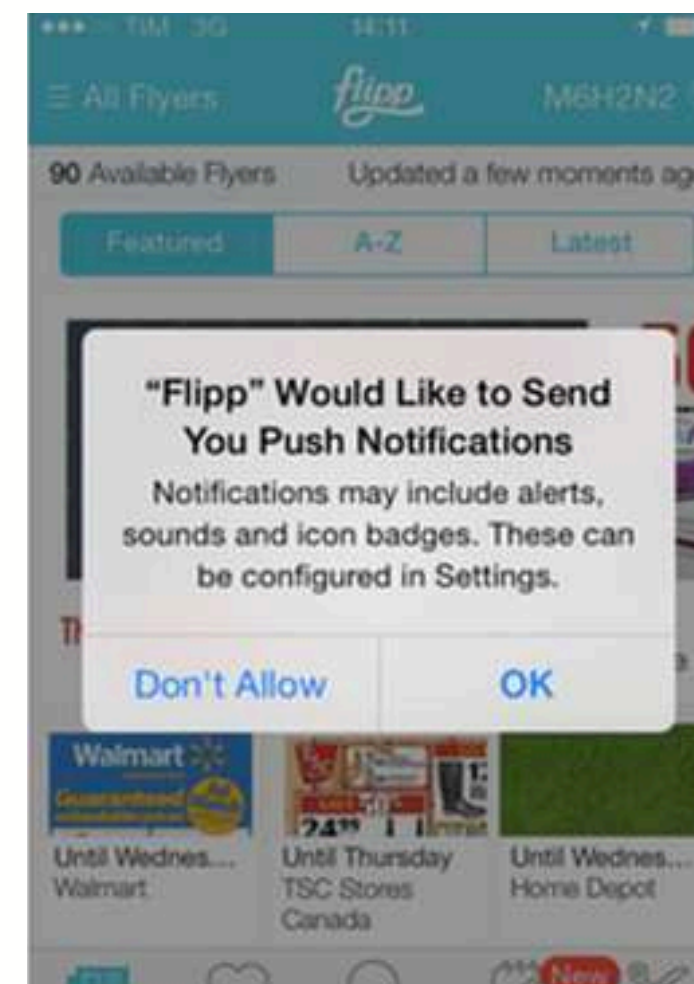
The permission setting of browser push notifications could be vaguely compared to that of iOS wherein you have just one attempt to launch the push notification permission dialog box. In iOS, to reduce the possibility of 'don't allow', developers camouflage the system opt-in with a pre-permission dialog box.



Without a pre-push popup



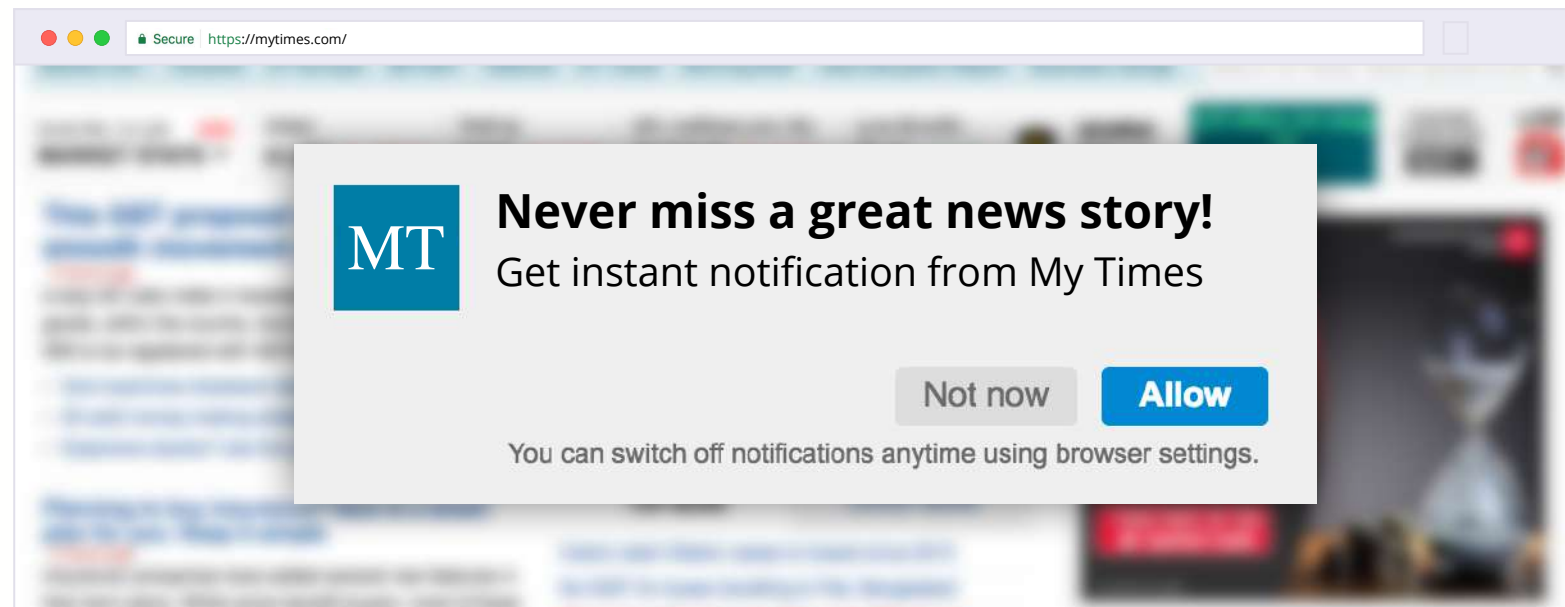
Camouflaging with a pre-push popup



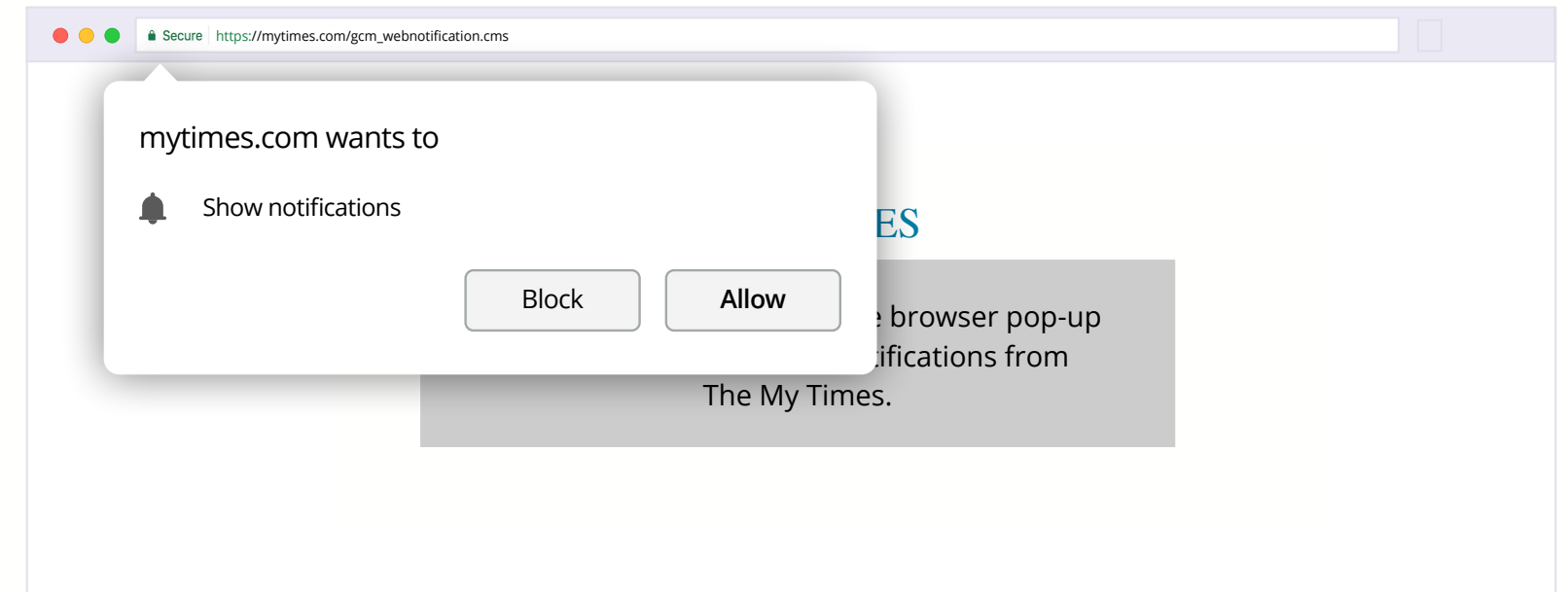
Right way to ask for permission to push' from user

INTRODUCE A PRE-PERMISSION POPUP

We are going to adopt the same trick for browser push notifications too. Instead of directly launching the system permission dialog box, we are going to take the consent of the user via a pre-permission popup that would nudge them to opt in the next step.



Pre-permission popup before system opt-in



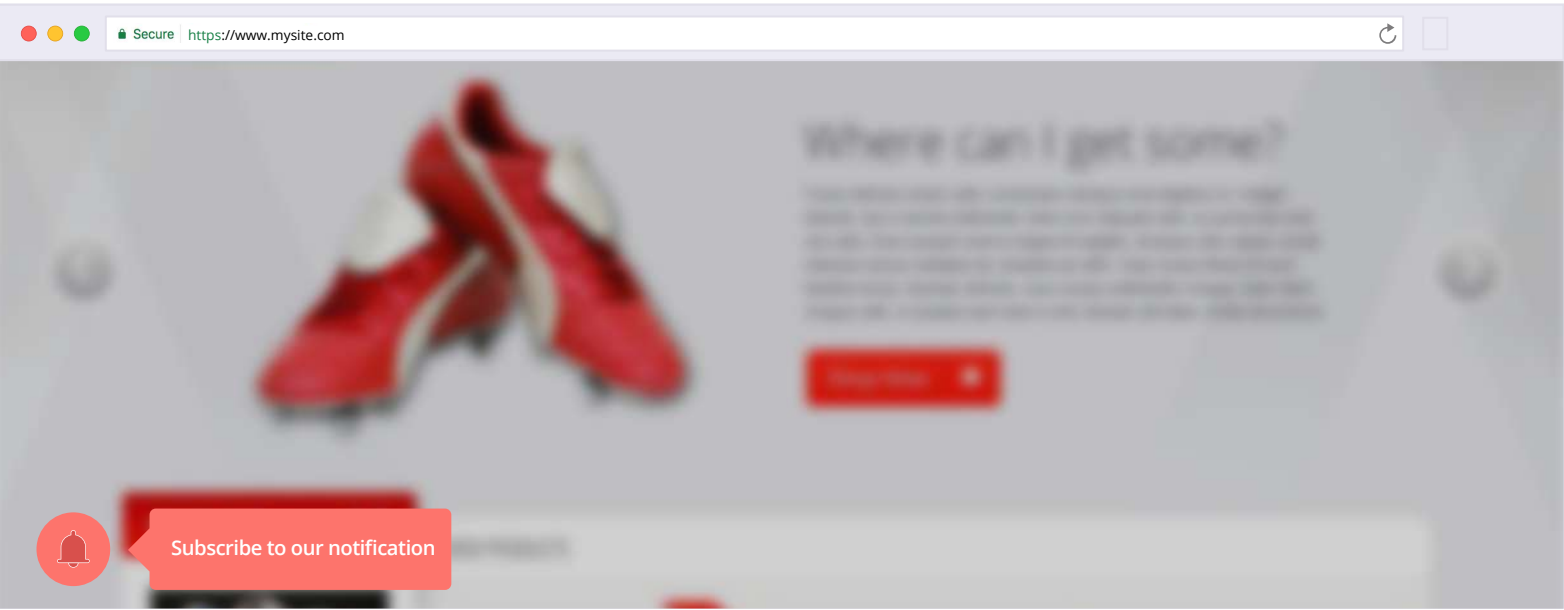
System opt-in if the user clicks on 'Allow' in previous step

Right way to ask for permission-to-push from user

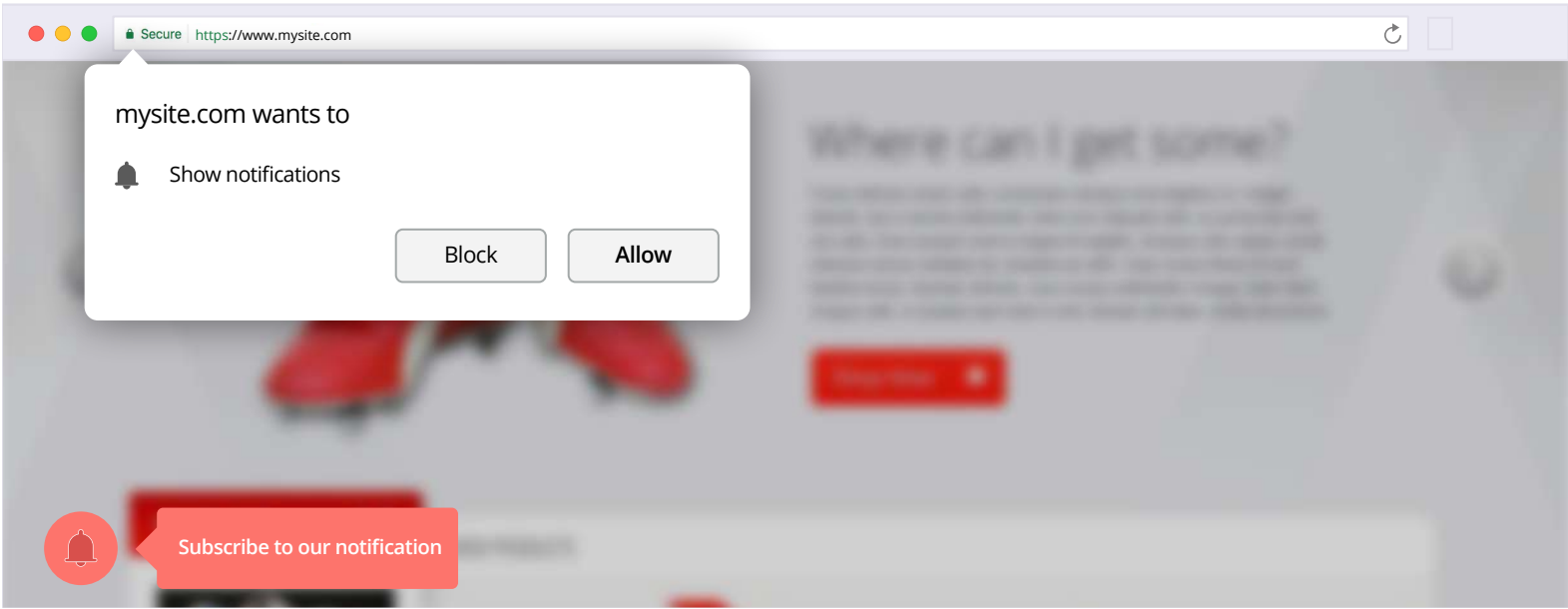
TRIGGER 'SYSTEM PROMPT' VIA CLICK ON A BUTTON OR WIDGET

Another safe way of getting permission is to place a CTA button or widget on your website which triggers the permission dialog box upon click.

This is a naturally fail-proof method because it is entirely driven by the user’s actions.



User clicks on the widget at the bottom left



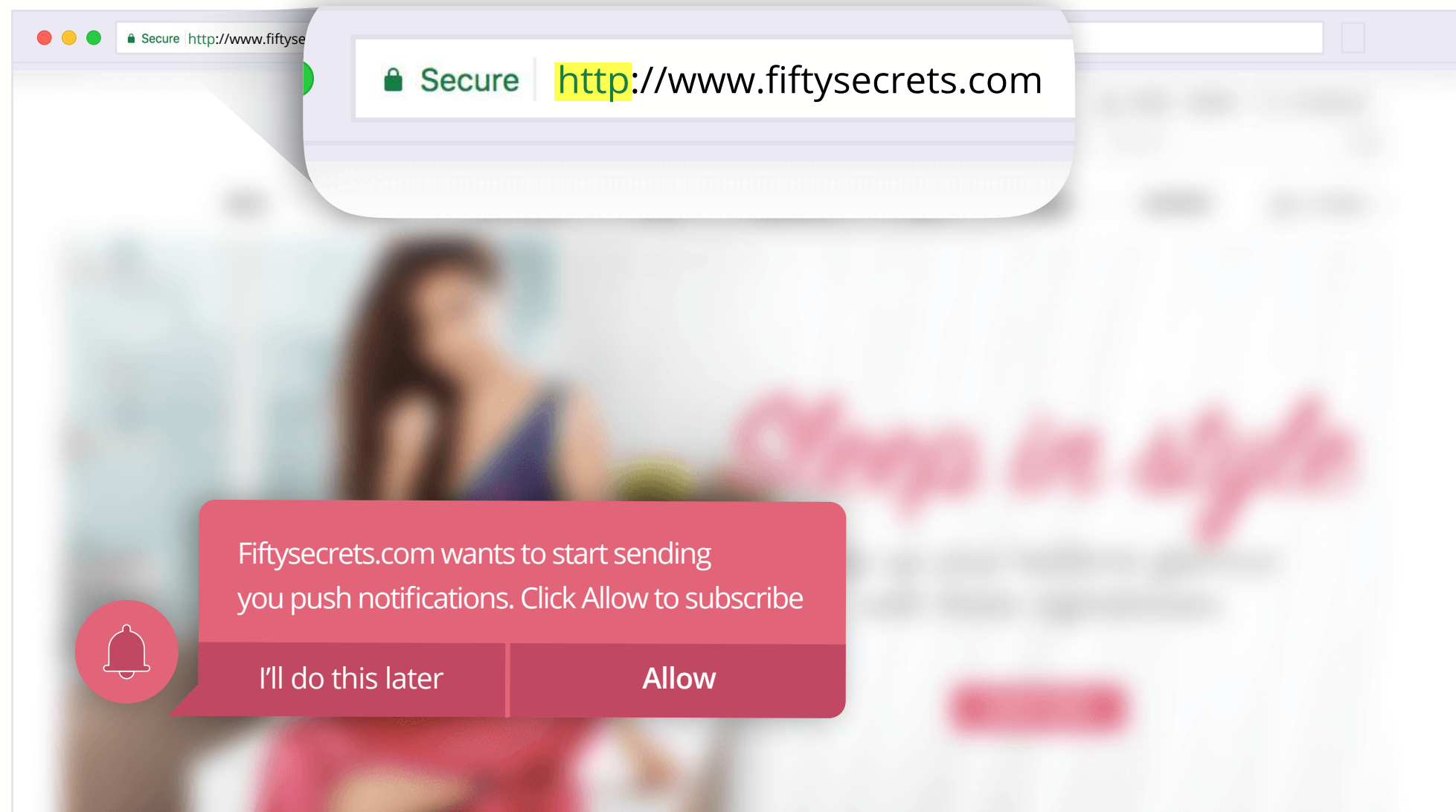
System opt-in is displayed

Right way to ask for permission-to-push from user

WORKAROUND FOR NON-SSL SECURED WEBSITES (HTTP)

The previous two tricks that we learned are possible only for https websites because non-SSL secured websites cannot trigger browser push in the first place. These websites have to employ a service provider, like WebEngage.

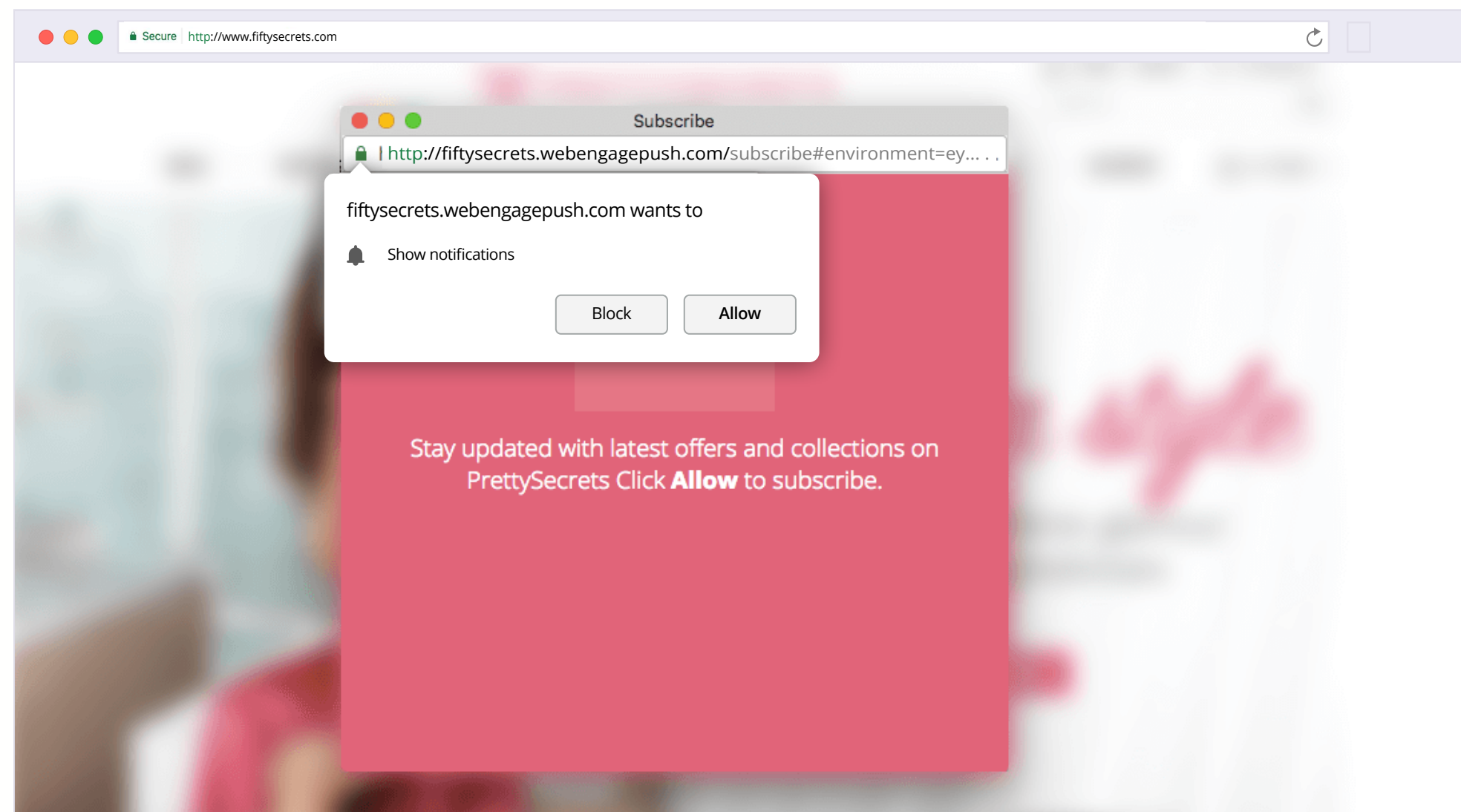
Let me illustrate an example: First, service provider pops a custom notification like the one below.



Right way to ask for permission-to-push from user

WORKAROUND FOR NON-SSL SECURED WEBSITES (HTTP)

If the user clicks on 'Allow', then the service provider (WebEngage) is going to open an opt-in window (prettysecrets.webengagepush.com) which in turn is going to pop the browser prompt. This means that if the user opts in, then the notification would be sent to the user from the service provider managed domain (prettysecrets.webengagepush.com), rather natively from the client's domain like it was in the case of SSL secured websites.



What makes a browser push campaign successful?

- Segmentation
- Journey Orchestration Engine
- Event-triggered messaging

What makes a browser push campaign successful?

SEGMENTATION

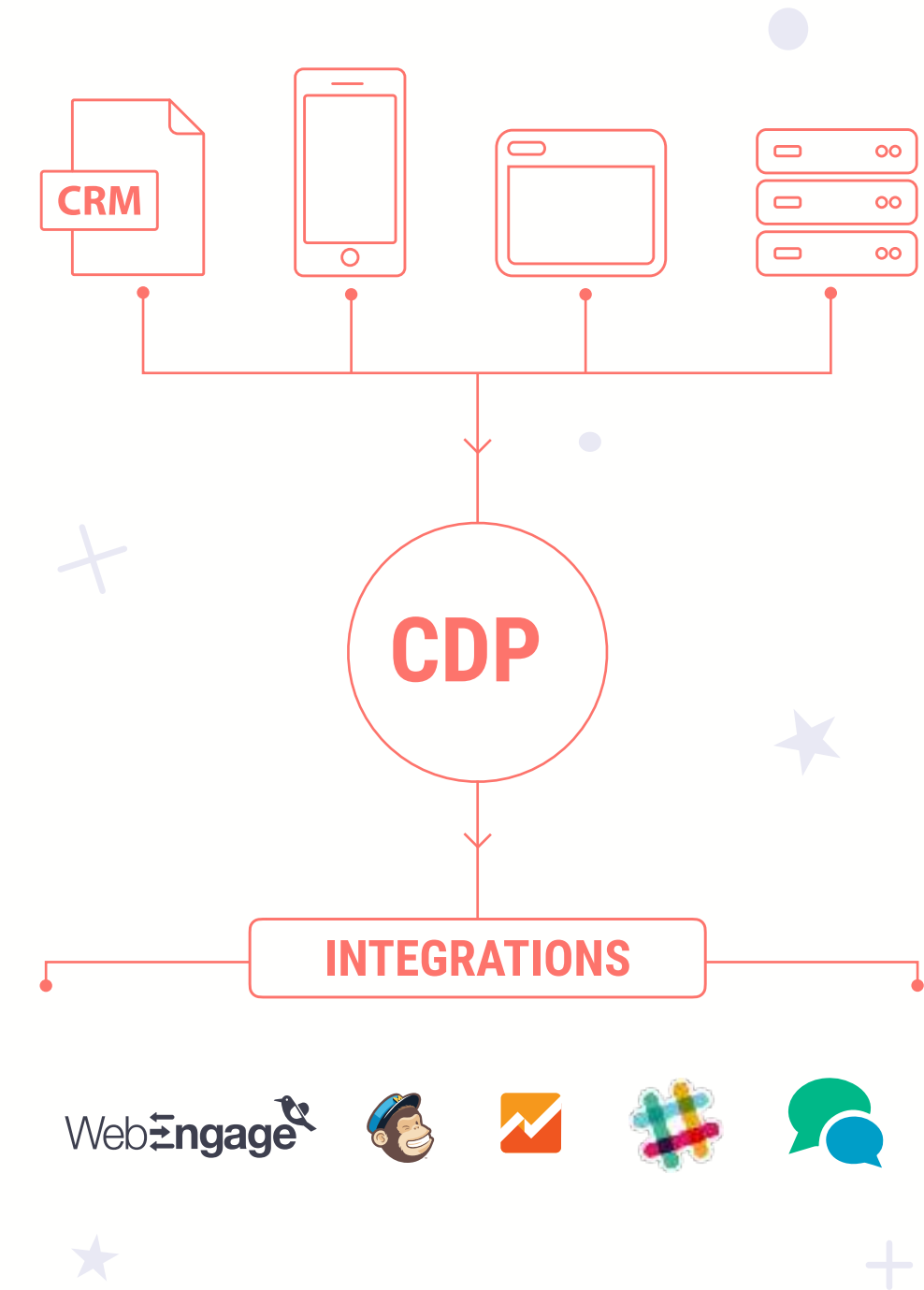
"Segmentation is useless," said absolutely no one ever. Yet...

1. To effectively do segmentation, the primary requirement is: data. **B2C enterprises need a platform that acts as a central repository of consumer interaction across all devices and channels.** This is where Customer Data Platform (CDP) comes into the picture.

CDP is a unified central database that pulls and stores data from all your integrations. Subsequently, this data is then made available to all the integrations on your system. Refer to this brief by [Martechtoday](#) for better understanding. Also explore [CDP Institute](#) run by Daavid Raab.

2. **Secondly, invest in a push engine that lets you collect data beyond system events.** Your push engine should allow you to collect user's custom in-app actions that let you target them based on their in-app behavior.

For instance, collecting system attributes, such as device id, browser version, OS version, language, etc., is fine but if you need to target users who have spent more than 30 seconds on your checkout page, your system should enable you to do that.



What makes a browser push campaign successful?

INVEST IN A JOURNEY ORCHESTRATION ENGINE (JOE)

Refrain from treating web push as a standalone channel.

Marketing campaigns individually across channels are very primitive. If you still haven't adopted a system of running coordinated messaging across all channels, museum guys are looking for you.

To accomplish that, invest in a journey orchestration engine and **build workflows for your users across multiple channels.**

This effort will be supplemented by a customer data platform, as it will collect the data and JOE will decide how to treat customers across devices and channels. WebEngage's 'Journey Designer' would be an excellent choice if you are at it. Check out few other alternatives on **David Raab's blog.**



Watch the above video to see how WebEngage's JOE works

What makes a browser push campaign successful?

EVENT-TRIGGERED MESSAGING

Triggered communication is more effective than an average one for the obvious reasons. We already discussed how close you are to losing browser push as a channel forever, so the need to keep your messaging contextual is hugely imperative, if not downright critical.

To enable that, you first have to identify **events** that correspond to **high purchasing behavior**. For instance, a 'payment reminder' 30 minutes after the incomplete transaction would always amount to sales uplift.

Identifying those events is another challenge in itself. Several tools are incorporating predictive technology to make suggestions to users. Manually, you have to map the customer journey and shortlist high value events.

In the next section we are going to see multiple use-cases where web-push has been automated against an event trigger.

Who should get this email?

Send this email to

Segment "Send quick reminder" contain users

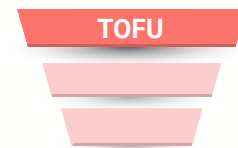
Who are "Repeat visitors"

Who performed event "incomplete_checkout" **30 mins ago** minutes ago

Use-cases

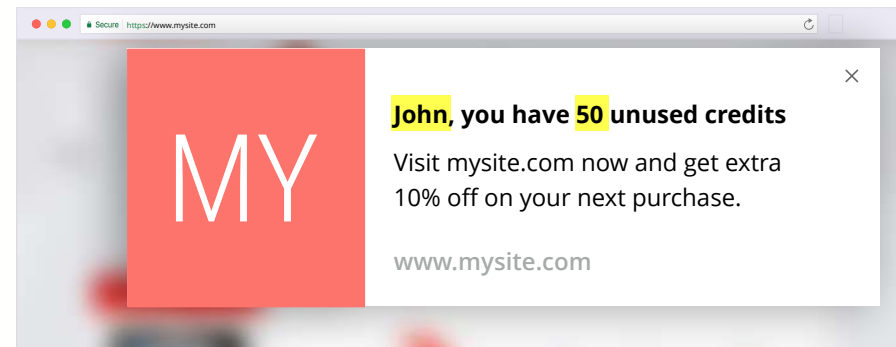
- E-commerce use-cases
- News and Media
- OTA

E-COMMERCE



Use Case

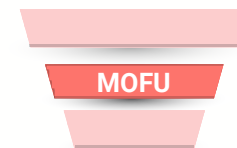
Reactivation campaign



Trigger- Last login 7 days ago

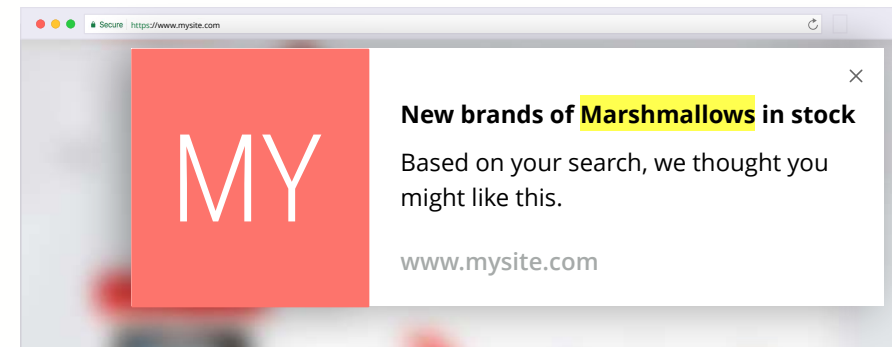
Segment- inactive users

Personalization tags- [first_name],
[credit_amount]



Use Case

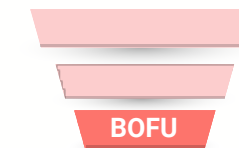
Abandoned search



Trigger- No transaction post search

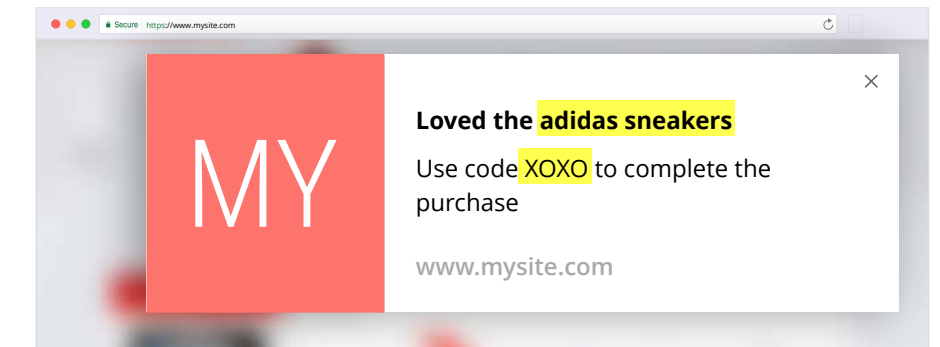
Segment- inactive users

Personalization tags- [searched_item]



Use Case

Cart abandonment



Trigger- 30 minutes post cart abandonment

Segment- users with incomplete transaction

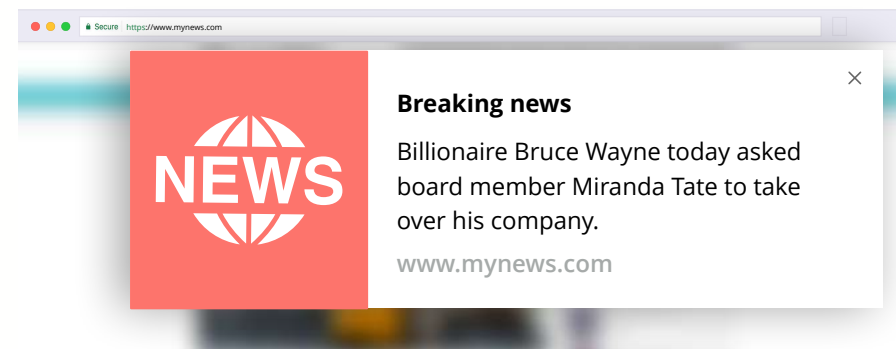
Personalization tags- [abandoned_product],
[discount_coupon]

NEWS AND MEDIA

TOFU

Use Case

News update



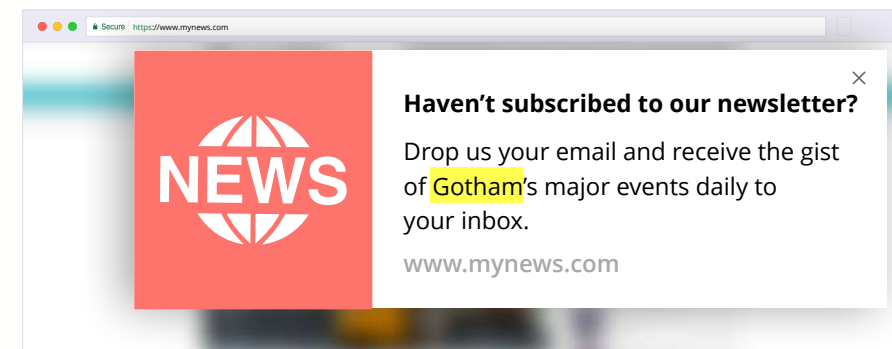
Trigger- Breaking news

Segment- All subscribers

MOFU

Use Case

Nudge for newsletter subscription



Trigger- Engagement score crosses threshold

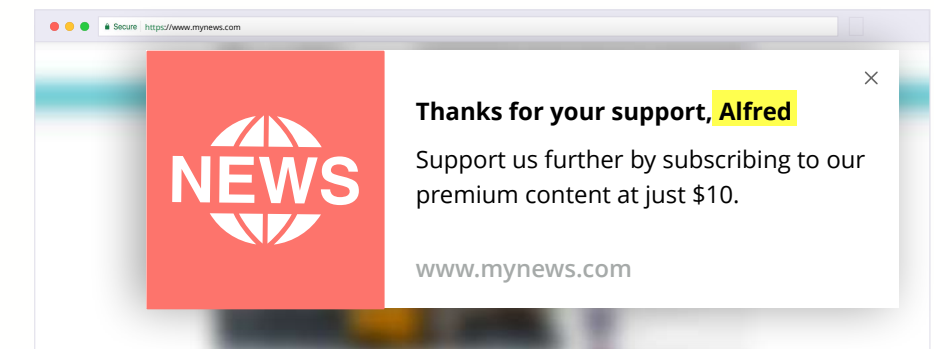
Segment- Not subscribed to newsletter

Personalization tags- [subs_city]

BOFU

Use Case

Promote premium content

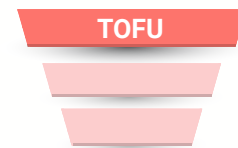


Trigger- Not visited 'Premium' landing page

Segment- All engaged users

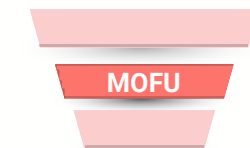
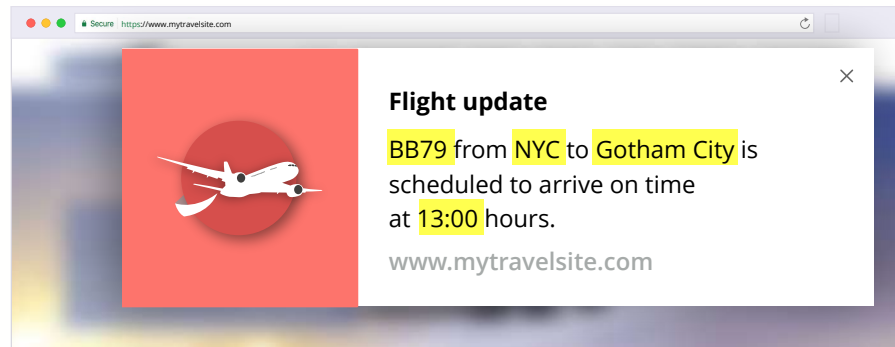
Personalization tags- [first_name]

OTA



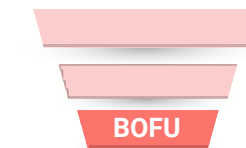
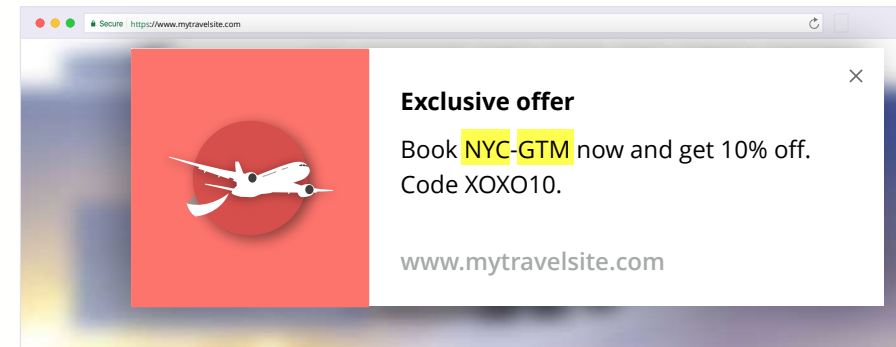
Use Case

Flight status update



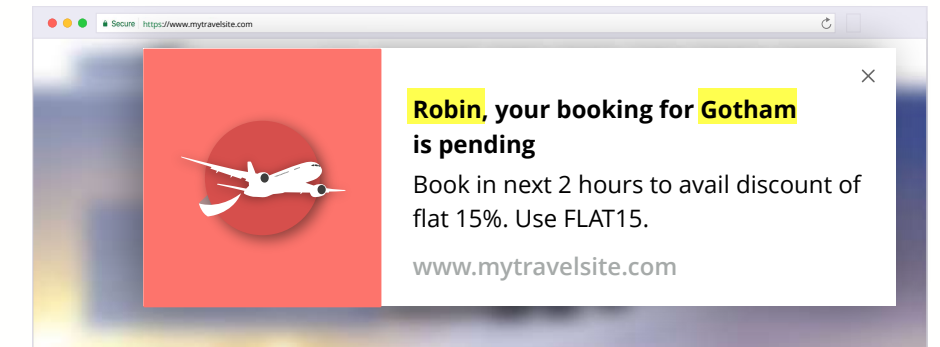
Use Case

Promotion



Use Case

Recover booking abandonment



Trigger- Update on the flight status

Segment- All relevant travelers

Personalization tags- [flight_id], [dep_city], [arr_city], [dep_time]

Trigger- Incomplete transaction

Segment- Users who searched for NYC-GTM

Personalization tags- [dep_city], [arr_city]

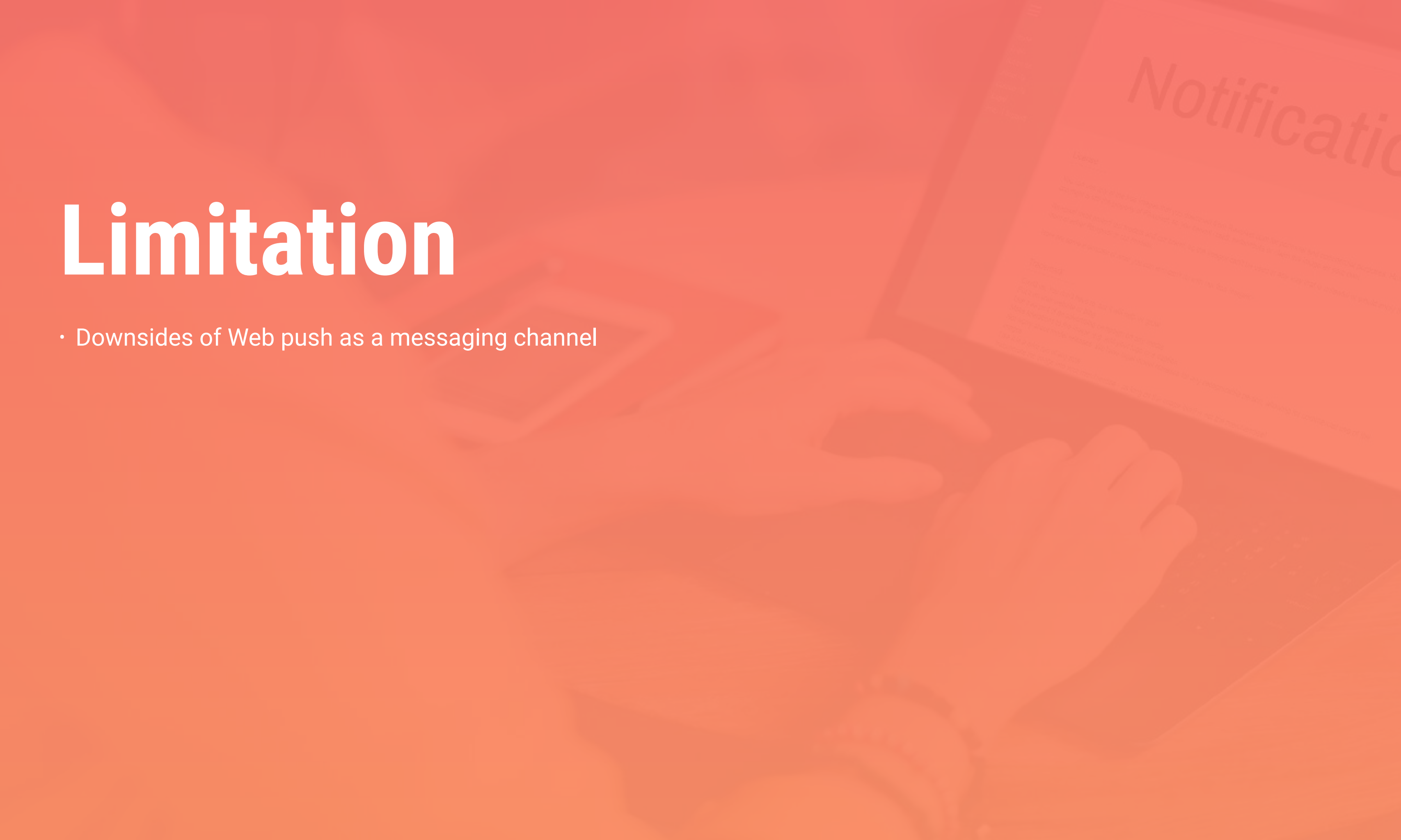
Trigger- 30 mins after user abandons the booking

Segment- Abandoned cart users

Personalization tags- [first_name], [dep_city]

Limitation

- Downsides of Web push as a messaging channel



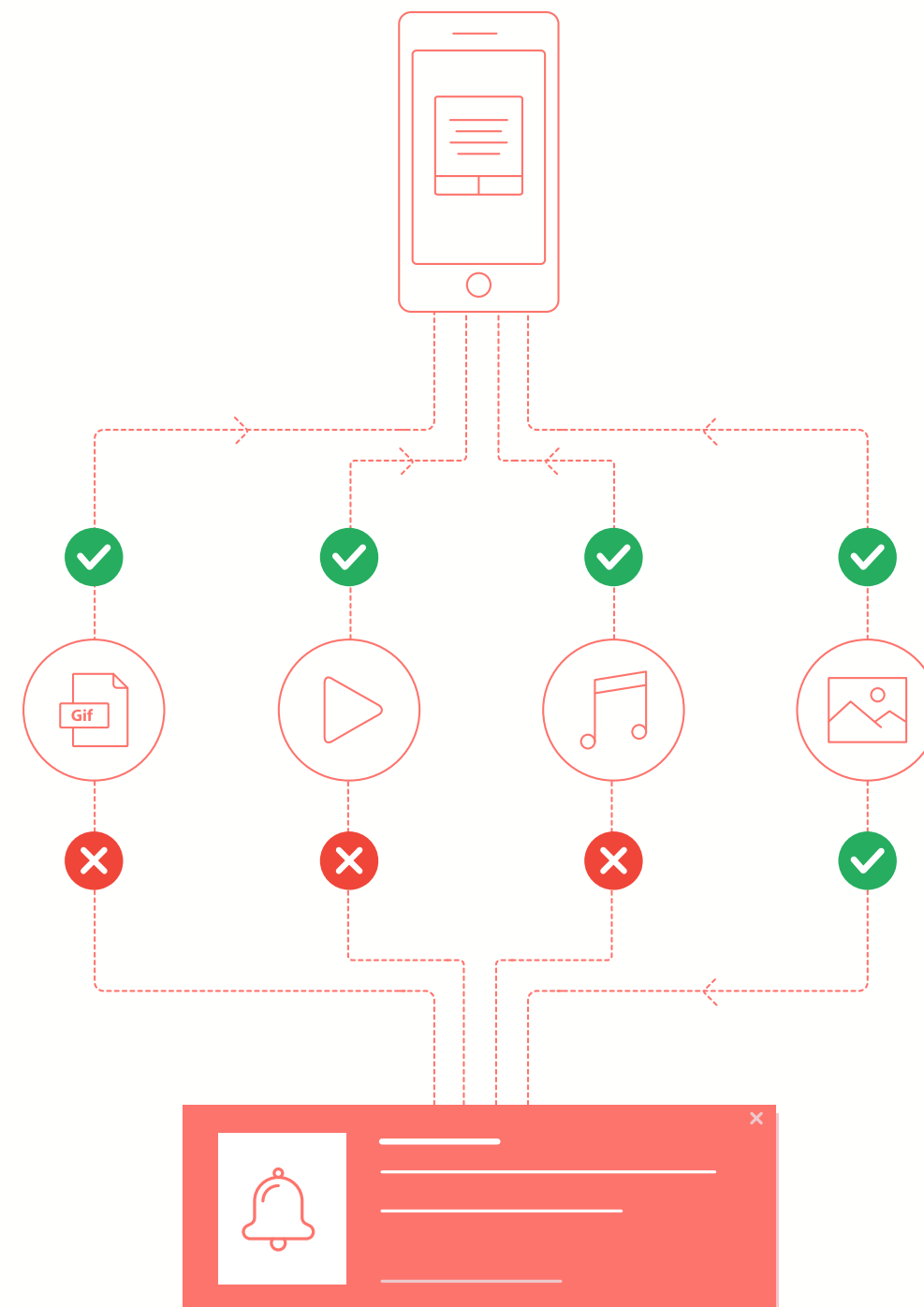
LIMITATION

Finally, it should be noted that browser push notification as a channel is still in its nascent stage and has several limitations, some of which I have listed below:

1. By using default OS notification, browsers can have their not-interacted-with push messages reside in the notification centre. With the notification in there, users can interact with them whenever they wish to.

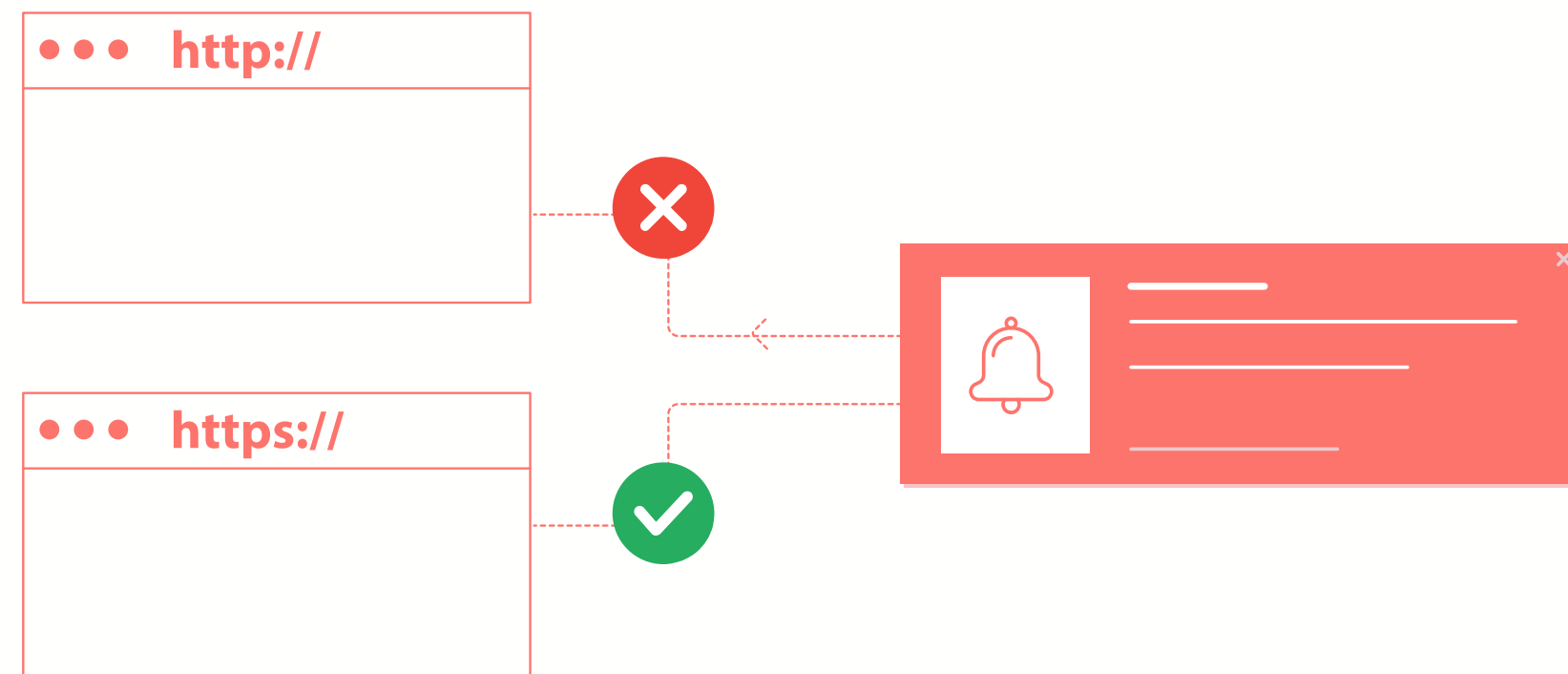
Problem is, Chrome doesn't use default OS notification in Windows like it does in MacOS (Firefox does in both). So web push messages in Chrome cannot be reside in the notification centre thereby decreasing the probability of user engaging with them.

2. It is not feature-rich like its mobile counterpart. In mobile push notification you could add media attachments like gif, video, audio files etc. But in web push, the maximum you can do is add an image which is also a latest addition only available in Chrome.



LIMITATION

3. Web Push only works with SSL secured websites. Obviously this standard is there to curb the security threats but since the majority of sites on the web are non-SSL, this still counts as a limitation.
4. On desktop, web push can be delivered only if the browser is open. Thankfully this limitation is only there in desktop, as, on mobile devices, push notification can be delivered as and when you want it as long as the browser is not uninstalled.
5. Web Push doesn't work on iOS. A big limitation whose size depends on how big your iOS base is.



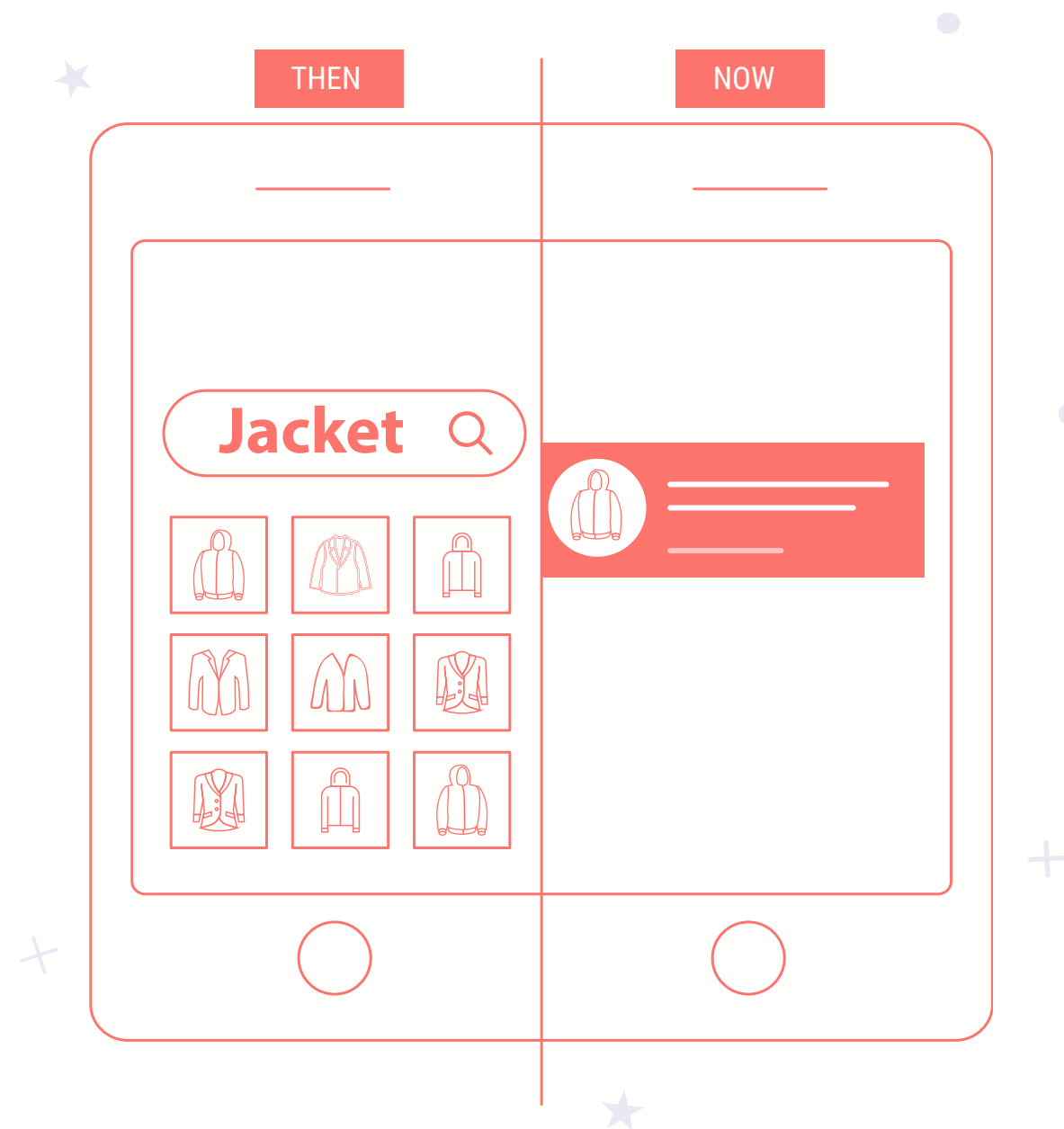
CONCLUSION

Push notifications are changing our online behavior, probably at a scale that is not immediately apparent, but certainly there.

It is directing our engagement with the internet. For instance recall how until 2-3 years ago, you used to scour all your content from just one source: Google. You determined what you needed and then you pulled it out from Google - which is essentially just one app. But now, as the world is transitioning from desktop to mobile - which is swamped with gazillion apps - the data is determining what content may interest you and it is then pushed to you via notifications.

The paradigm has changed, or perhaps is changing rapidly, from 'pull' to 'push'.

Push notifications, be it web or mobile, are an incredible opportunity to B2C marketers. We hope that this ebook comes useful in helping you leverage that.





**Also check out our most popular
ebook so far:
Accelerate your B2C sales funnel
with multi-channel marketing
campaigns**

Get the Free Guide

42,000+ online businesses use WebEngage everyday to improve their user engagement and retention.

Talk to Us Today



WebEngage is a multi-channel user engagement platform which automates communication across users' life-cycle. It enables you to connect with them via. Web Messages (notification, survey and feedback), In-App Messages, Push Notifications, Emails and Text Messages.

+1 (408) 890-2392



support@webengage.com



@WebEngage



/WebEngage