

# Wild World Web

Web Development in a World of Ever-Changing  
Browsers, Platforms & Compatibilities

# The Web a Dozen Years Ago



Microsoft®  
**Internet  
Explorer® 6**

# Just Click the Big Blue E

- Windows XP + Internet Explorer (5,5.5 and 6) was 95% of the web
- Just 2 screen resolutions mattered



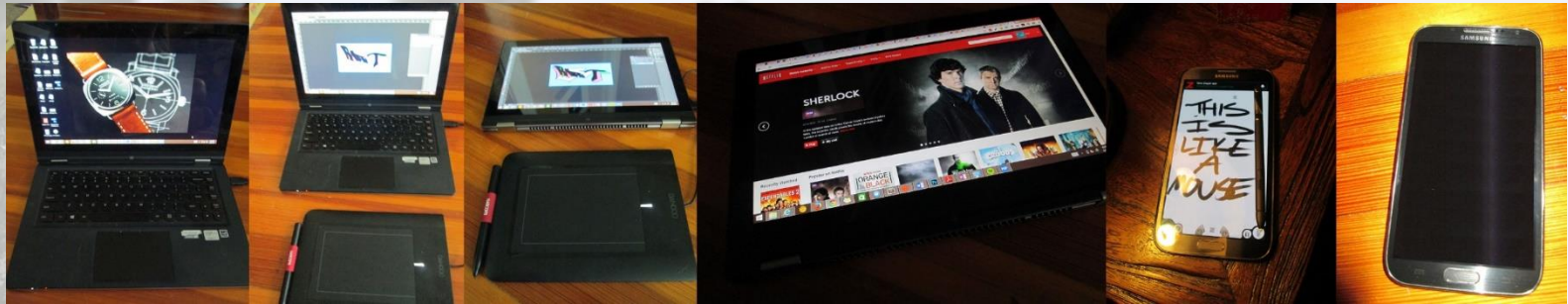
# Things Were Pretty Stale

- We had *rules* and browser specific fixes
  - Think: 960 pixel grids, the Netscape Navigator resize fix and IE conditional comments
- Calcified specifications
  - Between December 1997 and September 2001 we had: HTML4.0, XML 1.0, CSS level 2, ECMAScript version 3.0, XHTML 1.0, and SVG 1.0
  - After that... not much for many years on the specification front

# In the Mid-2000s Things Started to Change

- The WHATWG was formed
- Ajax-based development + the explosion of JavaScript Libraries (esp. jQuery) meant the open web platform was cool again
- New browsers came online (Firefox, Chrome, Safari,) Opera continued to fight for the open web
- Even IE was eventually reborn since they had real competition on multiple fronts
- A new dedication to standards development by the W3C

# Now? An Explosion of Devices & Browsers





# Browsers!



# No Longer Just the Big Blue E

- Resolutions from 240 x 320 to 3840 x 1080 (or more)
- Pixel densities from 72ppi up past 300ppi
- A broad range of input options (keyboard, stylus, finger, mouse)
- A dozen or more major and hundreds of minor browser versions in the wild
- A rapidly changing open web platform landscape



# So... Let's Make More Rules?

Initially, developers and designers tried to navigate this complicated new reality by creating new rules.



# New Rules? Not so Great.

As soon as a new rule was created, it would start to fall apart.

People built “iPhone” sites because the iPhone was the only mobile device worth targeting.

People test for touch APIs and assume that those users don’t have a mouse.

# Blame Pesky Device Manufacturers & Browser Vendors

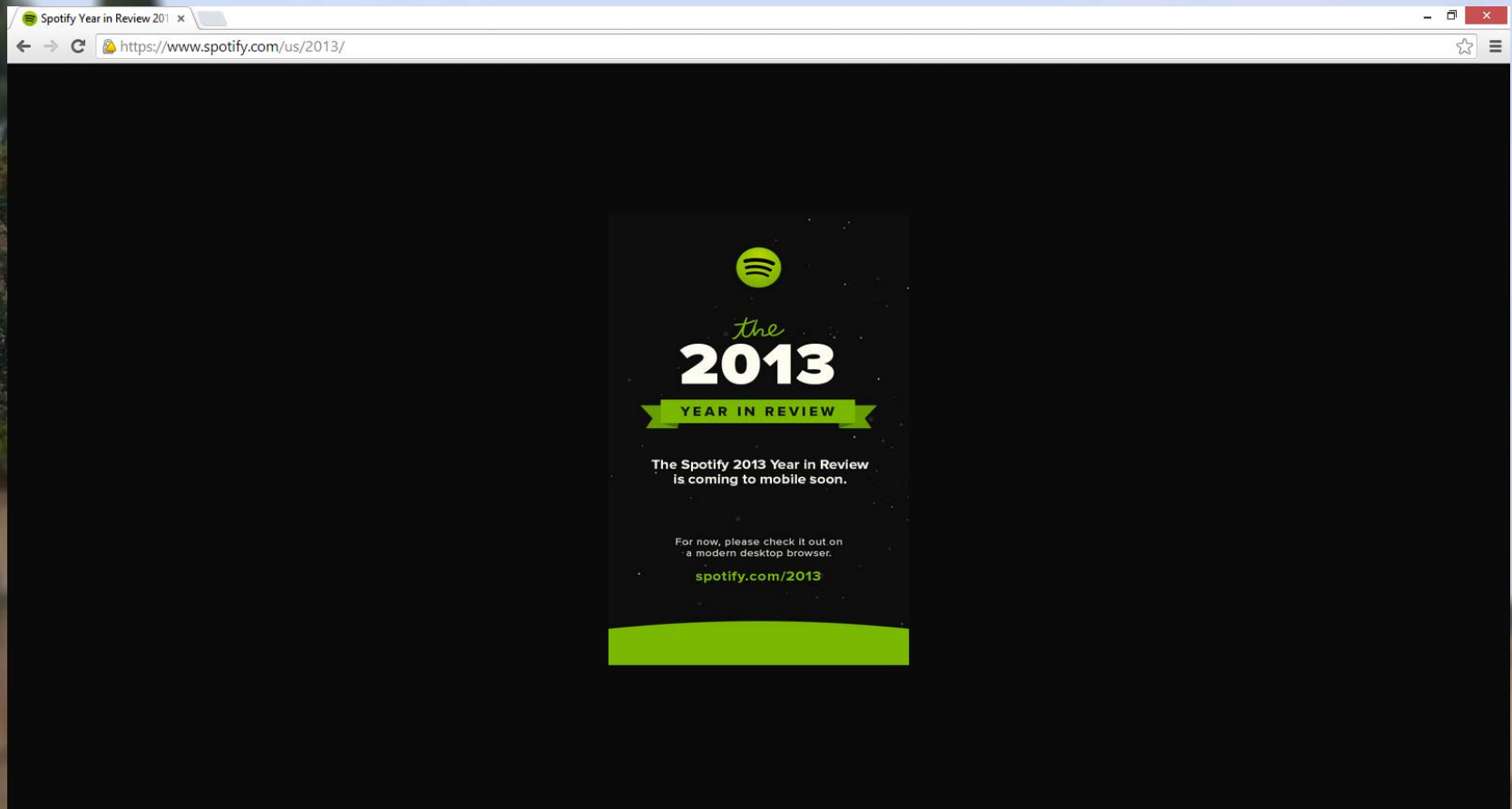
As Android's huge growth over the past few years, and the presence of Chromebooks and Windows 8 laptops with both mouse and touch capabilities have proved, those new rules have a short shelf life.



Nothing could ever challenge  
the iPhone, right?



Modernizr.touch == true  
means you've got a phone?



Modernizr.touch == false means  
you can't interact with the screen?





# One Size Fits All?

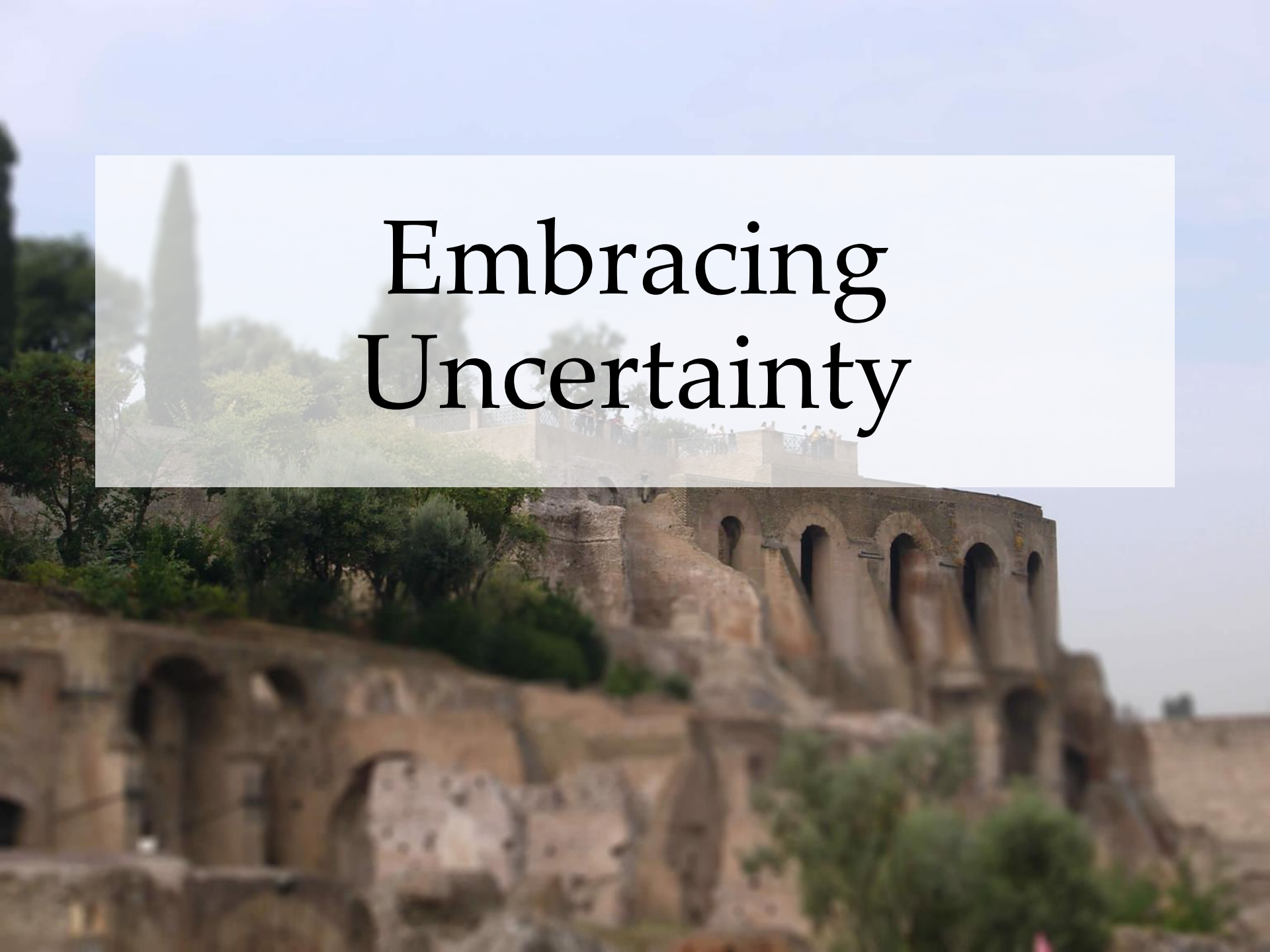
Even patterns like Responsive Web Design, which some people see as the solution for everything, can fall apart when faced with complicated application patterns, and, if you're not careful, bandwidth limitations.

# So What Should We Do?

The best way to approach the web today is to forgo hard and fast rules and one-size-fits-all solutions and design for uncertainty.

This is your best bet for creating future proof web solutions.


# Embracing Uncertainty





# Embracing Uncertainty

What follows are a series of high level ideas that will allow you approach compatibility in a nimble way and will help you when you're faced with the web's uncertainty.



Don't Blame the Web  
for being the Web

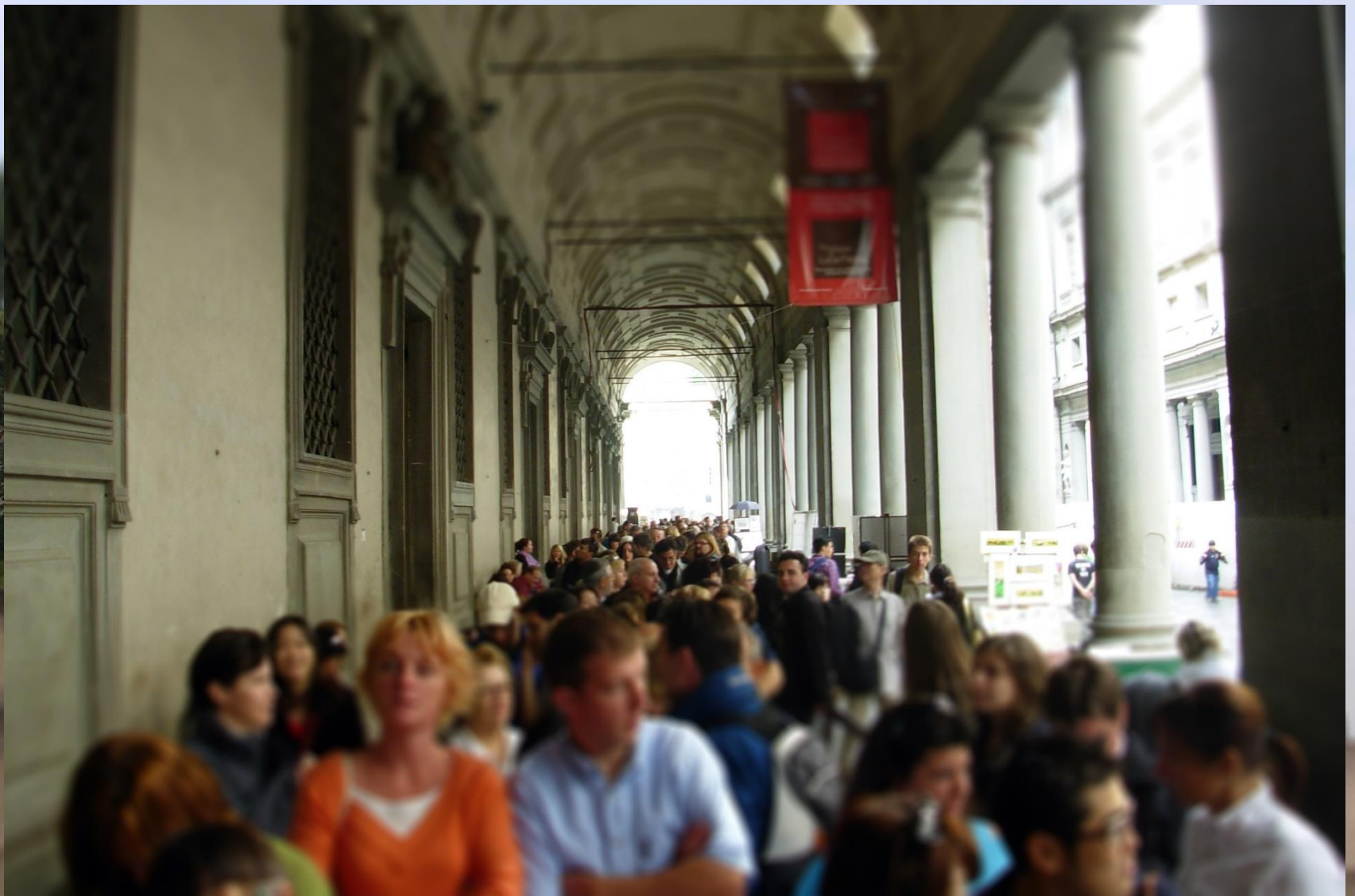
# Don't Blame the Web for being the Web

- The web is a diverse place that's getting more diverse every single day.
- If you accept the web's diversity (and maybe even celebrate it) and you're getting angry about one thing (maybe Internet Explorer 8) or another (the stock Android Browser) just take a minute to remind yourself that *this is just the way the web is*.



# This is Just the Way the Web Is

Fixating on the web's shortcomings does no one any good. We need to focus on what the web provides (billions of people,) do our best to make the web a better place and accept the diversity as the price of admission to reach billions of people.





Identify and embrace  
your audience



# Identify and embrace your audience

While you can look at web-scale statistics for browser and operating system market share to get some idea of where things are and where they're going, the only metrics that truly matter are those for *your* audience.

# Who? What? Where?

- Who are they? What do they use?  
Where do they live?
- Knowing this information makes your life significantly easier.
- Without knowing your specific audience you're just guessing.  
Engineers shouldn't guess.

# Act on the Info

If the majority of your audience is coming in on mobile, then you probably want to skip the multi-megabyte images and autoplay HD video you were planning on featuring.



# It Can Make a Big Difference

Or, maybe you get a large percentage of your visits from some place halfway around the world.

For example, your servers are in Virginia and your audience is in Australia.

*Time to make sure your CDN is up to snuff.*



Test and Pray for the  
Best

# Test and pray for the best

Once you've identified your audience, where they are and what they're using it's time to define the technical demographics you're going to target.





Photo by [Andreas Dantz](https://www.flickr.com/photos/szene/8220511232/) <https://www.flickr.com/photos/szene/8220511232/>

# Testing Could Look Like This:

- Samsung Galaxy S3
- Samsung Galaxy S4
- Samsung Galaxy S5
- Samsung Galaxy Note III
- Samsung Galaxy Note 4
- Nexus 5
- Kindle Fire
- Motorola Droid X (Android 2.3)
- Google Nexus 7
- Nokia Lumia 920
- iPhone 4S
- iPad 2G
- iPhone 5s
- iPhone 6
- Chrome (latest) Mac, PC and a Chromebook
- Firefox (latest) Mac and PC.
- Opera (latest) Mac and PC
- IE 8.0 on Windows XP
- IE 9.0 on Windows 7
- IE 10.0 Windows 7/8 - including a touch screen laptop
- IE 11.0 Windows 7/8/8.1 - including a touch screen laptop
- Safari 6.0



# Or Scaled Down to This

- Samsung Galaxy S4
- iPhone 6
- iPad 2G
- Motorola Droid X
- Chrome (latest) Mac and PC
- Firefox (latest) Mac and PC.
- Opera (latest) Mac and PC
- IE 8.0 on Windows XP
- IE 9.0 on Windows 7
- IE 10.0 Windows 7
- IE 11.0 Windows 8
- Safari 6.0



# Whatever your testing set-up looks like...

- Test on as many *real devices* as early and as often as you can. It makes a big difference.
- Try to have dedicated devices for everyone on your team to use.



Focus on optimal, not  
absolute solutions

# Focus on optimal, not absolute solutions

Your site is not an absolute thing. The best possible site you can have will be the best possible site for everyone that visits it. If that means it's a high DPI, 25MB monstrosity for a guy on a MacBook air in a coffee shop in Palo Alto or just a logo and an unordered list for someone on a rented-by-the-minute feature phone in Lagos, then that's the way it is.





# Embrace Accessibility

# Embrace Accessibility

If your site is accessible you're guaranteeing that you'll be able to reach the largest possible audience. You're also doing the right thing.



Photo by [Jeremy Keith](https://www.flickr.com/photos/adactio/89778576/i) <https://www.flickr.com/photos/adactio/89778576/i>



# Millions of Users are Directly Affected

Based on the 2010 [US census](#), 56 million Americans were classified as having a disability. That's 18.7% of the population. Not all disabilities would hinder the ability of a user to access the web, but it still breaks down to millions of users.

And that's just the US where stats are somewhat available.

# *Every* User is Indirectly Affected

The following slides list some of the more obvious ways that accessibility techniques can help *all* users.

# Accessibility Guidelines + the Multi-device Landscape

- Provide text alternatives for all non-text content
- Ensure that information and structure can be separated from presentation
- Make all functionality operable via a keyboard interface
- Provide mechanisms to help users find content



# Accessibility Guidelines + the Multi-device Landscape

- Help users avoid mistakes & make it easy to correct mistakes that do occur
- Support compatibility with current and future user agents

# Don't Stop There

In addition to being a vital link for disabled users, *all* the WCAG guidelines are also going to make your site more robust for all users. These examples are just the most obvious ones.



# Lose Your Technology Biases



# Lose your technology biases

Tech folks generally have great hardware and new, high powered smart phones and tablets. Most other people in the world don't. Tech folks tend to forget that.

# The iPhone isn't the only mobile experience

- At the height of the iPhone's dominance as a mobile platform, it was typical to base mobile web designs on the interface and interaction model of the iPhone.
- And then came Android.

# Even Now...

- Many designers and developers have never been hands-on with an Android device.



# Where Do We Put the Back Button?

If your vision of the web is iPhone-centric, inserting a back button into your web UI seems like a good idea. The thing is, every Android device has a back button built in, either as a dedicated software button on screen or as a physical button on the device.

# Closed. Won't fix. Can't Reproduce.

*"That animation is super fast on my machine."*

Another painful example of the trap tech folks fall into is with application performance.

Most people don't look critically at their application performance in enough devices to truly test performance.

# Contrary to Popular Opinion Internet Explorer Does Exist

People don't test enough in Internet Explorer.

Whether it's Windows-based developers working all day in Firefox or Chrome or developers on a Mac not wanting to fire up Parallels, people don't test in IE early or often enough.





# Contrary to Popular Opinion Internet Explorer Does Exist

I know it's the bogeyman, but it remains a huge portion of the browser market with hundreds of millions of users

Yet, people treat it like an afterthought.

# This is Why You Hate IE

Since so many people save IE for later on in the development process, or downright ignore it, their only experience with the browser is one of shock and betrayal.

If IE was constantly sneaking up on me and punching me in the face because I wasn't paying attention, I'd be mad at it too.





Embrace Empathy

# Embrace Empathy

- Don't blind yourself to what your audience actually is by assuming that they are just like you.
- Try to get in their shoes instead of assuming everyone else is in yours.



# Lose Your Stack Biases



# Lose your stack biases

Your users don't care if your stack is clever. What they care about is the speed, usability, look and feel, interactivity and features of your site. If your stack isn't adding to one of those then you might be going down the road to stack obsession.



Photo by [Jeff Kubina](https://www.flickr.com/photos/kubina/278696130/) <https://www.flickr.com/photos/kubina/278696130/>



# Your Stack is Really Cool

At the end of the day developer comfort isn't the most important part of this equation. The experience of the users of your site trumps everything else. Or at least it should. To that end, doing things like pushing 1Mb of fancy framework JavaScript down the pipe on a site that's meant to be consumed on a mobile device over a potentially dicey connection is simply a terrible idea.



# Don't Turn HTML back in XHTML

While Front-end Model View Controller (MVC) style libraries and frameworks are great, these libraries and frameworks are really designed for *application* development- they shouldn't be used for every circumstance.

# We Already Had This Figured Out

For example, don't use one of these libraries in place of server side templates for a content site. One of the first lessons of web performance was that most of the performance hit on the page happened in the browser, not on the server. Templates on the server aren't a performance problem. Why, then, are we rushing headlong to push functionality that was handled perfectly well by the server down to the front end?

# The Uncertain Web





# Question Your Assumptions

At this point, I'm assuming at least half of you think I'm an idiot.

If so, I must be onto something.

Whatever percentage of these concepts you agree with or feel like are applicable to you and your particular situation, I urge to question your assumptions. Your users will thank you.

# Accept the Things You Can't Control

The web has *never* been a static platform, no matter how much people might wish it were so. You just can't control who's going to request your content. You can't control the browser or device they're using and you certainly can't guarantee things like the operating system, screen resolution, bandwidth or available system fonts.

# Today's web is a wild place.

Standards are changing on, in some cases, a daily or weekly basis; new devices are coming on-line at a furious pace and browser vendors are going at it tooth and nail. With an ecosystem like that, trying to collapse everything you do as a developer into something that can fit into a neat little box is a recipe for frustration.



# Embrace Uncertainty

Embracing the ecosystem for the wild mess that it is and developing with an eye towards the uncertainty the web will throw at you is the best way to reach whoever might want to get at your site or application with whatever they have in their pocket or on their desktop- now and in the future.

# Thanks!

[roblarsen](#) on Github

Twitter:

[@robreact](#) (art | culture | etc.)

[@roblarsenwww](#) (tech)

Blog: [htmlcssjavascript.com](http://htmlcssjavascript.com)

Books: [is.gd/rob\\_larsen\\_books](https://is.gd/rob_larsen_books)

New book! [The Uncertain Web](#)

My company

[palatinoconsulting.com/](http://palatinoconsulting.com/)

