THE JavaScript WEB DEVELOPMENT Revolution

AN EBOOK
BY SCRIVITO

The JavaScript Web Development Revolution

How Technologies Like ReactJS and Serverless Computing Will Change the World

An eBook by

scrivito

Written by: Thomas Witt

Edited by: Zack Young

Contents

1 Introduction & Background

7 The evolution of Web Development

18 Shorten your development process with ReactJS

25 The evolution of serverless computing

29 How web experiences for end users have been impacted by these changes

32 Summary of topics which have been covered

33 How to implement these ideas in your organization

37 Further reading on Scrivito and how to get in touch

Introduction

2018 is a watershed moment for web development. Legacy content management systems (CMS) like Wordpress et al have failed to keep up with the demands of modern web development. As more organizations and digital agencies embrace an agile approach to marketing in order to deliver dynamic, ever-changing customer experiences, the technologies we choose to drive those web and app experiences can create or remove huge obstacles in the development cycle.

Enter the dream team of serverless computing and ReactJS. Considered the next generation of cloud computing, serverless is making it possible to build powerful apps and websites without a server, without the maintenance headache, and without the need for constant security patching. Combine that with ReactJS, the innovative JavaScript library that's backed by Facebook, and now you're able to quickly deliver secure websites and apps with dynamic front-ends, highly interactive content and sleek UIs.

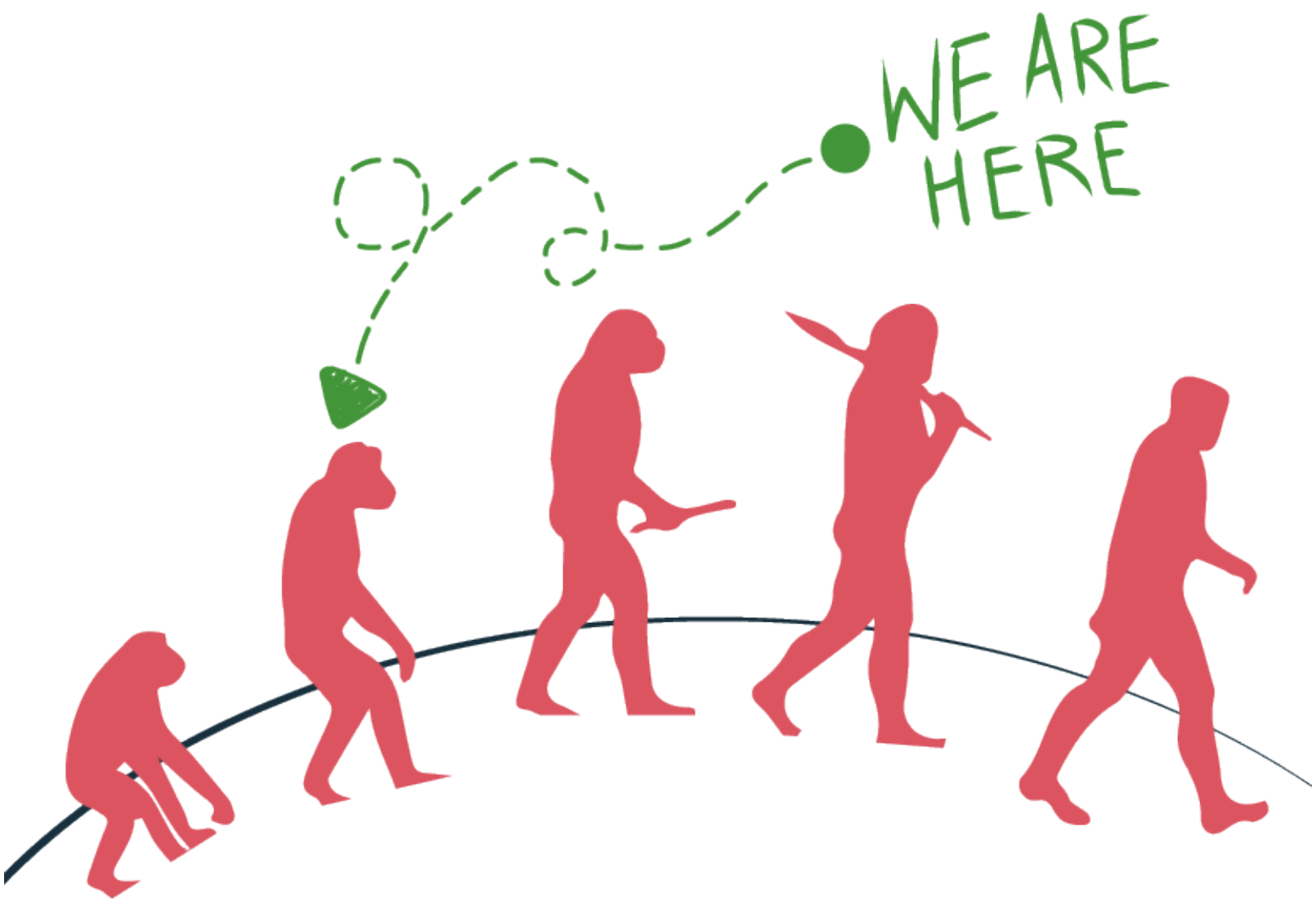
Welcome to the web development revolution. Anchored in massive technology leaps that are driven by client-side rendering and serverless computing, it's a new way of development that drives better performance, maximum security, lower cost of scaling and a better developer experience.

Background

“

The Web as I envisaged it, we have not seen it yet.
The future is still so much bigger than the past.

- Tim Berners-Lee



Technology, and more specifically, web technology is still in its infancy. It’s still learning, growing and maturing. Developers today can have a huge impact in creating the web of tomorrow. When Tim

Berners-Lee first created the Internet, he was focused on the impact to humanity where a tool was enabling a social connection. His focus was rarely on the details, but instead the impact they created for his vision. It's surprising today, that part of the Internet is subject to censorship and Net Neutrality is on the political agenda. It seems the original vision must be defended, or the Internet will be open to control for pure commercial or political gain.

Whether you're building web applications in your organization or you're focused on how technology impacts your business objectives, we've produced this eBook to help you navigate the newest technologies and latest solutions and help inform your development decisions throughout 2018.

Is your organization harnessing the power of the latest web technologies?

Legacy technologies such as PHP, Java and even Visual Basic are still used by many companies, but they are often costly to maintain, have endless limitations and could be responsible for the painful days when your developers start their next sprint - right out of your company. A legacy system is not necessarily defined by age. Legacy may refer to lack of vendor or community support, making it vulnerable, insecure and incompatible with future required integrations. Fresh talent in the market may not be skilled or interested in maintaining your legacy applications, leaving you to source an older generation of developers who prove more challenging or even impossible to find.

The performance and success metrics of a web application are no longer simply measured by the volume of consumer interactions. The core metrics we use today are centered around user experience, speed, and performance across a growing number of devices.
The output is, of course, unchanged, but the journey to achieve this output is more important than ever before.

And to meet the modern requirements of web development – requirements around speed, security, and interactivity – legacy CMS tools won't get you there. Like the adage of 'fast, good cheap – you can only pick two,' with legacy CMS, you were always having to make sacrifices. Thanks to the momentum of new current JavaScript language and serverless technologies, you no longer have to choose between speed, security, and experience – it's possible to meet the demands of all of today's success metrics.

Aside from the impact to the consumer or end user, it's essential to factor in the internal resources required to develop and securely maintain core web applications. As project leaders, we must consider the true cost of using legacy technologies and the internal pain this can generate. Now with more organizations adopting an agile approach to development, our applications are ever changing and fluid experiences. The technologies we choose to drive these applications can create or remove huge obstacles in our development cycle, which often directly impacts business performance.

The internet is no longer just another marketing channel, it's within our human fiber. A new world of IoT devices are now embedded into our

lives, we've become dependent on this to drive our decisions. It's also become integral across all of our business functions. Building for the web is part science and part art. Now we must consider a user's experience, whilst delivering the function which the application was built for in the most efficient manner.

We're Infopark AG.

Our company was founded in 1994 when the Internet was considered the next big thing in technology. This was a time before browsers were developed, design thinking did not exist and user experience was not a job type. We could see the potential and the curiosity in the market. We could see a world where internet technology was ingrained in our lives - much like the world we live in today.

This led us to form Infopark, with a focus on becoming the partner for companies through digital transformation. Over the last 20+ years, the internet has evolved many times over, and with our expertise, we've guided companies through the many changes the web brings every day.

We've worked with thousands of companies on a wide range of technology projects, and opened offices in Berlin, Wroclaw and most recently Boston. Over the last two decades, we've built a team of 70+ specialists and have started developing our own web products to serve the needs of the companies we work with. With each new cutting-edge technology the market has delivered, we've been inspired to explore new approaches and applications for use. Adapting to the changing world of technology is in our DNA, and why our clients trust us to advise them on their technical projects.

Our vision for Scrivito is to become the number one serverless CMS in the world - enabling the next generation of web development for digital agencies and medium to large sizes businesses. It is at the forefront of the current JavaScript web development revolution, enabling companies to get high-quality editable outputs with less effort. You can learn more about Scrivito here: www.scrivito.com

Or, if you'd like to say hello - reach out by email:
therevolution@scrivito.com

The evolution of development and how this has impacted developers, end users & the businesses who are creating digital products and experiences

> *Never before in history has innovation offered the promise of so much to so many in so short a time.*
>
> ***– Bill Gates***

In its infancy, Internet technology was seen as simply a new method of communication. But, it's become so much more than that today, finding itself engrained in the very fiber of our lives. It's virtually impossible to live in any modern society without depending on Internet technology in some way. Every year, we see more legacy systems utilize the power of the web to become more efficient, deliver a better user experience and power the growing number of demands on services.

To create some context and to better understand how web technologies have developed so far, let's explore a brief history of the web. The idea of the 'web' was first formed in 1989 by Tim Berners-Lee and born to the world in 1991. Since its birth, great minds saw its potential and

have built many tools enabling us all to share, create and contribute.

The web was, in the early years, seen as a world separate from our own, as if it was a country, which you traveled to. This was due to its new nature, non-consistent interaction and the lack of hardware in homes and workplaces (and now pockets) enabling access. It's come a long way since the initial static pages into a platform we all share.

KEY POINTS OF WEB EVOLUTION

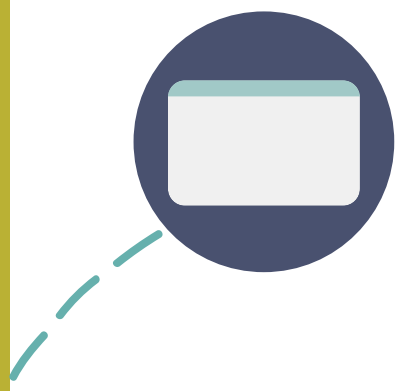
Here we'll cover some of the highlights from the early web into the web we know today.

Web 1.0

The first version of the web was labeled 'Web 1.0'. Upon launching into the world as a place to communicate in a standardized way, it was a period which lay the foundations of many of the technologies, or the early versions of them, which led to the experiences which power our modern version of the web today.

One of the missing components of the early web, or Web 1.0, was a lack of interaction and design-centered development. It was about creating a place where information could be accessed globally and not designed

to factor in the experience of the user. It was far from the web as we know it today. Websites and applications today have a huge amount of input from a variety of disciplines, many of which didn't even exist until recently. We've come a long way from black and white pages displaying data, into something that is whatever you want or make it be.



Browsers Develop

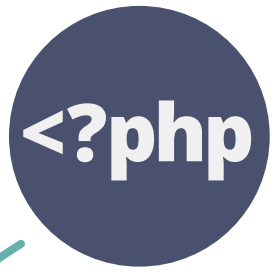
Beyond the existing 'web browser', which was a series of hyperlinked applications first developed by Tim Berners-Lee, we saw the real development of web browsers in 1993. Marc Andreessen and his team launched Mosaic, which supported multiple Internet protocols and was one of the first graphical web browsers. It enabled users to have a standard interface when accessing data.

Inspired by the success of Mosaic, in 1994 Netscape released their first web browser 'The Netscape Navigator' which was adopted widely. It was the first browser to show the content as it was downloaded, a unique feature as other browsers only displayed the content when it was fully downloaded. At the time, in an age of dial-up Internet, it was a unique feature.

Netscape's web browser grew successful in the early days with the help of ISPs and computer magazines, which featured images of the early web housed within its browser.

JavaScript Created

In 1995, JavaScript was born. The original version was called LiveScript and shipped with Netscape's web browser. It was quickly popular for a variety of reasons including the ease of learning since it shared a similar syntax with English; the fact that it was a client-side language so it saved the developer server resources; and that it wasn't dependent on any specific platform.



Arrival of PHP

Also in 1995, Personal Homepage Tools (known as PHP) was developed by Rasmus Lerdorf. This had a host of new functionalities enabling developers to create dynamic web pages and turned the web into a three tier architecture: client (browser), processing (web server) and data storage (DB).

Internet Explorer 3.0 launches with Windows 95

In 1996 IE3.0 arrives bundled with Windows 95. This browser enabled many new capacities and a host of new powerful functions to be accessed via the browser. What also made browsing possible was the state of Windows 95 - which had more powerful functions for the average user. Consumers began to realize that computers could be used to find a wealth of information and

communicate with long-lost friends, and for entertainment. This meant owning a computer became more desirable and sales boomed as Microsoft promoted the personal computing experience.

Google

In 1997, Google arrives on the scene - enabling users to find data faster than ever. The birth of Google brought a host of additional thoughts for website owners. This included search metadata - which meant website owners needed to factor in website structure, readability and more for search bots.

Blog Comments

In 1998, Open Diary brought a new innovation to blogging by enabling readers to comment on the author's blog entry. Giving consumers of data the ability to now comment and interact changed the dynamic of blogging. It was no longer a one-way delivery, but an open dialogue. Feedback left bloggers vulnerable to criticism but equally helped develop a rapid feedback loop - which has helped create many of the product development methodologies today.

Blogging without the tools we use today such as WordPress, Blogger and Medium were almost always static web pages, which had to be updated manually. Break-

throughs in 1998 shaped what blogging has now become - often referred to as a community, where news is shared and a place for multiple parties to communicate.

Wikipedia

Wikipedia, an online encyclopedia, arrived in 2001. Now a vital resource for all, the launch of this company was a key step in shaping the idea that no central organization needs to control or edit data heavy resources. The vision for Wikipedia was to enable anyone to contribute and edit.

Web 2.0

“What defines Web 2.0 is the fact that the material on it is generated by the users (consumers) rather than the producers of the system. Thus, those who operate on Web 2.0 can be called prosumers because they simultaneously produce what they consume such as the interaction on Facebook and the entries on Wikipedia.” - George Ritzer, American Sociologist

In December 2003 Web 2.0 was born, opening up new means of communication and collaboration. Web 1.0 brought us email, Web 2.0 brought us social media and many ways to communicate and collaborate. Many ap-

plications and websites were more user-driven and relied on user-generated content and engagement to deliver widespread value.

Facebook

Although social media was already around with platforms like Myspace and Bebo, Facebook arrived on the scene in 2004 and became widely available from 2006. Facebook standardized the social media experience and welcomed developers with a range of platform tools.

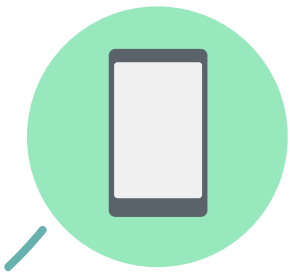
JavaScript Frameworks

In 2005, the first frameworks arrived. Frameworks enabled improved code quality, maintainability and added a layer of structure when building applications. The first frameworks were ‘Prototype.js in 2005 and ‘jQuery’ in 2006.

Amazon Web Services (also known as AWS)

Although similar cloud tools were available, the real birth of cloud computing was in 2006 when Amazon Web Services enabled true virtual cloud computing. They brought developers EC2, enabling highly flexible

computing resources and S3, a robust cloud storage solution.



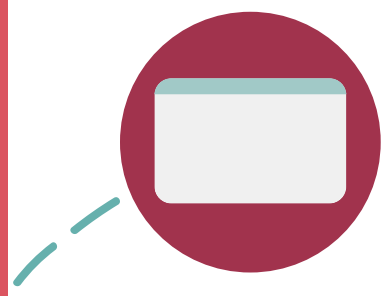
Apple iPhone

In 2007 a new wave of technology arrived when Apple launched the first iPhone. It was a breakthrough device, which brought real power to mobile computing, whilst also bringing the full web experience to mobile - leaving the previously basic web mobile sites behind. At a similar time, 3G started rolling out across mobile networks making more powerful mobile experiences possible.

> *An iPod, a phone, an Internet mobile communicator... these are NOT three separate devices! And we are calling it iPhone! Today Apple is going to reinvent the phone. And here it is.*
>
> ***– Steve Jobs, Apple***

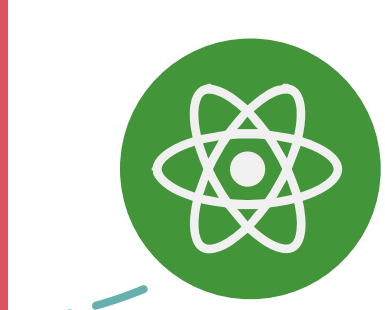
Before the iPhone and 3G speeds, the mobile web was a slow and highly limited resource. Website owners would usually be selective about what their mobile website would display, leaving out the majority of the informa-

tion. In many cases, companies didn't even have a mobile version, as smartphones were yet to really take off.



Google Chrome Browser

In 2008 we saw Google Chrome arrive, a modern browser which promoted itself on speed with a variety of Google-powered functions. The bundled features of the Chrome browser were great for both web users and developers. Plus, they opened the browser up to third-party plugins, so users could customize their web experience.



JavaScript Evolves

Since 2011, every year a new JavaScript framework has arrived. Many of these frameworks addressed the challenges, which were present, made reusing code a breeze - speeding up development time and enabling a standardized structure. This was especially useful for multiple developers working on the same application, or ongoing maintainability of the application. Some of the main frameworks which arrived were: ExtJS, Ember, Knockout, Backbone, Angular, and ReactJS. Nearly every year, a new framework came to light; the speed of innovation within the JavaScript community is still unprecedented. Today ReactJS has emerged as the clear

winner of this year-long competition and is here to stay - last but not least because of the backing of Facebook.

AWS introduced 'Lambda' - enabling JS code to run without server requirement

In 2014 Amazon Web Services released Lambda, a powerful cloud platform enabling developers to run code without the need for managing services. The platform enables rapid access to resources however large or small.

SUMMARY

This brief history shows us just a short snapshot but highlights some of the key progressions in the world of technology. These progressions come in various forms, from the physical developments in hardware to fresh ideas, which have opened up many new opportunities and the web technologies which enable these powerful digital experiences. Moore's Law suggests exponential growth will continue in the technology sector, increasing speed and performance, whilst driving down cost.

JavaScript is the fastest moving sector online right now. Developers have created at least six major JS frameworks over the last four years and users have started experiencing upgraded browsing experiences

for many of the applications they use due to the fast adoption rate of these frameworks. Also, thanks to the innovation speed in browser technology, the user's software gets automatically upgraded all the time to provide better experiences - there were about ten releases of Chrome in 2017.

Much like in the physical world, history has helped us become smarter, more efficient and more user-friendly with design and development. This should also be mirrored online, to ensure the experience of using the web continues to improve over time and evolves as our needs do. Our expectations of technology are now higher than ever, pushing the developers behind the devices and the platforms we use today to create experiences which are seamless, interconnected, secure and without boundaries. We must avoid legacy technologies and adopt and embrace the future technologies, which will power our world.

How and why JavaScript and ReactJS have changed the process of development

and what this means to your business

That leads us to today, where there is a revolution happening with web technologies. It all starts with JavaScript, the language which has been around since 1995 and is considered the glue in any web build. In the early days, everything had to be rendered server-side, simply because the browsers and devices were not powerful enough, especially on mobile phones. Dynamization could be done partially at best, but not fully.

An important factor was SEO: Search engine traffic continues to dominate the game of Digital Marketing; if played right it's the most affordable way to acquire inbound traffic. With traditional server-rendered pages, the process for setup and optimization was standardized for many years. When ReactJS (often just called React) and other technologies arrived, search engines couldn't see the content as the content was rendered locally. The search bots relied on fully loaded HTML to see the page content to generate data which informed the ranking. From 2015, things changed when the devices and browsers used by the majority of internet users were powerful enough to render a page completely client-side, without the need for a server. This was done with some HTML and JS files, plus data delivered through APIs to the web application. At the same time, Google announced they could now crawl and render JavaScript and CSS files, which was not achievable before:

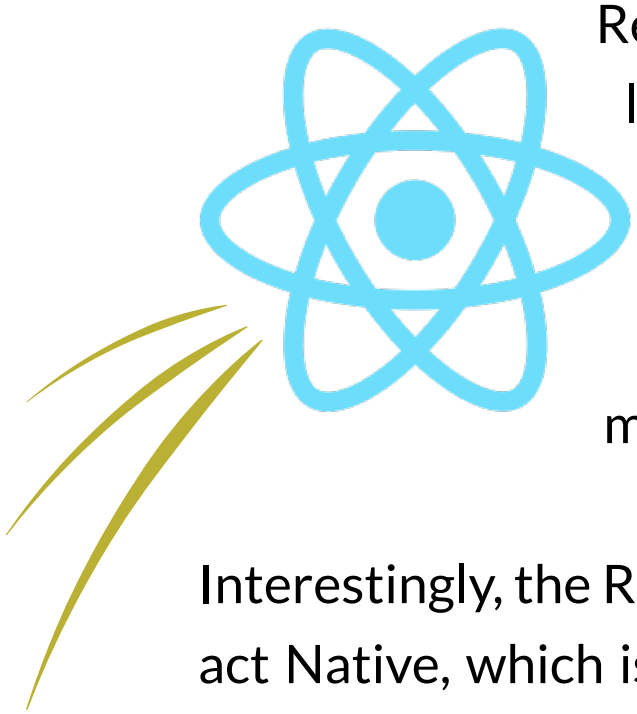
> *As long as you're not blocking Googlebot from crawling your JavaScript or CSS files, we are generally able to render and understand your web pages like modern browsers. To reflect this improvement, we recently updated our technical Webmaster Guidelines to recommend against disallowing Googlebot from crawling your site's CSS or JS files.*
>
> ***- Google***

Since then, pages built with these new technologies would not be penalized and could rank just like server-rendered pages, without the need for any specialist tools. With this SEO barrier torn down, people started creating client-side rendered pages. They're faster, more dynamic and more secure whilst also having much less moving parts.

Also in 2015, ReactJS was created by Jordan Walke, a software engineer at Facebook. It was designed to enable speed, simplicity, and scalability. React was needed for their own site 'facebook.com' as it became very challenging to handle. React, an isomorphic JavaScript library enabled developers to render applications client-side, and shift away from server-side rendering.

The biggest disadvantage of server-side rendering was the need for a complete page refresh anytime the user made a minor change to the user interface. React brought developers a virtual DOM, which

allowed developers to update the contents in a web application in a far more efficient way while ensuring great performance, which beats server-side rendering hands down speed-wise. Every time the DOM changed, browsers needed to recalculate the CSS, redo the layout and re-render the webpage - which takes a lot of time. By abstracting, reducing and batching these changes to the DOM completely new possibilities arise for the web developer by reducing the complexity at the same time. It is a simpler way to render and change things in the browser, which makes optimizing the user experience easier and results in a huge performance and productivity boost.



React is simply a JavaScript library. There are very limited amounts of API to learn, just some React functions and how to use them. A JavaScript developer can become a React developer in only a few hours, removing the steep learning barrier many newer programming languages have.

Interestingly, the React universe also includes a technology called React Native, which is a mobile framework enabling developers to create cross-platform native applications - by using the same code for the web application. This is a huge advantage, enabling the recycling of much of the codebase developed already for web into the mobile application development. Naturally, the greatest benefits here are the speed of development, cost savings and the ability to recycle most of the code when maintaining your applications.

React is all about creating rich websites and applications, which are anything the client sees in their browser. Traditionally, any changes to a web page would mean a reload was required. Now, React removes this barrier and enables real-time loading of UI changes. With any change, however large or small, the user makes in the browser, the UI is updated instantly, even if the same information is reflected multiple times on the same page.

It's super fast since logic is rendered locally in the user's browser, minimizing the amount of data being transferred from the web server. This is also often combined with CDN (content delivery network) technology, to bring the JavaScript files and associated media content as fast as possible to the client using the latest advanced technologies such as compression, minification, and chunking.

React also has integrated performance-monitoring tools to analyze which parts of the application are incorrectly subjecting the application to re-rendering. This makes life easier for the developer.

React also introduces a kind of a new paradigm to the web here, as it breaks down the functionality of a site into so-called components. Components are independent, encapsulated and reusable pieces you can use within a website. In the old days of the web, people tended to think more in web pages in terms of a classical page. You can see that pretty much at typical legacy CMS systems, which often ask you when creating a page to choose a template: Would you like to create a press release or a blog post? And that choice pretty much defines the functionality on the whole page.

Not so with React: You still create a page but you can choose from your library of components, distribute them all over the page and even re-use them between projects. Built this cool CRM integration for one project? ***Let's reuse that in the next project. This saves time and money*** for larger businesses managing multiple sites in multiple countries for multiple products.

This is especially important for digital agencies while dealing with multiple clients. In an ideal ReactJS world with a CMS, which supports that paradigm, they continue to build their library of components which they can reuse between different client projects, leveraging their degree of prefabrication and thus lowering their costs and efforts to create client projects.

Last but not least, moving from server-side to client-side, especially combined with serverless computing, is way more secure than traditional server-based approaches because simply there is no server and therefore no attack vector. You don‘t need to patch anything because the code runs in the browser and not on your machines.

Amongst developers, this way of building apps is increasingly referred to as JAMstack applications: we're not talking about specific operating systems and web servers anymore, but rather about creating a new development architecture, a new way of building websites and apps that delivers better performance, higher security, lower cost of scaling, and a better developer experience. And this way is mainly based on client-side JavaScript.

We here at Scrivito call it the Web Development Revolution (www.webdevelopmentrevolution.com).

The barrier between native mobile apps, native desktop apps and web pages is blurring. Clients are becoming faster and more robust, leading to dynamic front-ends, interactive content, and sleek desktop-like user experiences.

JavaScript is also the perfect language for the cloud (otherwise known as serverless computing), as it's built for compartmentalizing software, which has no connection to other software on the same machine. You can easily run multiple JavaScript applications on the same machine, without interfering with each other. As this sandboxing is also very important in browsers, in order that one loaded page does not interfere with another, JavaScript engines are highly secure and built for the cloud - on the browser as well as in serverless computing.

The progression into serverless computing, its benefits and how a traditional barrier to web development has been lifted

> *Every kid coming out of Harvard, every kid coming out of school now thinks he can be the next Mark Zuckerberg, and with these new technologies like cloud computing, he actually has a shot.*
>
> ***- Marc Andreessen, Partner at Andreessen Horowitz, Co-founder of Netscape and Board Member of Facebook***

Serverless computing is more often referred to as the next generation of 'cloud computing', and is a breakthrough in the way we handle the backend of our applications. It also supports cost efficiency, scalability, security and the reliability of our applications. Much like the journey software has taken to get to this point, server technology has been on a similar journey.

It all started with classical server hardware in racks, with installed databases and web servers. This classical structure was high maintenance, not very fault tolerant and the utilization was low. It was also expensive to setup and manage, making it a barrier for new application development.

The breakthrough server technology needed occurred in 2000, when VMware arrived. Virtualization technologies enabled the installation of multiple web server instances on one machine, so applications could run on virtual private servers (VPS) and have independent core

resources, while not being impacted by other applications on the physical hardware. This improved hardware utilization made installation smoother and reduced the cost of launching and hosting new applications.

In 2006 Amazon Web Services launched computing on demand, enabling the rental of computing power and servers by the hour. This opened up a new dimension for applications to handle load peaks and solved many of the resource challenges, which were presented before. It also enabled fast growing applications to scale, without the same complications.

Docker also brought us containerization technology in 2013, allowing several components to run in their own environments on the same server. Even still, these developments were simply the evolution of technology, and far from a revolution. You still had servers to manage, whether they were physical or virtual, you had to choose configurations and there was still a barrier to getting an application set up.

But, in the end, nobody wants a server. That‘s just the necessary tool to run the important thing: your application. In the end, everybody wants an application, not a server.

The real revolution in serverless computing was when AWS introduced Lambda, which enabled anyone to execute code without the need for a server. The whole environment was powered by AWS; you could simply upload code and Lambda would run it for you. It opened up a host of new opportunities and broke away a huge barrier when

creating new web applications. The solution took care of the pains associated with managing a server; from cost over scaling to security, it was a complete solution. For rapidly growing applications, you could rely on AWS' resources to ensure availability.

This was the first time a developer could create a web page or application with React, whilst directly communicating to a backend without the requirement for a server. JavaScript could be the language, which ran the backend and the frontend. It was the real breakthrough revolution for serverless computing, now anyone could build powerful applications without the need for a server and the maintenance efforts that come with it.

Technology startups today no longer have to be concerned about the technical infrastructure, which was a huge barrier to getting started. The cost and expertise required to design and manage the infrastructure needed for growing technology companies was often a reason for failure, or perhaps a reason one would never hit the market.

How web experiences for end users have been impacted by these changes

With these advancements in technology, how has the web experience for the end user been impacted?

Websites and web applications now have teams behind them, often very large teams comprised of: designers, developers, UI/UX specialists, eCommerce specialists, SEO/SEM specialists, project managers, content producers and many more. A whole industry has been created, called 'User Experience' or UX. It's an industry dedicated to creating user-friendly design and experiences.

With the advancement of mobile technology and the infrastructure to keep us connected, we now have access to data faster than ever before. This has changed and continues to change the way we do just about everything. It's changed not only our behavior but our expectations. These expectations are now very high - we expect our web experience to be super fast, dynamic and highly interactive. These higher demands continue to push developers to deliver what we want in new and interesting ways.

“

Whatever the device you use for getting your information out, it should be the same information.

- Tim Berners-Lee

Today the web experience for users is standardized through a set of universal principles like HTML5, CSS3, and JavaScript adopted by teams worldwide. The World Wide Web Consortium (W3C) is one of the main organizations driving these standards, with a mission to lead the web to its full potential. Universal standards are vital since they help ensure much like in the real world, that various functions work as we expect them to. These improvements have been pivotal in building more trust in the web. We have become dependent on an internet-powered world, for everything from communicating with friends and family, doing our taxes, booking our flights, navigating foreign streets and more. Without this connectivity - we're detached, lost and in some ways helpless. Therefore, it's become ever more important to create robust applications which are always available, work as expected and across whichever device is being used.

Thankfully, as technology forges ahead and scripting languages evolve, we are now at a point where the advances we have covered in this ebook will enable forward thinking developers - and the agencies that support them - to make a tremendous impact on how we all experience the web. And thanks to serverless computing, it will be more secure, more stable and more scalable.

Summary of topics that have been covered

We started this eBook covering some of the key highlights from the history of web technology. Each of the highlights we discussed earlier has played an important role in creating the web environment we use today. From the progression of physical hardware to the pursuit of big ideas, they have all impacted our Internet-powered world.

We then explored JavaScript and React, two technologies which are powering the web development revolution. These technologies are paving the way for serverless development and removing traditional barriers to launch and painful obstacles to maintain. It's clear that with greater power and mobile technology, a serverless world is near.

An actionable list how to implement these ideas in your organization

Knowledge is not power.
Knowledge is only potential power. Action is power.

- Tony Robbins

Now that we've covered some of the key technologies, which are driving the web development revolution, how will you take advantage and start applying them in your organization? To help you on your journey, use our actionable list below to establish your current position and highlight what needs to change.

01

We recommend you complete a Strategic Technology Audit. This will involve revisiting the initial project scope, which drove the development of your application(s). Does the same scope, and sets of project requirements apply today? Ask yourself why the application(s) exist and what problems do they solve? Are any changes needed, or perhaps some of your applications no longer serve their purpose? Just because

a technology fulfilled its purpose some years ago doesn't mean it's still the right tool for the job.

02

Beyond scope, vision and purpose - what technologies are your applications built in? Did you follow JAMstack ideology and choose this stack of technologies with intention? You may find through this audit that your application is serving a completely new purpose. The technology used may not have been chosen with intention; in addition, the project scope may have changed dramatically, along with the end user expectations and experiences (knowledge, browser technology, different devices). Also, keep in mind that users nowadays consider and expect different possibilities from web applications than a couple of years ago.

03

Is talent readily available when you need it? Are the technologies you use to build or manage your application still considered important in the market? If you answered 'No' then perhaps it's time to consider reviewing the technologies which drive your application. It's essential in this environment to have access to a skilled talent pool, who are equipped to manage and develop your applications. If the skills required to maintain your applications are becoming dormant, there is a high risk that you'll face hiring or retention challenges.

04

How secure are your applications? If you've had multiple developers contribute code to your applications or are depending on legacy platforms, do you know how secure your applications are?

Many organizations continue to keep legacy systems alive, as they fear the unexpected costs or don't want to dedicate resources for a rebuild. This poses a huge risk to both the security and availability of the application, whilst also potentially causing hiring challenges. With JavaScript and React dominating the revolution of web technologies and serverless computing powering the next generation of development, it's time to review what and who is going to power your websites and applications moving forward.

05

Have you calculated the cost to maintain your legacy solution, versus leveraging the latest technologies? Factor in the direct, opportunity and HR costs associated with holding on to old systems, and the people who need to maintain them. Even though some of your software components might be open source and therefore "free", they're far from free when it comes to operating, maintaining and patching the whole system. Even if you rely on components like Apache, MySQL, Wordpress, and others, they need to be maintained in a classical server-based approach. For these technologies combined there were hundreds of thousands of vulnerabilities in 2017 - which means you need to patch your server every week or sometimes every day in order to stay safe and secure.

In response to the many technology challenges organizations face today, we decided to build Scrivito as the next generation Content Management Platform for dynamic, interactive websites and applications. Leveraging the full power of the Internet today and being the first cloud-CMS, we have created a service that is highly secure, flexible and cost-effective. It evolves with the needs of the application and we believe with all of our expertise, it will remove the many pains and risks involved with website and application development.

Most organizations prefer to get started themselves, but a full project management team is available for more advanced use cases. You can learn more at www.scrivito.com.

Your websites just run better with us.

Our recommendation in one sentence? Go the JAMstack way and join the JavaScript Web Development Revolution!

Further reading on Infopark and how to get in touch

We hope you've enjoyed reading this eBook and have found value from the contents. We recommend you register for our quarterly newsletter covering the latest developments worth taking note of. You can register at www.scrivito.com/eBooks.

If you'd like to learn more about Infopark, Scrivito or the technologies we've covered you can use the links below for further reading.

You may also get in touch if you have any questions:

therevolution@scrivito.com

- About Infopark:
  www.infopark.com
- About Scrivito, the world's first serverless CMS:
  www.scrivito.com
- JavaScript Resources:
  https://www.javascript.com/resources
- React, a JavaScript library for building user interfaces:
  https://reactjs.org
- AWS Lambda, run code without thinking about servers:
  https://aws.amazon.com/lambda/

Created by Infopark AG
Kitzingstraße 15
12277 Berlin

First Release Date: January 2018
Last Updated: January 30th, 2018

Written by: Thomas Witt
Edited by: Zack Young
Designed and illustrated by: Jasmina Kloss

Disclaimer