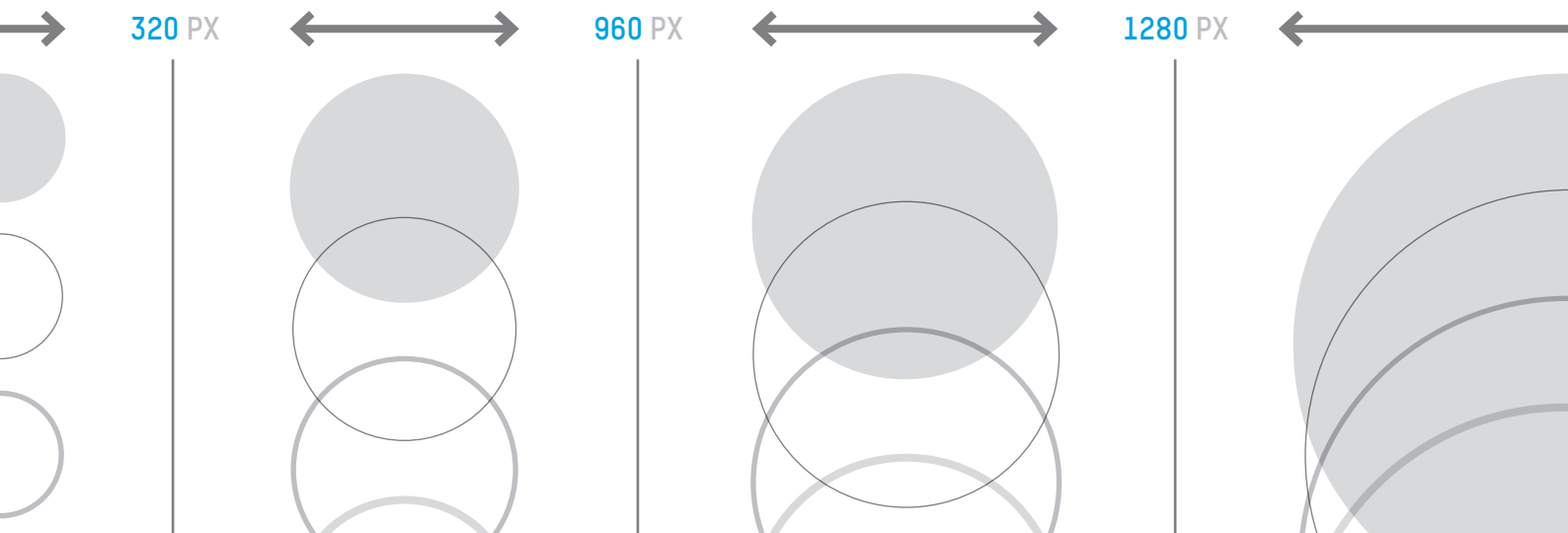
RESPONSIVE

Why **Responsive Design** Isn't Good Enough...

...And **How To Fix** It

MOOVWEB



Executive Summary

The mobile revolution isn't near, it's here. Our world is full of hundreds of new post-PC devices, not just with different screens dimensions and resolutions, but with increasingly different features and input methods. But as we race into the post-PC era, companies are lagging behind. Few have a holistic Web strategy that includes a post-PC vision and as a result the potential of surging mobile traffic is not being realized. Early approaches that integrate mobile with existing sites and applications using custom APIs or Web services have proven to be costly and ineffective. And cookie cutter solutions don't reflect the unique aspects of your business. Yet there has to be a better way to protect the billions of dollars already invested in existing Web stacks.

RESPONSIVE WEB DESIGN

There's lots of buzz around techniques like Responsive Web Design (RWD), which have made it easier to leverage a single set of content and code more effectively across different channels, but how do organizations deliver compelling mobile experiences with this model? It turns out that while RWD allows for lots of code sharing, this approach has significant challenges that companies need to consider before investing.

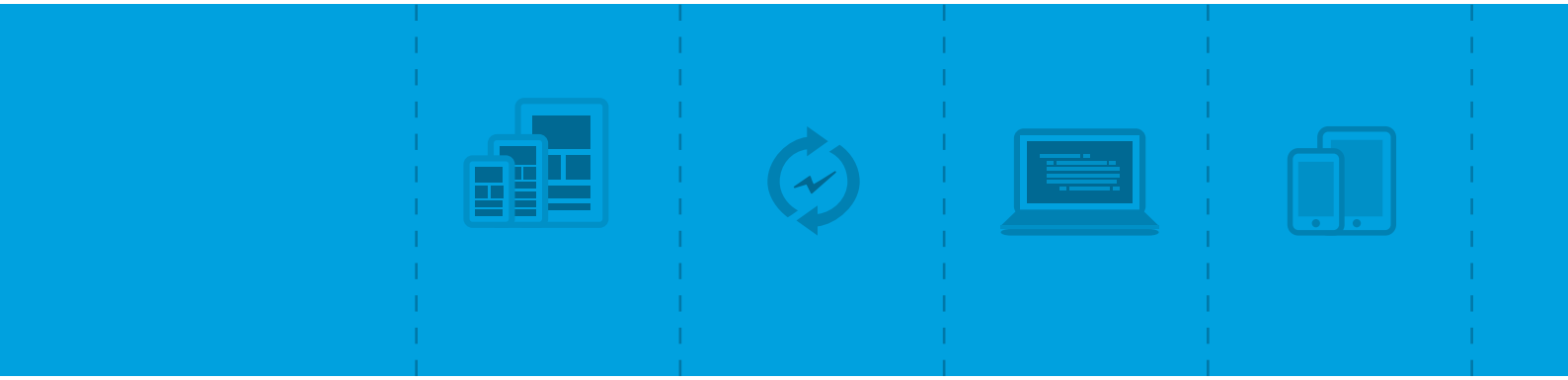
This white paper will dig into the RWD hype, discuss the pros and cons of this approach and propose a next generation approach that has the best of RWD, without the issues that RWD imposes.

PROS

- All code in one place
- Only one URL per page
- Adapts to changing screen sizes

CONS

- Difficult to optimize individual experiences
- Heavy, slow-loading pages
- Requires front-end rebuild
- Not Post-PC ready



Device **Proliferation**

THE GREAT PROBLEM OF THE POST-PC ERA

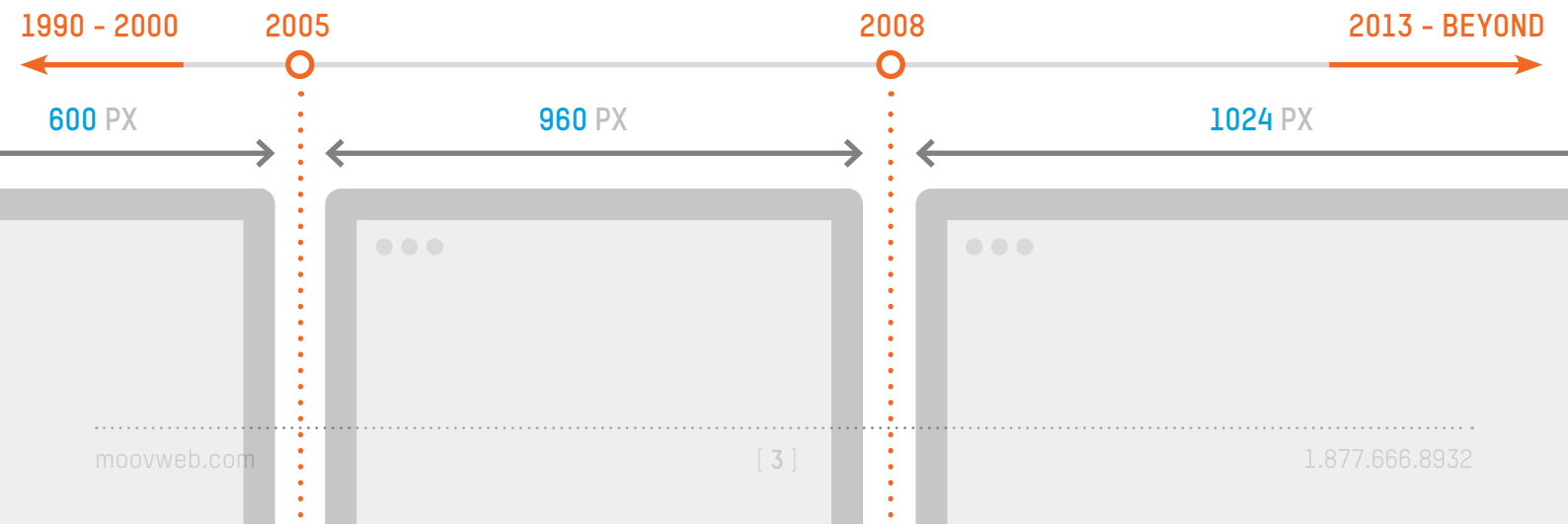
IN THE EARLY DAYS of the web information was presented to its users in the familiar form of pages. It was a digital representation of physical pages from the world of print. And since web pages were modeled after journals or books their pages adopted the print-based practices of physical design.

WITH THE ADVENT OF MOBILE AND TABLET BROWSING, **SCREENS OF ANY SIZE** MUST BE ACCOUNTED FOR.

But in the realm of books or newspapers design is done in fixed sizes. This constraint affects all subsequent design decisions. The content has to fit the paper. You can't have text running off the side of a piece of paper.
In the past, web designers behaved like print designers, imposing their own constraints by restricting the design of their site to fixed widths and sizes. They simply designed with the resolution of the common monitor in mind. In 2000 that meant pages were typically no wider than 600 pixels, while today pages are designed for 960 or 1024 pixel widths.

However the web is different because the canvas isn't set. On a desktop the browser window is often resized from large to small, covering a continuous range of widths. And with the advent of mobile and tablet browsing, browsers of any size must be accounted for.

The great problem that the post-PC era presents is device proliferation. New devices with vastly different resolutions, dimensions (landscape or portrait), and capabilities are accessing the web.



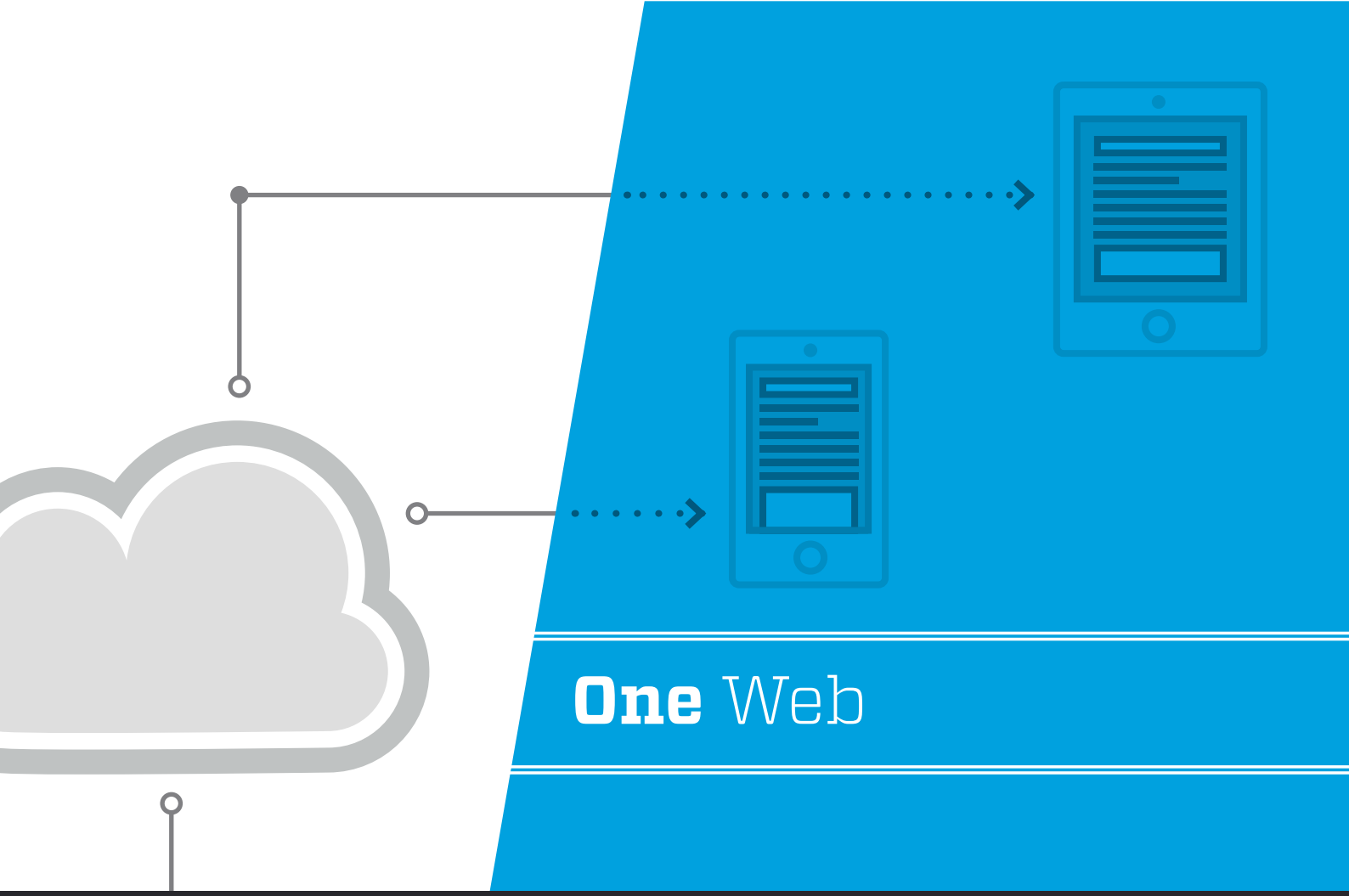
The **Splinternet**

WHEN FACED WITH meeting the mobile challenge, companies found that they could reuse their existing web services to create a mobile site. Multiple websites and even mobile apps can be built on top of a web services layer. Each of these web assets can hook into the existing web services and recreate all the rich features from the original website. In many cases, the web services can be used in a completely different way, allowing for the creation of highly optimized and rich features tailored to a specific device's experience.

However, as the web becomes increasingly complex and interactive, more logic and behavior piles up above the web services layer. Now, maintaining one website is getting more difficult every day, and the problem gets worse when you have to keep more than one in sync with the first.

web services are very good at creating rich, interactive experiences, but when it comes to reusability in the post-PC era, they are poor at leveraging the work that has already been done above the web services layer. A great deal of application logic, and everything on the front-end, must be redeveloped from scratch if you want to create a second website. As the feature set of the desktop site evolves, it becomes increasingly difficult and expensive to keep features on the mobile or tablet site on par with that of the desktop.

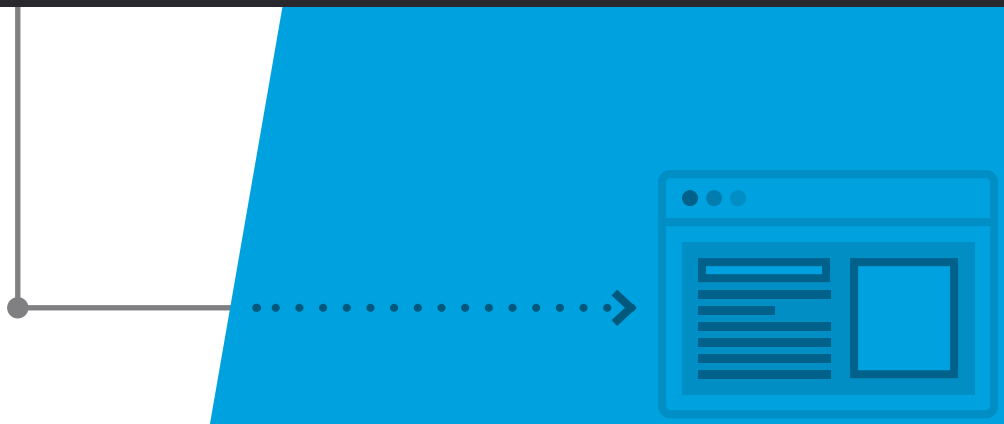
Trying to solve device proliferation using web services has led to stack fragmentation, sometimes also referred to as the "splinternet". The entire technology stack that originally supported one website has suddenly fragmented into multiple, independent stacks, all sitting on top of the web services. Companies building multiple websites on top of their web services now have to maintain separate and distinct applications and front-ends, one for each website.



One Web

SINCE STACK fragmentation has been a problem for a long time, a movement has begun to combat it. That movement is called One Web. It is the fundamental opposite to stack fragmentation, and a call to action to search for a way to reduce the incredible burden and complexity of managing multiple fragments of a splinternet. One Web is one part philosophy and one part development strategy.

The philosophy of One Web is to reuse as much of your existing technology investments as is possible for current and future web touch points, with the objective of achieving simpler and cheaper ownership of your organization's web assets over the long term.



Responsive Web Design

THE HUNGER for a one Web solution led to widespread enthusiasm for Responsive Web Design when it was first introduced. For the first time, developers suddenly had an alternative to web services for solving device proliferation – an alternative that promised to adapt to all sorts of devices with a single code base.

When people are asked, "What is Responsive Web Design?" they often respond with some variation of, "Responsive Web Design is the use of fluid grids, flexible images, and media queries to create a website that changes its layout based on the device's viewport." But this definition is missing two tremendously important distinctions.

Responsive Web Design is first and foremost a philosophy, and the development strategy that is so often quoted in isolation, such as above, was created as a way to apply that philosophy. Focusing on just the development strategy of Responsive Web Design is missing the forest for the trees. The bigger picture, and the whole context of Responsive Web Design, is about the movement towards a website built for the post-PC world.

Responsive Web Design is also not merely a technique to adapt a layout to different device viewports – it allows you to change a website's layout based on an entire range of viewport sizes. In other words, Responsive Web Design can help you build a website optimized for both an iPhone and a desktop computer, but also every device of every resolution that falls in between, and even a desktop browser that's being resized from small to extremely large. A responsive website is a single front-end that leverages flexible grids, flexible images, and media queries, maintaining all the logic behind multiple device layouts

> "RESPONSIVE DESIGN IS ONE PART PHILOSOPHY, AND ONE PART DEVELOPMENT STRATEGY."
>
> **ETHAN MARCOTTE**
> AUTHOR OF RESPONSIVE WEB DESIGN

in one place. Features built on the site appear on multiple touch points at once. Most useful of all, a responsively designed website automatically adapts to a whole range of viewports, and is ready for new devices with new viewport sizes that haven't even been shipped yet.

How Responsive Web Design **Falls Short**

Unfortunately, the single front-end of a responsively designed website has limitations and introduces some major challenges.

The first thing that development teams discovered was that Responsive Web Design was so fundamentally different from traditional web development – the CSS and HTML had to be completely overhauled – that making an existing website responsive meant having to rewrite the web application layer that serves all of the HTML content and all of the CSS, and as a result, also had broad impact on JavaScript and other front-end features. Essentially, the entire front-end of a website had to be torn down and rebuilt from scratch.

> STUDIES HAVE SHOWN THAT CREATING A RESPONSIVELY DESIGNED PAGE CAN TAKE UP TO **TWICE AS LONG.**
>
> **HIROKI TAKEUCHI** / CO-FOUNDER OF CARDLESS / "DITCHING RESPONSIVE DESIGN"

Studies have shown that creating a responsively designed web page can take up to twice as long as normal. Responsive websites may have a single code base, but a responsive site still has to account for three or more different screen sizes (desktop, tablet, mobile). In comparison to developing a feature designed for just one screen size, this can be very time consuming.

Not only is this a costly and time-consuming process, but it introduces additional problems. Existing Content Management Systems (CMSs) and eCommerce platforms may not natively support a responsively designed website, and developing a responsive website on top of such a system may be arduous at best or next to impossible at worst. In some scenarios, therefore, development teams found that taking their website responsive didn't just entail a rewrite of their front-end – it had far-reaching implications on the architecture of their entire web infrastructure.

Finally, on top of difficulties with architecture and development, a responsive website can suffer from many user experience drawbacks.

The single set of features means the same content and information is served on all devices, requiring mobile users to download as much as desktop users. This often results in over-downloading. Studies have found that up to 86% of all responsively designed websites send the same amount of content to mobile users as desktop users, which translates to low-performing mobile websites with increased load times.

Developing a responsive website with a mobile first philosophy can also lead to "dumbed-down" sites with

reduced feature sets. In such a case, the mobile site may not be heavy, since it was designed for mobile first, but the tablet and desktop sites may end up lacking experience differentiators or more robust features.

for immediate pickup, or looking up product reviews and comparisons so that she can make a decision about a purchase on the spot, or she might even be showrooming. All of these different behaviors depends on your specific business, but ignoring these very real situations will degrade your

While a single code base may be more maintainable, it also means that you tend to have undifferentiated experiences across all touch points, and are unable to take advantage of differences in user behaviors. For a relatively simple site like a blog or an online magazine, or content-driven sites like newspapers, this isn't a huge problem. After all, users are visiting such sites to read and consume content, regardless of whether they're on desktop, tablet, or mobile.

However, as features and interactions get more complex, users start to change their behaviors in different contexts. An online shopper visiting a site on desktop is interested in browsing and perhaps even purchasing, while the same shopper visiting the mobile site may be more interested in locating a store where her saved item is in stock

competitive edge.

There are parts of a web page that can and should be responsively designed to work in any situation. At the same time, because different contexts imply different user behaviors and different use cases, there are also parts of a web page that need to be very different. The more complex your web application, the more difficult it becomes to optimize each touch point for user behaviors.

Responsive Web Design was first adopted with gusto because it held the promise of being the development strategy for applying a One Web philosophy. But there a constraints and limitations to it's approach, and in trying to solve an old problem, it introduced a whole new set of problems.

Server Side Solutions

SERVER SIDE COMPONENTS attempt to solve some of these problems.

The core principles behind **Responsive Design + Server Side Components (RESS)** are delivering pages with some responsive content which is adaptable to screen variation, as well as device specific content, which is pre-selected by the server. By catering the content that is delivered to the user, this solves the problem of over-downloading and allows for the creation of individual experiences.

However, RESS still requires a front-end rebuild. And with all its promise, there is no standard platform for implementing RESS. So anyone looking to take advantage of all of its benefits would need to build their own from scratch, which is prohibitively difficult and costly.

That's where Moovweb enters the picture.

WITH ALL ITS PROMISE, THERE IS **NO STANDARD PLATFORM** FOR IMPLEMENTING RESS.

Moovweb

At its heart, Moovweb is the realization of a One Web strategy, but without the inherent limitations of RWD. Using a single set of reference code, the cloud-based Moovweb platform transforms existing websites into compelling experiences across a variety of touchpoints. Through a patent-pending site virtualization and transformation technology, Moovweb empowers businesses to deliver powerful experiences while at the same time maximize code sharing.

The process for using Moovweb is quite simple, as illustrated here:

1

SITE VIRTUALIZATION

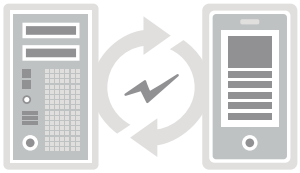
Virtualize your desktop site to create a mobile version. The mobile site instantly inherits 100% of the content, features and business logic, along with other APIs already built and tested on the desktop site. This usually takes just a couple of minutes.

2

CUSTOMIZE

Using the SDK, front-end developers create amazing post-PC web and app experiences in a fraction of the time of other approaches.

3



SYNC

Push the mobile site live in the cloud,which syncs everything across all your Web experiences – in real time. Use the cloud-based console to collaborate with other developers and manage projects, updates and analytics.

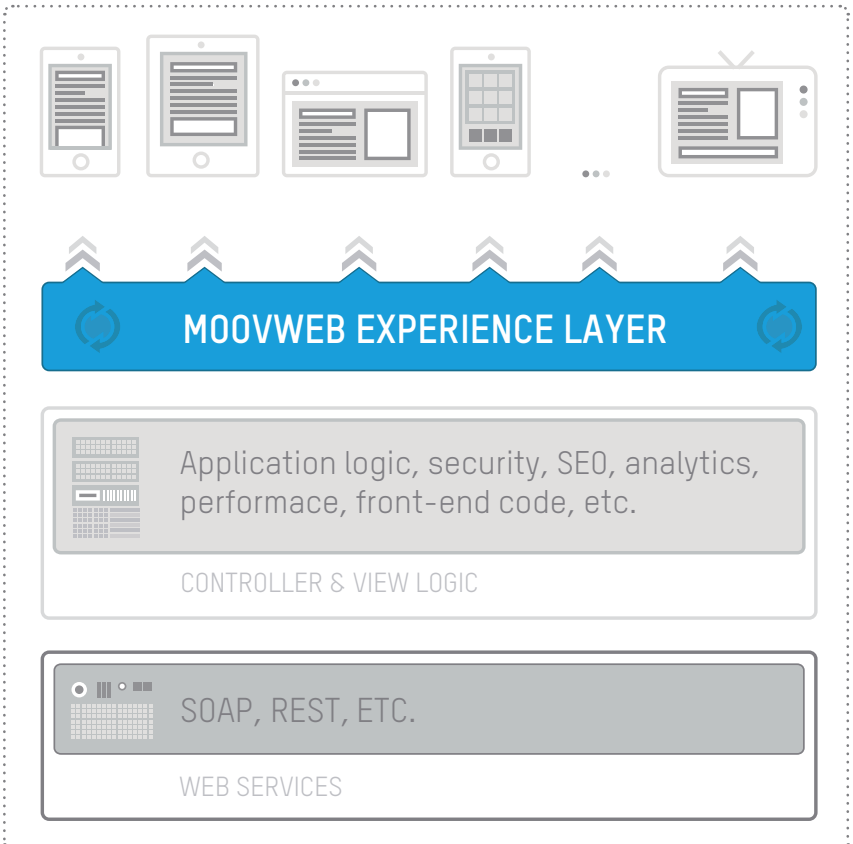
When any device (like a smartphone or tablet) requests a Web page through Moovweb, the page content is instantly retrieved from the desktop site, transformed and delivered with a customized look and feel.

Full stack site virtualization is the answer to stack fragmentation. With Moovweb, all features, content and business logic on the virtualized site are instantly inherited by any end point. Any changes made to the original site are automatically reflected on the virtualized site. In this way, a single feature like a new payment process or store locator can automatically appear on any number of touch points automatically.
Site virtualization not only helps with instant deployment of new features from an origin site to multiple end-points, but also acts as an additional layer of modularity in your Web stack. Because the virtualization depends solely on the front end of your website, you are free to change your underlying back

end implementations at will. Changes to your CMS, eCommerce engine, or the underlying integration for a specific feature are all invisible to the platform as long as the HTML on the front end does not change.

Real-time HTML transformation is the other pillar of the Moovweb process. It offers complete control and flexibility so that you can add, remove, or modify anything on the virtualized website.

By default, virtualized websites begin life as an exact copy of the original, down to the back-end integrations such as search and user authentication. Through



Moovweb's real-time transformation technology, these virtualized websites can be dramatically altered in look and feel to create a user experience tailored to any device. Any site can also be completely transformed in structure and layout so that, for example, a website that was once built for the desktop is now responsive and adapts to a whole range of screen sizes.
In addition to site virtualization and real-time transformation, here at Moovweb, we believe in leveraging the best of Responsive Web Design to truly craft a future-proof post-PC Web solution.

As we mentioned earlier, one of the glaring issues with Responsive Web Design is its stark difference from how websites have been built up till now, therefore requiring an expensive and time-consuming rewrite of an entire website's front end from the ground up.

With Moovweb you don't need to undertake such a costly endeavor. With the power of real-time transformation, you can transform the HTML of any website into the HTML for a responsively designed website.
With Moovweb, downstream agility is accelerated since front-end developers can now focus on delivering unique experiences for mobile, tablet and other touch points. Front-end developers are designing and delivering front-end code, not worrying about building custom integrations to existing back-end code. Using this approach, another big hurdle to RWD is solved. Instead of having to rewrite your entire front end, you can allocate your resources to building new front-end experiences and enhancing existing ones.

With Moovweb, it's easy to transform selected components of a website in any way that you desire. Entire sections of HTML, corresponding to specific components, can be modified

depending on the device type you want to optimize for. Meanwhile, responsively designed components that were made to scale from desktop to mobile are left unchanged, and automatically adapt to their device constraints.

In addition to modifying existing features, you can even insert completely new ones that didn't exist on the original website. For example, with a few simple commands, you can insert something as radical and interactive as a real-time concurrent "likes" system for your conference attendees. Not only do you get a highly optimized experience tailored to mobile conference attendees, but you also don't have to burden your tablet and desktop users from downloading a feature that they won't use.

Finally, Moovweb never overloads mobile devices with responsive pages designed for a desktop site. Only the data that matters is downloaded to the device, not all the stuff you don't need. When matched with smart caching, performance is stellar no matter what device is being used.

BENEFIT	RWD	RESS	RWD + MOOVWEB
Single Code Base	√	√	√
Single URL	√	√	√
Adapts to all screens	√	√	√
Easy to optimize individual experiences	x	√	√
Easy to optimize performance	x	√	√
Post-PC ready	x	x	√
Mitigates long term ownership complexity	x	x	√
No need to re-write front-end	x	x	√
Development platform available	x	x	√

Conclusion

TODAY, WE ARE only at the beginning of the full-fledged post-PC era. In a matter of years, companies will have far more than just desktop, tablet, and mobile to contend with and manage.

Responsive Web design is a great solution for many companies that want to maximize code sharing and deliver experiences that automatically tailor themselves to different screen real estate. This approach works best for simple, non-transactional publishing sites. For more complex sites, RWD has some serious drawbacks, some of which are solved with a RESS approach: moving more code to the server. But RESS introduces it's own unique set of challenges.

WHEN COMBINED, RESPONSIVE WEB DESIGN AND MOOVWEB FORM **A REAL POST-PC SOLUTION**

Moovweb solves these issues with a next generation approach to mobile that not only maximizes code sharing (One Web), but makes it easy to deliver custom mobile experiences for multiple touch points. The platform has been doing this for some of the biggest brands on the planet, with billions of mobile page views transformed in just the last two years. Moovweb combines the two groundbreaking technologies of full stack site virtualization and real-time transformation to solve stack fragmentation once and for all and realize a true One Web solution for the post-PC world.