Mobile Web Development For Portable Devices (CS420)

Welcome

- Welcome to CS420 Mobile Web Development. We're here to teach you to apply your web development skills to this Mobile Web Development because as you're probably aware the mobile web is kind of a big deal.

- Most statistics project that mobile web usage will soon over take desktop web usage, so you might ask what’s the difference between developing for the desktop web and developing for the mobile web.

Topics To Be Covered:

1) Introduction to Mobile Web Development
2) Differences of Mobile Web and Ordinary Web
3) Development Tools
4) Development tools for Desktop
5) Mobile Development Tools
6) Setup for Mobile
7) Using Development Tools on Mobile
8) Reverse Port Forwarding
9) Mobile Tools for iOS
10) Mobile First
11) Constraints of the Canvas

Introduction to Mobile Web Development

Introduction to Mobile Web Development

The mobile web is one of those over-used terms that has lost all of its meaning, or worse yet, continues to confuse and perpetuate the mobile myth.

If you were to ask web developers to define what the mobile web means, you would get as many different answers as people you asked. It is important then, to define what I mean by the mobile web and how you should be discussing and thinking about it.

Differences of Mobile Web and Ordinary Web

- Well mobile web development is really just normal web development with some additional key considerations and a few additional API's. We're going to assume that you're familiar with the building block HTML, CSS and javascript. The mobile web is also built on these technologies, although we are going to cover some specific advanced features that apply to the mobile space. Of course, there are a lot of additional considerations for developing for mobile.

- Things like smaller screen sizes and limited network bandwidth and computing power. Well you might think these constraints are a bad thing. In fact, developing mobile first forces you to focus on the key points of your user experience.

Differences of Mobile Web and Ordinary Web

- Before we get started, I want to be clear about what we're not going to teach you in this course. We're not going to be teaching mobile frameworks like bootstrap, or native app wrappers like PhoneGap.

- These are great tools, but we want to teach the fundamentals that you as a mobile web developer need to know first.

- Mobile websites' content must comply with users' expectations. In most cases, users visit mobile webpages in search of specific information, they want speedy access. Available content must therefore be clear, accessible, and needs to have undergone rigorous selection.

Differences of Mobile Web and Ordinary Web

- On a traditional website, depth and completeness are king: a user must be able to find all the information on a brand, a product. In many ways this is the opposite to a mobile web page as a mobile user only accesses essential information and functions.

- These differences entail that these websites' ergonomics must be adapted to phone and tablet format in order to allow the user to browse through the mobile webpages with ease.

Difference of Mobile and Web Development

For example, the Google News experience on the desktop shows a lot of information, but the mobile experience focuses on just the news stories. Pushing all of the secondary information out of the way.

Google news

Development Tools

- In this topic, we're going to be discussing the mobile development tools, showing you how to hook up the DevTools to your mobile browser. The Chrome developer tools, which enable you to analyze and debug, your web applications.

- Now you may already, be familiar with the Dev Tools. But, what's really exciting, about them, is that you can now, easily ,use these tools, for your mobile debugging. Let's get started.

Development Tools for Desktop

- On any web page, like www.vu.edu.pk, you can open the con developer tools in the google chrome by clicking on the shortcut listed in tools, developer tools.

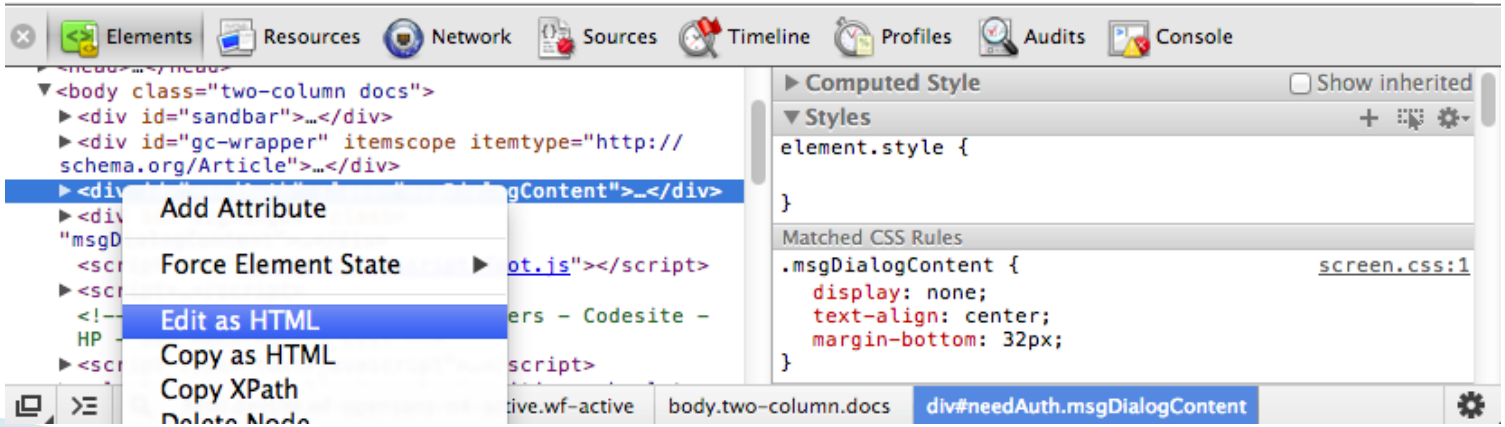
- Now technically speaking you don't need a mobile device in order to take this course or even to test out most of what you'll learn. But you're missing a lot of the experience of mobile web development, without a device. Preferably at least one Android device.

- You can use Chrome's great mobile debugging tools. Which we'll be using in this course. But, the more devices you have, and the more variety, the better.

Development Tools for Desktop

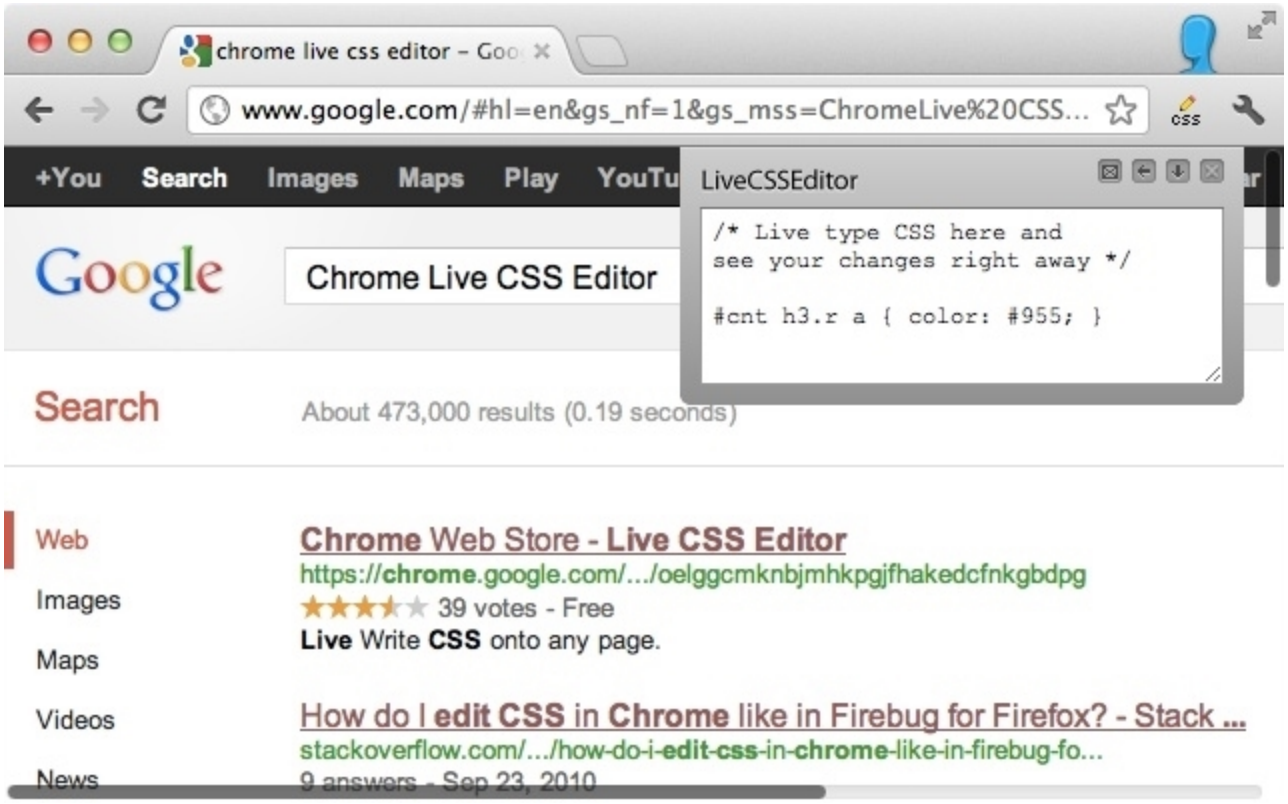
- A traditional website is accessed via computer, with help from a mouse and a large screen. A mobile website, on the other hand, is accessed via smartphone or tablet, with a smaller screen and touchscreen navigation.

- Chrome Dev tools provides a series of tabs that allow you to debug and inspect your web apps. For example you can select any element on the page by right clicking the element and selecting inspect element or as I'll do here you can click on the magnifying glass and hover over the area that you would like to inspect

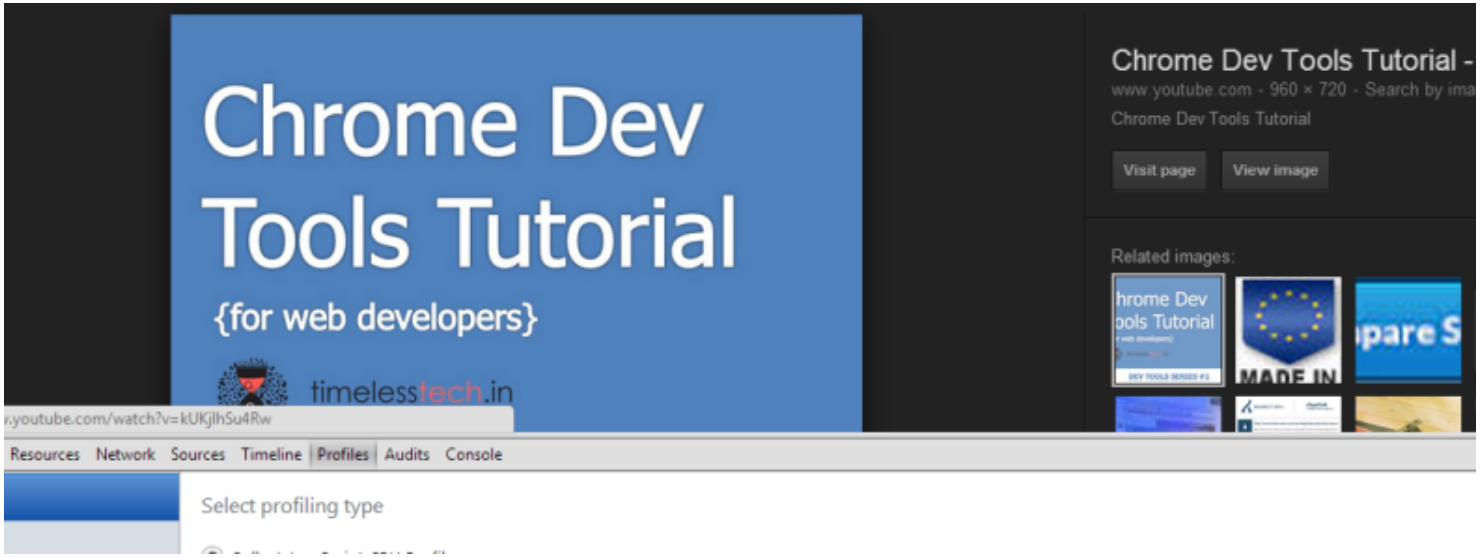


Development Tools for Desktop

- Let's change the text in one of the floor selectors. We can do that by simply expanding the element that contains it and then editing it live on the page.



Development Tools for Desktop

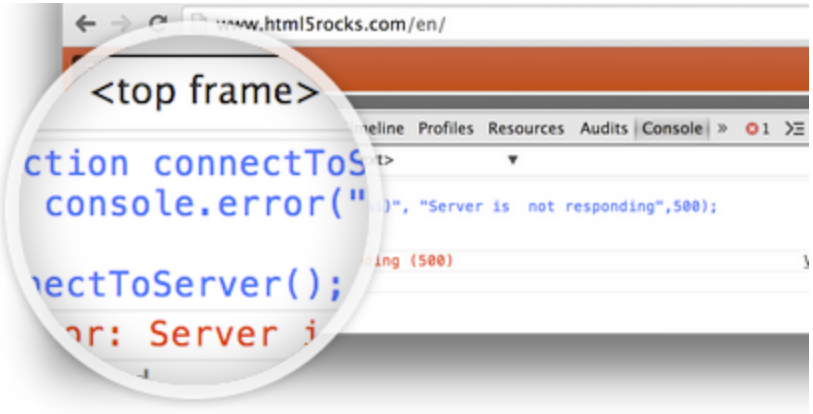


- You can also make live edits to the CSS. Let's take a look at changing the selected floor class. Here, I can set the background color to whatever I want. For example, blanched almond because who doesn't like blanched almonds. Because the background color is now pretty light, we may also want to change the text color. Chrome Depth Tools provides a handy color picker for that. We can pick any color we want.

Development Tools for Desktop

- Shift + clicking the color picker, allows you to cycle through the different color schemes. RGB HSL, and so on. If we scroll a little bit further, so our floor selector class, we can play around with the different settings there. You can use the scroll wheel to cycle through the different sizes. And we can set the border width in a similar fashion.

- As you can see, Chrome Dev Tools provides with rich functionality to edit and inspect your pages. And than was just one tab. We'll be diving deeper into the Network, Timeline, and Profiles tabs.



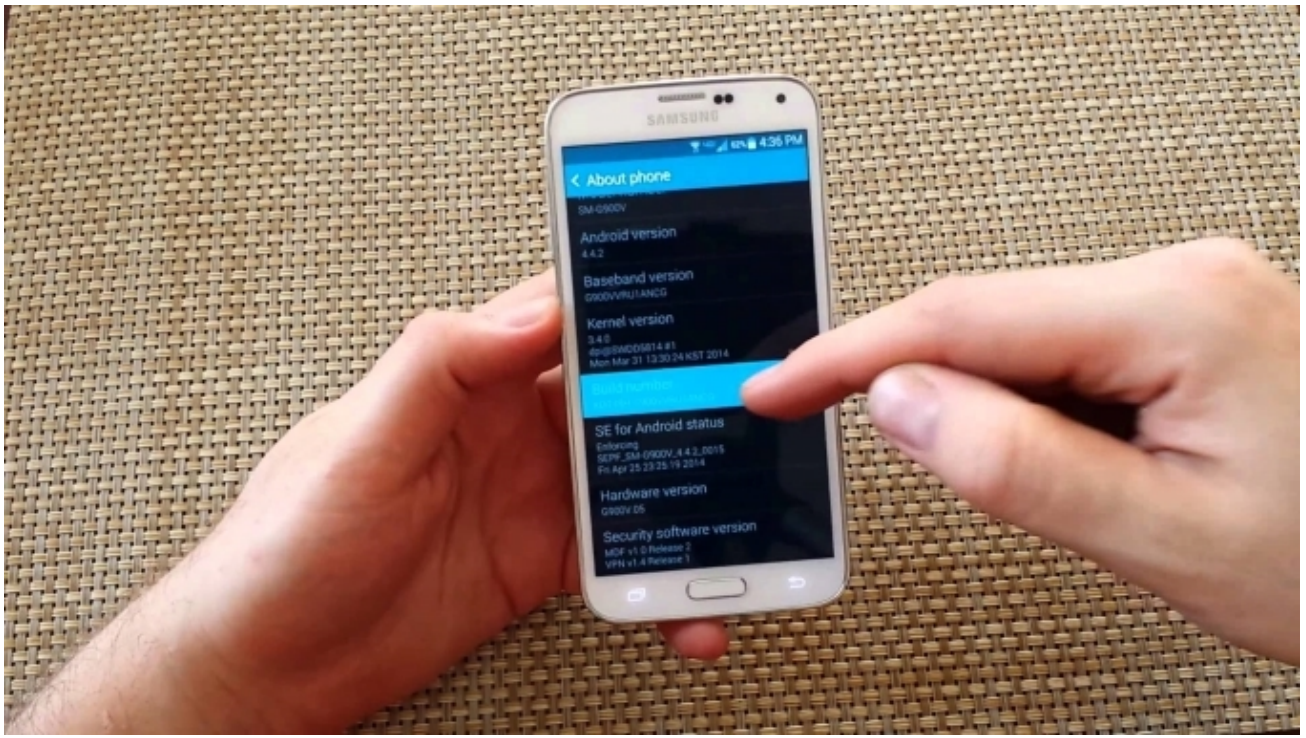
Setup For Mobile

- You also want to make sure you're set up correctly to debug and profile your web apps on mobile. We're going to show you how to do that with chrome dev tools and chrome for android.

- The set up is simple. All you need is an Android device, a USB cable, and your development machine. Let's take a look. Before you get started you need to turn on the Developer Mode in your Android device.

Setup For Mobile

- This may be different on any given device and you can check your device's manual on how to do this. In many cases though, you need to go to your device's settings, click on About Device and then click on Build Number seven times. Seriously.



Setup For Mobile

- Next you'll want to turn on USB debugging Again, this varies slightly on your given device, but this is usually located in the developer options.

- We also need to make sure we have the right tools. On my laptop, I have Chrome Canary and on my mobile device, I have Chrome Beta installed.

- Now that we have everything set up the way we need,
- open Chrome on your development machine and go to Chrome inspect. Make sure the site you want to debug is open on your mobile device and then connect your laptop to your mobile device via USB.

- Then confirm that you want to allow USB debugging. Back in our development machine, we can see a list of the attached devices and the Chrome tabs that are open on the devices. You can even open other tabs. We can also focus on specific tabs, you can reload them And you can even close a tab.

Using Development Tools on Mobile

- The best part of course is that you can inspect the pages that are running on your mobile device, from your development machine let's take a look.

- One of my favorite new features is the new screen cast mode. This allows you to drive the experience on your mobile device from your development machine. You can click on links, and see them update simultaneously on the device. As well as on your desktop.

- While continuing to cast a video in full screen mode to your TV. Now you can! With Chrome 35.0.1900.0 (or later), when you go full screen (with both HTML content/video and Flash) when you're already casting a tab, we'll size things correctly for TV, but show that content within the tab. On the TV, content should fill the entire screen.

Using Development Tools on Mobile

- To try this feature, you do need to be running the very latest and greatest version of Chrome. It's available today if you install Chrome Canary (https://www.google.com/intl/en/chrome/browser/canary.html) as a secondary browser on your computer, and it works on Windows, Mac, Chrome OS, and Linux.

- As you can see, you have all the familiar features from the development tools available for mobile now.

Reverse Port Forwarding

- Now all of these examples, we're accessing a live site. But you can also setup Port Forwarding to allow your mobile device to access a local server on your development machine over USB. Let's take a look.

- To do this, you want to make sure you have a server running on your local development machine. In this case, I'm going to use Python's simple HTTP server on Port 9999. Now to verify that it is actually working, I'm going to access that on the local machine and it's working fine. If I want to now access that same page on my mobile device, I need to set up Port Forwarding. I can go back to the Chrome Inspect page, click on Port Forwarding and set up a port forwarding rule.

- In this case, port 9999 on local host or IP address 127001. I enable Port Forwarding and click on Done. When I refresh this page, you'll see that Port Forwarding is now running on Port 80 80, and 99 99.

Mobile Tools for iOS

- At this point, we're all set up. So let's try it out. I'll open a new tab, navigate to localhost 9999 on the mobile device. And the page is ready to go.

- Now, that was really easy and it's also possible to do this on mobile Safari with the web inspector using the iOS web kit debug proxy. Now, that's a little bit harder to set up.

- Now that you are equipped with the right tools we can get to work. In the next lesson, we're going to get started with a mobile user experience.

Mobile First

- We will discuss now the tools which build designs that scale across multiple devices. And one of the first challenges with developing for the mobile web, is the display screen.

- The typically small display screen compared to your desktop or laptop device really forces you to focus on what's critical for your users, in fact this is what's led to the recent adoption of the mantra mobile first. Now, mobile first isn't about designing for the mobile web before you even think about the desktop experience.

- It's really a design philosophy of stripping down to your core user experience and then layering additional gravy on top if you have the appropriate space. For example you might put your company info links inside a menu rather then cluttering up your home screen with them.

Mobile First

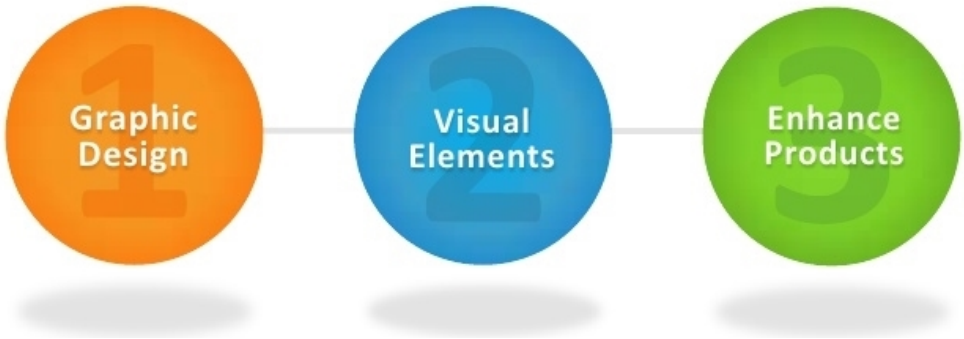
- What is Mobile First?

- Mobile first design is an approach outlined in 2009 by Luke Wroblewski. Simply, mobile first is an approach to responsive design: design for smaller screens first, then add more features and content for bigger and bigger screens. This design approach is also known as "progressive enhancement.“

Constraints of the Canvas

- When creating a visual design layout, it's tempting to think of your design surface like a canvas, a fixed surface. Now web design used to be like this. You drew on this canvas, it was a fixed size and shape, just like an artist draws on a canvas. What do you think the problems with this will be, as you move to a mobile first design.

- Fundamentally, the problem is all of these. With a layout that's designed for a particular screen size and aspect ratio, the user is going to have to zoom and scroll around in order to see everything and there will frequently be blank space left on the screen.



Screen Size for Mobile

When Mobile browsers first came along the content on the Web wasn't designed for narrow small screen devices. It was designed for Windows that were around 1,000 pixels wide and wider than they were tall with easy scrolling. To show this content into a tiny Mobile screen since rendering a Web Page designed for 1,000 pixels across and a 320 pixel wide screen would mean you'd be scrolling a lot. Mobile browsers basically lied about the Window width.

They made the Window act as if it were 980 pixels wide, even though the original iPhone was only 320 pixels across. This enabled sites that were designed for a 1024 by 768 screen, that is, that were around 980 pixels wide to fit on the Mobile screen. Although you needed to do a lot of Zooming to read the text sometimes. Unfortunately if your site did not happen to match that 980 pixel width you were either going to overflow or underflow the screen.

Screen Size for Mobile

Either wasting space or forcing the User to Zoom. In order to control this, Apple provided a viewport meta tag to be added to your HTML to control the default. For how big should my screen act on this page? The default is 980 pixels. So, if you put 980 pixels here, it would have no effect. The Mobile browser already defaults to 980. But setting a viewport tells the browser how wide the content is intended to be, and then the browser scales to make that size fit on the device's screen. There are two ways to use this tag.

Screen Size for Mobile

- The first way lets us take a page that was designed to be precisely one size. This is an application I built that's using a fixed layout, and it's a bit smaller than default. It's 916 pixels wide. Notice that it's not using all the space on the screen because the browser by default is assuming the
- layout wants to be 980 pixels wide.

- So let's a viewport that tells the browser it was designed for 916 pixels across. What would you put in the viewport to tell the browser that this should be 916 pixels?

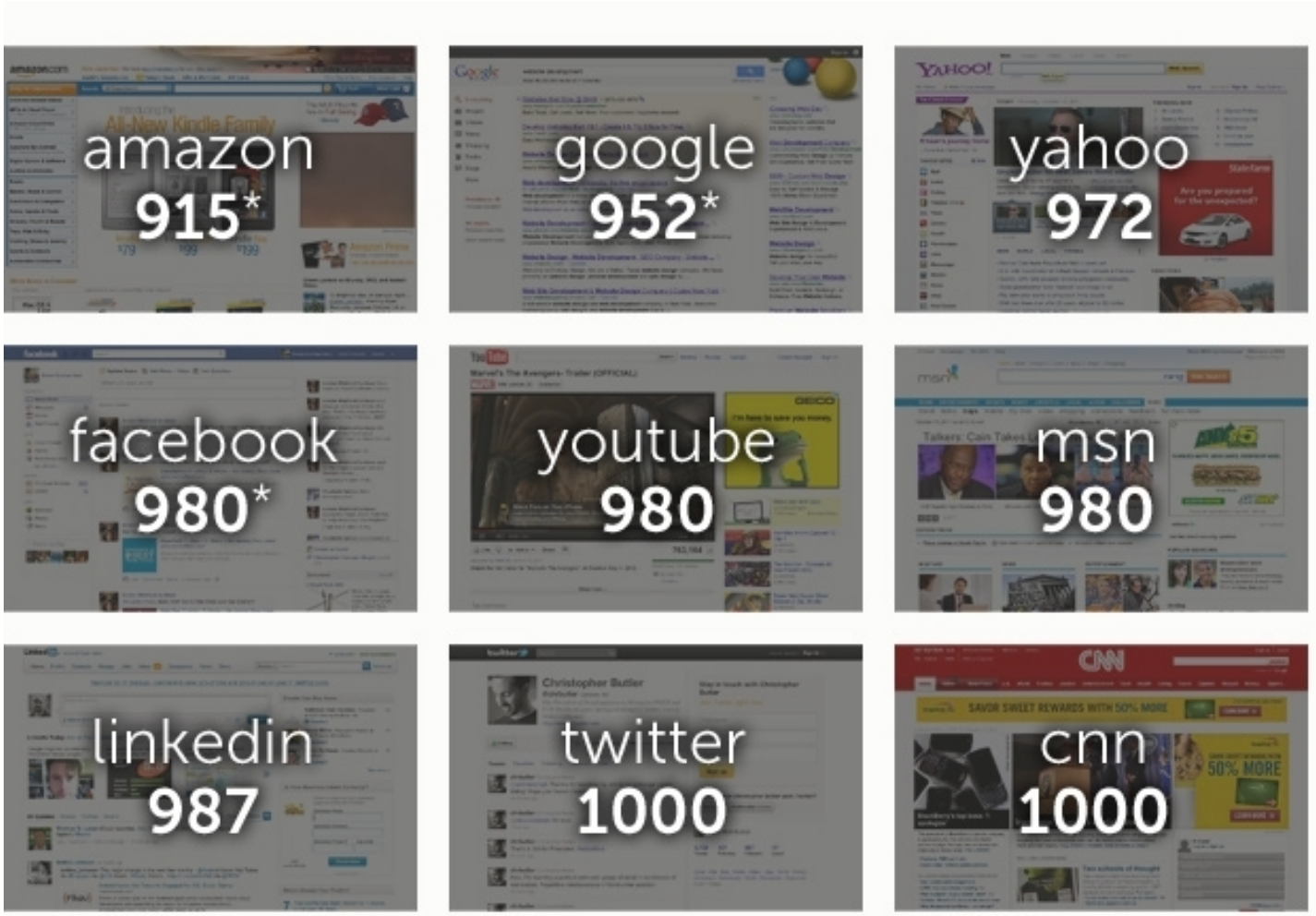
- Now the mobile browser will automatically scale that width to fit on the screen, no matter what size screen the user has, or whatever device width the user has.

Fixed Width

Let us take a page that was designed to be precisely one size. This is an application I built that's using a fixed layout, and it's a bit smaller than default. It's 916 pixels wide. Notice that it's not using all the space on the screen because the browser by default is assuming the layout wants to be 980 pixels wide.

- Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.

- Then, when we started surfing the internet using tablets and mobile phones, fixed size web pages were too large to fit the viewport. To fix this, browsers on those devices scaled down the entire web page to fit the screen.



A quick survey of nine popular websites reveals an average minimum width of 975 pixels, with all of them fitting comfortably inside a 1024 pixel-wide browser window. Most of the sites are set to a fixed width, except for Amazon, Google and Facebook which have partially flexible site designs.

Fixed Width

This was not perfect!! But a quick fix. So let's a viewport that tells the browser it was designed for 916 pixels across. What would you put in the viewport to tell the browser that this should be 916 pixels?

The viewport is the user's visible area of a web page. The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.

Now the mobile browser will automatically scale that width to fit on the screen, no matter what size screen the user has, or whatever device width the user has.

Rescaling On Devices

Let's take a look at an app given below across a few devices with different width screens, an iPhone at 640 pixels across and a nexus 7 tablet in landscape, at double that width 1280 and note that it rescales properly on both of these devices. Note this takes into account orientation as well.

Rescaling On Devices

The default scaling is to available screen width, so it will properly adjust when you flip the screen. There's also a height property on viewport which you could use instead of width, to control the scaling based on the height, but of course, most designs are built as width primary. That is, they're designed to scroll up and down.

Changing Viewport Settings

Although typically View-port is only being set on Load, you can actually play around with the View-port settings in the development tools to tweak it, and get it just right. If I go into the page and set the Viewport meta element contents from the mobile dev tools It will change the page, as if it had been refreshed, however, and this is where it gets a bit confusing, the zoom level is maintained by the browser across pager refreshes, so when you change viewport settings in the source code, and you're reloading, be sure to actually close the tab first. Don't just hit reload or it won't necessarily show the effects on the screen.

Changing Viewport Settings

For example, let's go back to our last bit of code. And change that width to something really different. Let's double the width to 1832. And now let's save that to the server and reload it on our mobile device. Our new View-port setting doesn't take effect. On the other hand,

if we close the tab first and then reopen it, our new View-port setting now takes effect. So pro tip, always remember to close the tab.

Downsides of Fixed Widths

- Your content is almost always going to be scaled.
- Users of some devices will have to zoom.
- Text & UI will look small on a large desktop screen

A fixed width lets you tell the browser what width your webpage was designed for and the browser will automatically scale that width to fit to the screen no matter what size screen the user has or what device width the user has. Now what would be some of the downsides or side effects of using a fixed width view port?

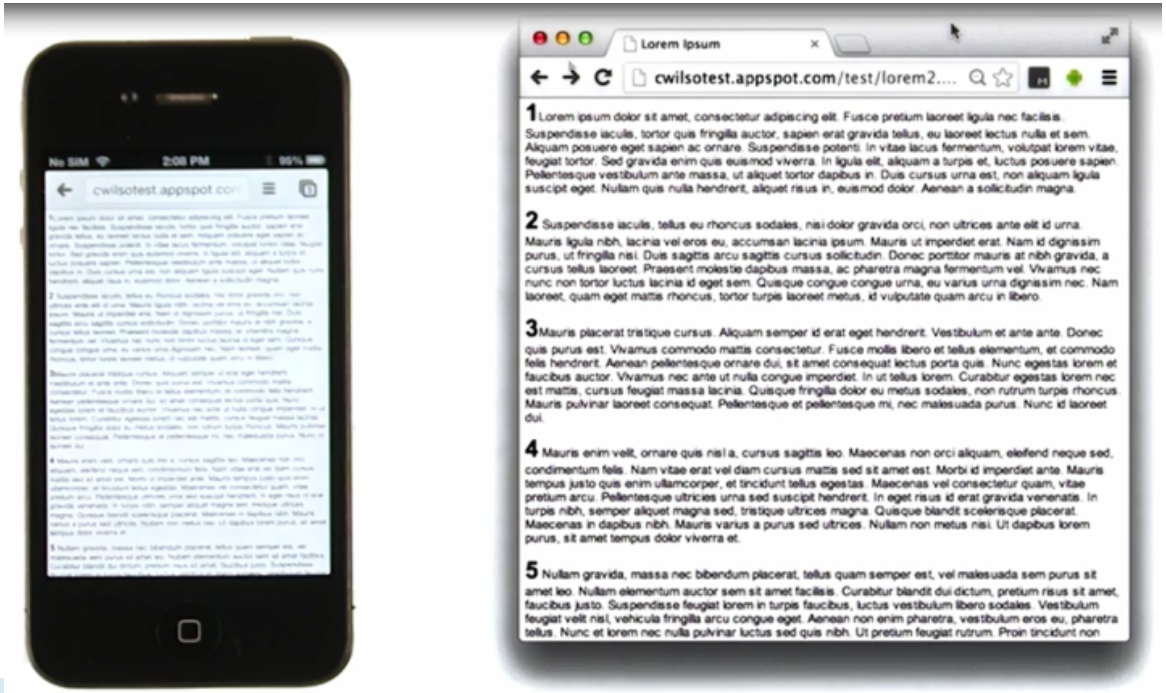
It is true, that using effects with viewport, means that your content, is almost always going to be scaled, and not match the native resolution. And definitely, in some cases, users are going to have to zoom, to be able to see or read anything, particularly text. Since, everything, all scaled depending on the size of the screen, fitting the current layout, precisely, into the users screen.

Downsides of Fixed Widths

Whether, it's a four inch phone or a 12 inch tablet, but actually, your text in UI will not change at all on the desktop. Because, currently, at least, none mobile browsers, don't respond to the view port at all. In some cases of course, it's the right thing to do, if you just want to get the UI to fit on the screen, to use, fix with view port, like in my sync app, but of course those dials are awfully small, which makes the UI very hard to use.

Text Reflow

- Let's talk about how to make truly scalable pages. Now, HTML, by default, is supposed to reflow text anyway. And text sizes are supposed to be consistent. But the worst side effect of this viewport stuff, by default, is that, even with nothing set, the mobile browser is going to reflow text and render as if the window were 980 pixels wide. And then scale it to fit it on to the screen.



Text Reflow

This may mean that the text looks really small on my mobile browser by default. So, the second way to use viewport is if your page knows how to adapt to width. For example, if it knows how to wrap the contents based on the screen width. You can simply set the width to device width which tells the browser, my website knows how to adapt to your width. This is really the best approach. To build applications that scale their own layout and make intelligent decisions about how to do so, rather than just trying to scale a fixed layout to fit the screen.

<meta name="viewport" content="width=device-width">

Device Width

So let's try this out on this page. Let's add a device with meta element to a page, and refresh it on the mobile browser. This is what our page looked like before. Now with the meta tag in place. Let's try reloading it. And you can see, the page now chooses a better size because it's reflowing at the native size of the screen.

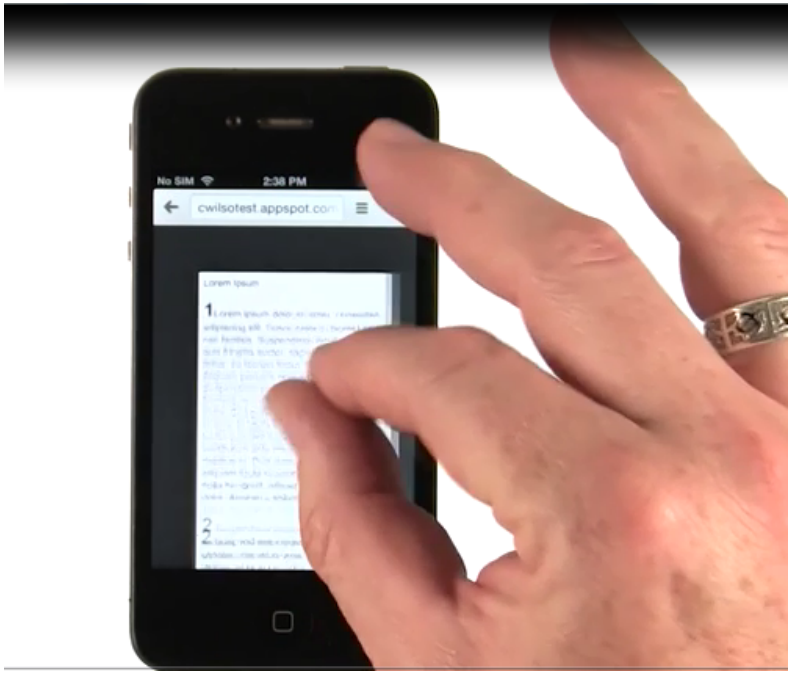
Other Viewport Controls: Initial Scale

On iOS, if you only set width to device width and you don't set the initial scale, like in this page, when you rotate the screen the iOS web engine will keep the same view port width and rescale it to fit across the landscape screen. It's just stretching the portrait layout, to fit, across the landscape width. Even though I've set width to device width, iOS is still scaling the landscape width.

In fact, the interesting bit is that even if you load this page initially in landscape mode, it still thinks it's the portrait width, it just rescaling it to fit in the landscape screen. Now, if you have the same page, but you set the initial scale to 1 in the viewport meta-element along with setting device width, it'll change the viewport size when you rotate, instead of rescaling. You can see now the window size is 480 pixels across. It's the landscape width, not the portrait width.

Initial Scale

I did want to mention at this point that there are other viewpoint controls. There's an initial scale property that lets you set what the browser's initial scaling factor will be. It defaults to one and usually, you don't really want to mess with it. If you change it to another number, this changes the initial zoom factor. And the user will probably have to pan or zoom. There is one very critical use to initial scale though.



Initial Scale

So, in short, this is what you really need to use as your default boiler plate viewport meta-element. You'll need the initial scale, so that IRS, when flipping from portrait to landscape mode will still scale correctly. An interesting side note I discovered is that on iphones - Although they are changing the viewport size properly if an initial scale is that, they are also changing the default font size for the document and orientation change, effectively zooming up the text when you go to landscape mode. This means you should probably set a default font size of the page, not just use an percentages. You may want to use a reset style sheet to do this, if you aren't already.

Fixed VS Device

So really, fixed viewport widths are historical. Resizing by default was an attempt to shoe horn the desktop web into a mobile device. Fixed viewport sizes were a quick way to provide some minimal controls on that resizing, with device whip, gets us back to the same scalable sized canvas that the desktop web has, so the right way to do fluid flexible design, in the modern mobile web starts with this tag. This marker let's the browser know your one of the cool kids and you know what you are doing.

There are also minimum scale and maximum scale properties too. Which is a way to limit the extents, that the system is allowed to scale the page on the device. That seems a little esoteric at first, but, there is one use that, I wanted to mention I don't want, you to think, that I'm biased.

Min and Max Scale

So, after describing how you need to add initial scale, for iOS to work as expected. I wanted to give equal time, for, Android. If your page, with device width happens to forcibly, overflow the page, like this page, where I've added an element that's very wide, Android will actually, do some rescaling, when you change screen orientation, to, try to get the whole page to fit onto the screen. Now, unfortunately, it gets it wrong, and it tries to zoom it the wrong way.

Now, the only way that I found, to get around this, other than not overflowing the screen to begin with, is to set the minimum scale and maximum scale properties, to one, which will prevent the user from zooming altogether. As well as, disabling automatic scaling. Let's do that in the dev tools now. Now let's try rotating again.

Min and Max Scale

We're back to normal here, and now, you see that even in landscape mode, we get the proper width. Now, the problem with this, of course, is i can't zoom anymore. We really don't want to disable zooming, for, your users; this is an accessibility problem. So, don't do this in production pages, but, it can help your testing, in, general just don't overflow the page and you'll be good.

User Scalable

There is one more property you can set. You can disable users zooming entirely by setting user scalable to no. Fundamentally, you really shouldn't do this. It's bad for accessibility and you're more likely to just annoy your users by preventing them from zooming than to make their experience any better.

Viewport Units

There is one more thing, on viewports. You're naturally, going to want to lay out, elements on the page, relative to the size of the viewport. Particularly, when you're sizing columns on the page, for example. So, we have a new unit type, in, CSS called, viewport units.

You can use these units, to size things, in percentage of width of viewport or percentage of height of viewport. Without having to push percentage sizing, everywhere, which makes things a little easier. The really exciting thing, is, these unit types even work in desktop browsers, as well as in mobile browsers.

VMin and VMax

There are also vmin and vmax units, which let you size things based on the smallest or largest units of width and height. This helps make layouts that stay consistent across portrait and landscape mode. Using this feature, it's pretty easy to create a button that takes up 1/3rd of the biggest square that will fit in the screen. Now using these units, the button stays as large as it can be and still fit in a third of the space.

Separate Desktop and Mobile

So you've had about wraps it up, for how to design for a particular view port. With all of this complexity though, you might have noticed, some developers decide to build two separate applications, a desktop app. And a separate mobile web app, but there are some problems with doing this, which of these problems might you encounter with having separate mobile and desktop sites?

Rules for Mobile Only

First, you really need all your features to be available and fully usable on the mobile version. If you try to guess what content or features your users won't miss on the mobile, you'll probably get it wrong. It might seem logical, right up until the point where I try to browse store inventory for my mobile device, only to find that feature is not availbale. Secondly, with two apps or sites, there's a natural tendency to focus on one or the other and to get out of sync between them.

Maintaining two separate applications and keeping them fundamentally the same is hard to do. And finally, there's the problem that it's difficult to identify when you really want the mobile version. My tablet for example has nearly as big a screen as my laptop, and it's actually higher resolution than my laptop. Mobile design is actually more of a spectrum that spans all the way from low-resolution devices up into desktop design.

Rules for Mobile Only

At the very least, there are three simple rules if you want to offer a mobile-only site.

First, you want to make sure a user can still get to the full site from the mobile site if they want to, in case you forgot any features.

Secondly, you should put canonical URL in meta-information. This is commonly done for search engine optimization, but it's also important to keep your mobile site and your desktop site together.

Finally, if someone links to a specific page on your desktop site, you shouldn't transfer them to your top-level mobile page. You need to transfer them to that specific page or its equivalent, on the mobile site. It's really frustrating when you get redirected to the top-level page.

Don’t Block Access

It's great if you've sunk development money into developing a native mobile app, and it's fine if you want to make sure your users know it's available, but you shouldn't block access to your web application with a full page ad for your native mobile app. Make this offer unobtrusive, and make my decision to ignore it sticky. Now, you can use smart app banners, in iOS, to make this offer a little less intrusive, but if I open a URL in a browser, I probably want to stay in the browser.

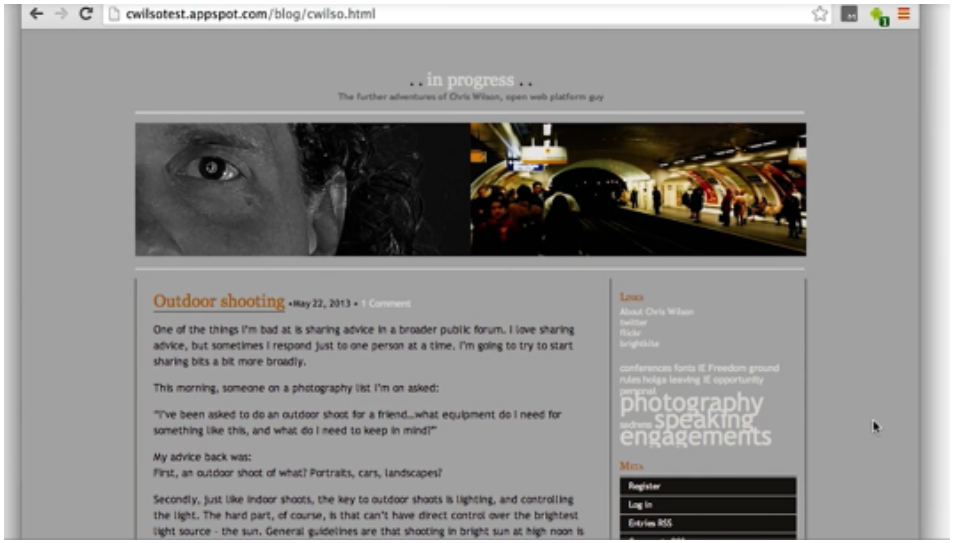
Building Fluid Designs

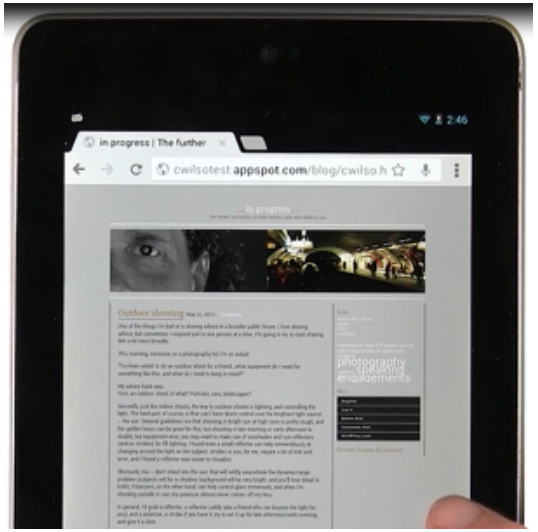
Fluid Layouts

As we discussed already that one of the first challenges of developing for mobile devices is the screen. As mobile developers, we need to design pages that can adapt across various screen sizes and scenarios. This will prepare you for building fluid layouts, designs that can adapt across different screen sizes. And hopefully we'll break the bad habits of designing to the fixed page.

Blog Layout

Now, if you resize this window, nothing really changes. Nothing adjusts other than the amount of margin space. It's really a fixed width layout. Now, let's take a look at this on the phone. Now here, I get a very different version of a blog. It fits well on the phone, but I'm only getting this because blog is running on Wordpress.

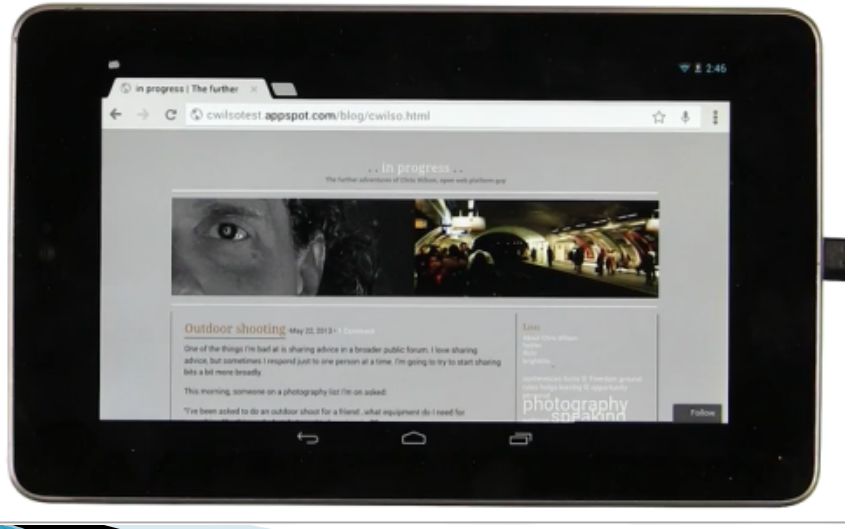




Blog Layout

And, Wordpress is configured to give a totally different version to phones. But, that's not actually what I want. Because I really want to have a shared layout between the two.

Let's take a look at this on a intermediate device. My seven inch Android tablet. Now here, what I'm getting is essentially the desktop version. I can still scroll through the site but you can tell all the text is pretty small. Although interestingly, if I rotate to landscape mode, it adapts.



Blog Layout

Now why does it do that? Well it turns out it's our old friend viewport. There's now viewport set in this page. So, what does that mean? Does layout happen if the device flipped? Or does layout happen at the magic 980 pixel width and then get rescaled to fit on the screen?

Digging Around

So if we dig around the page with the developer tools, we'd see a lot of pixel widths. Including the main content of the page, which is inside this page element. And you'll notice it has a width set in pixels. If I turn it off, well, the page is still not working properly. But at least we've started.

Instead of this fixed code let's use a different technique, Fluid Layout. Fluid Layout means I should stop fixing all those widths in absolute terms like pixels or points and start thinking in terms of areas of screen real estate. Percentages of the widths, for example, and sizes based on the initial font or Em units.

Fixed Viewport Width

With all this complexity, you might ask, why don't I just set the viewport width to 760 pixels? The width of that column of text I showed you in the Dev tools, and call it a day. Is it because that would be too easy? Or because, then my site would scale the same layout across all devices? Or because the fonts would be wrong in some way?

Well, the answer isn't because this would be too easy there really is a good reason for why you don't want to set viewport width to a fixed size. Namely that then your site would scale the same layout across all devices, you'd get the same layout of text and images on a four inch cell phone as you do on a ten inch tablet. And of course this means the fonts would be are consistent size with that layout. You'd get the same size fonts rescaled on to a four inch phone as you would on a ten inch tablet. And fundamentally this would result in a poor user experience across mobile devices.

Fixed Viewport Width

Well, the answer isn't because this would be too easy there really is a good reason for why you don't want to set viewport width to a fixed size. Namely that then your site would scale the same layout across all devices, you'd get the same layout of text and images on a four inch cell phone as you do on a ten inch tablet. And of course this means the fonts would be are consistent size with that layout. You'd get the same size fonts rescaled on to a four inch phone as you would on a ten inch tablet. And fundamentally this would result in a poor user experience across mobile devices.

Set Viewport Width

So let’s go fix my page. Now the first thing we need to do in order to make a mobile friendly page, is to set the viewport. The mobile browser needs to know that this webpage knows how to format itself on devices with varying width. It doesn’t need the mobile browser to pretend that it’s a desktop screen with a given width. Once you set the viewport correctly, it will be able to resize properly and lay out properly. So what should my viewport meta say?

Set Viewport Width

That's right, you need to set the width to device width, and optionally, you can set the initial scale to one if you want, to help avoid some iOS device formatting issues.

Making it Fluid

So now that you know how to support the view port meta, lets poke around in the developer tools to see what we need to do to fix this page. Now, the biggest problem here is that the page has a fixed column width. If I resize the window, you'll notice the column of text doesn't actually change size, just margin position. As we mouse around the elements inside the dev tools, we can pretty quickly find, the first offending element.

This page element here actually has a width set on it. Let's disable that and see what happens. Well, it certainly changed things, but I don't think that it actually improved things very much. Let's keep digging inside the content and see what we can find.

Making it Fluid

You will find a rapper element having a width of 100% already. Which is good. It means that it's not preventing us from resizing. But this content element does have a width set in pixels. Let's disable that. Now as we resize the page our content is fine but the sidebar is appearing and disappearing. Sure enough, it has a width set.

If we set its width as a percentage also, now we're resizing a little bit better. So now let's go back up to the page, and let's try giving it a width but let's give it a width in view port units instead, now our only remaining problem seems to be that the header is actually not resizing as we want it to let's go take a look at the header image again.

Making it Fluid

The image has a max width set, but not just a width. So, let's set its width. And now, everything seems to be resizing well. Now, this isn't perfect, but all that I really wanted to get at was that as you try to transition pages from fixed layout to fluid layout, The core things to look for, are fixed pixel sizes and lack of percentage resize.

Use Fluid Layouts

So there's really two rules for building fluid layouts. Use percentages or viewport units, something that'll size as you resize the window. And then, test while you're resizing the window. Let's take a look at how this worked out on the mobile device. We went from this layout without any changes to my blog. To this layout, definitely an improvement. I don't have to scroll left or right to see everything on the screen. And, you may notice the font size is actually a bit larger now. The font size was always set in M units, although it's actually still a bit small for a mobile device. So, we should probably bump that up. Just take a look on the tablet.

Use Fluid Layouts

In the tablet layout, you will get a definite improvement. This is actually a pretty readable experience. And, even when you rotate the screen, you end up with a nice reformatting. All this is really just a long winded way of saying, be sure to create fluid layouts. Be sure to reflow to use all the space on the screen and take advantage of every bit you can. And be sure you adapt to different screens as well. One tool that makes reflow a lot easier, particularly across very different screen sizes, is the new flex box layout in CSS. This new tool lets you stack elements in flexible rows or columns.

FlexBox Intro

FlexBox has been kicking around for a couple of years now, but it's gone through some pretty major revisions during that time.

It's just recently shipped in iOS 7, and it's been shipping in Android for a little while, but you should test your FlexBox layouts thoroughly, across browsers. This may be a little too bleeding edge for you, but, let's take a look.

We're going to walk through an interactive demo, talking you through each feature in FlexBox, as you change the page, live, in the developer tools.

FlexBox Compatibility

Okay, that was a ton of detail. Really I recommend you, just sketch out your design, and then party on it with FlexBox.

Flex Box is super powerful, and it's really super easy to use. Incidentally, I did mention, Flex Box is still making its way into all the browsers out there. So, there are some Flex Box polyfills out there, that can help you, like Flexy.

Fluid to Responsive Design

So these tools give us the ability to build designs that are fluid. But as it turns out, that's not enough. As you may have noticed as you went through this lesson, the experience of re-flow from a very small window size to a very large window size is not really a great one for the user. Now that you know how to build fluid designs, you need to learn to adapt those designs to radically different scenarios, like a four inch phone screen, and a 30 inch desktop screen.

Fixed, Fluid, Adaptive, and Responsive web design

- Fixed websites have a set width and resizing the browser or viewing it on different devices won't affect on the way the website looks.
- Fluid websites are built using percentages for widths. As a result, columns are relative to one another and the browser allowing it to scale up and down fluidly.
- Adaptive websites introduce media queries to target specific device sizes, like smaller monitors, tablets, and mobile.
- Responsive websites are built on a fluid grid and use media queries to control the design and its content as it scales down or up with the browser or device.

Different devices has different design

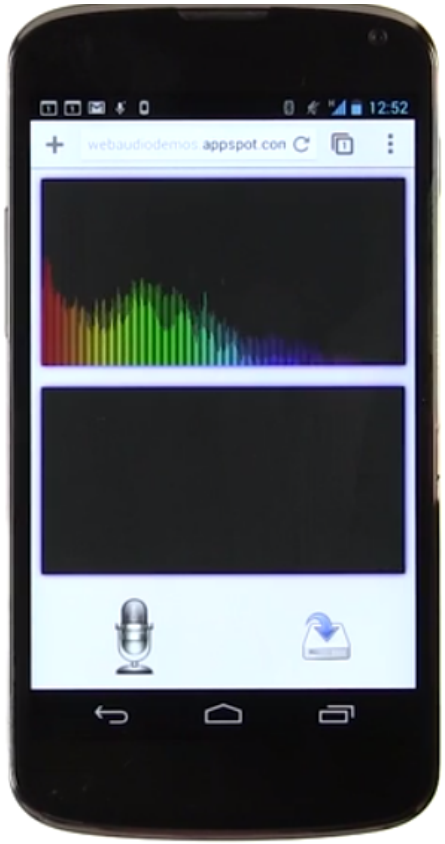
- we can't use reflow across all devices.

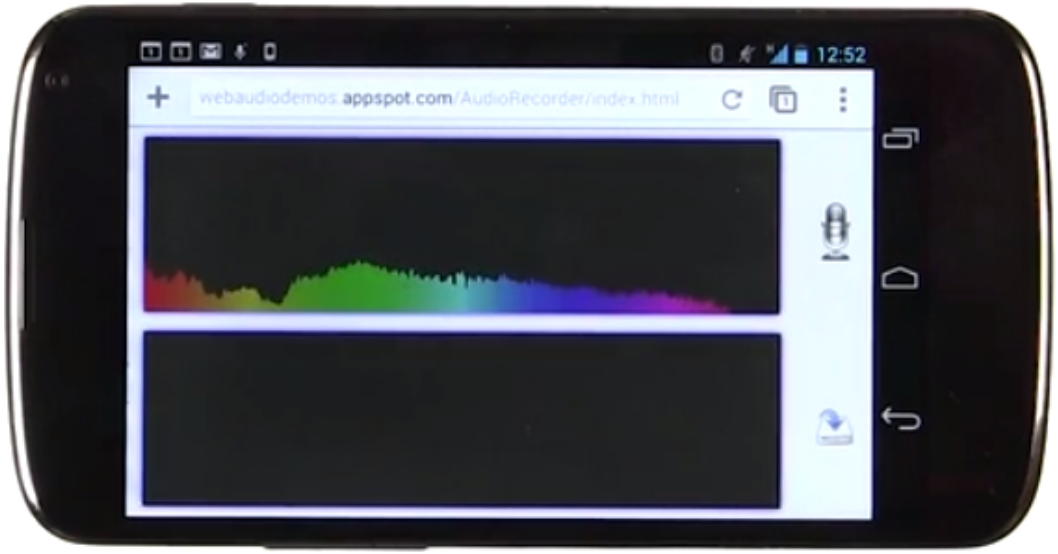
Why we can't use reflow across all devices?

Reflow across all devices

- New form factors come out all the time

Changing Layouts





Changing Layouts

Mobile

Tablet

Monitor

Responsive Web Design

- Responsive web design is the practice of building a website suitable to work on every device and every screen size, no matter how large or small, mobile or desktop.

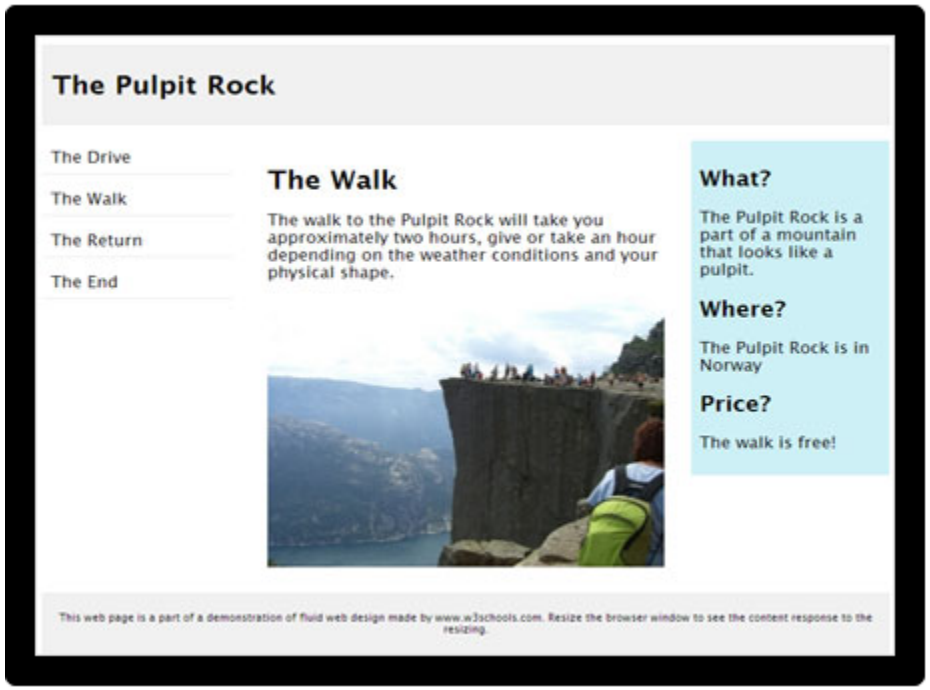
Responsive Web Design

- Responsive web design makes your web page look good on all devices.
- Responsive web design uses only HTML and CSS.
- Responsive web design is not a program or a JavaScript.

Why Responsive Web Design?

Responsive Web Design

- Monitor Tablet Mobile



Media Queries

- Media Queries is a W3C Candidate Recommendation—a widely reviewed document which is ready for implementation by browser vendors.
- It's an extension of media dependent stylesheets tailored for different media types (i.e. screen and print) found in CSS2.
- It uses the @media rule to include a block of CSS properties only if a certain condition is true.
- A media query consists of a media type and an expression to check for certain conditions of a particular media feature. The most commonly used media feature is width.

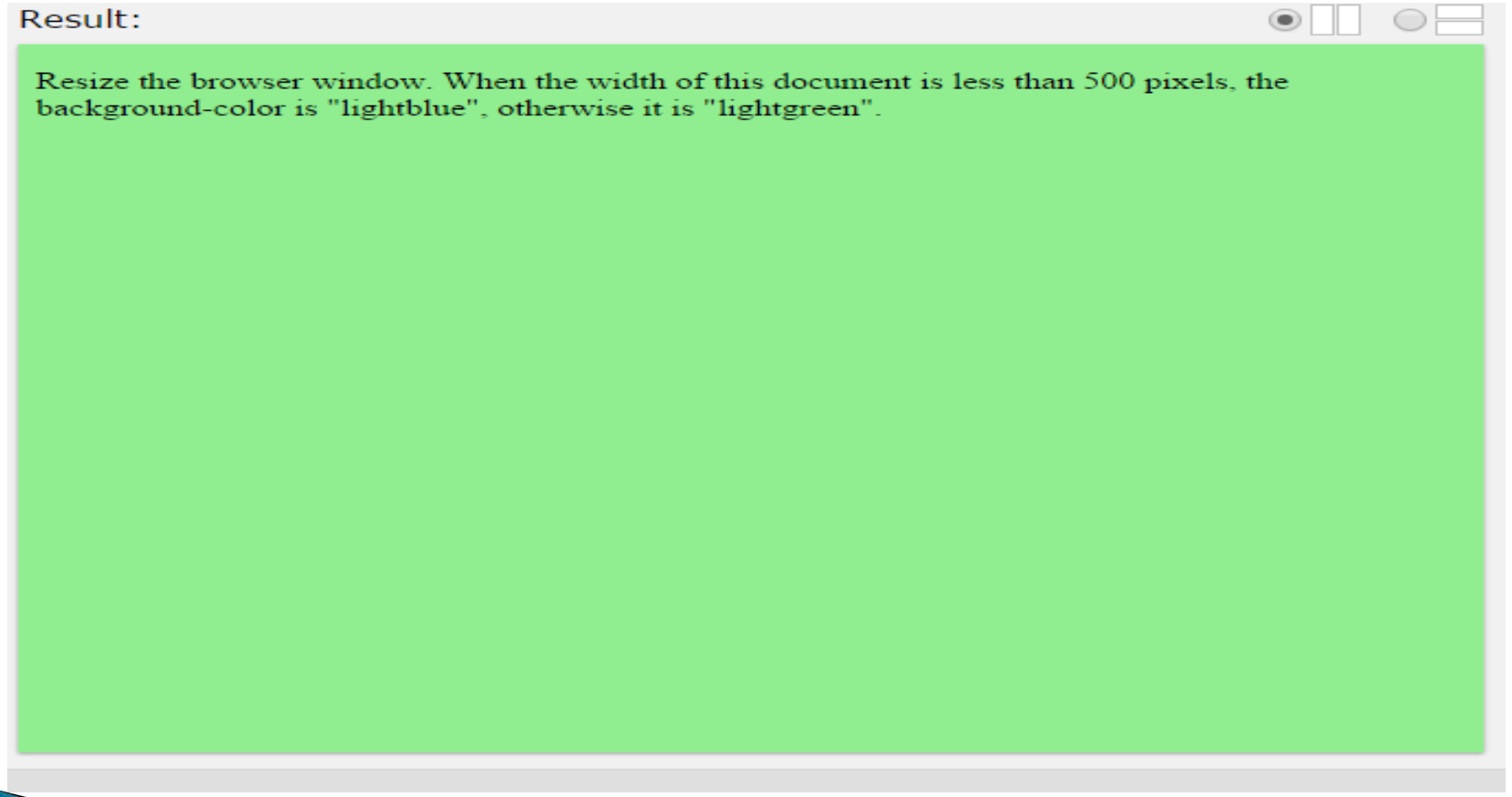
Media Queries Examples

- **Example 1**: If the browser window is smaller than 500px, the background color will change to light blue:

- @media only screen and (max-width:500px)

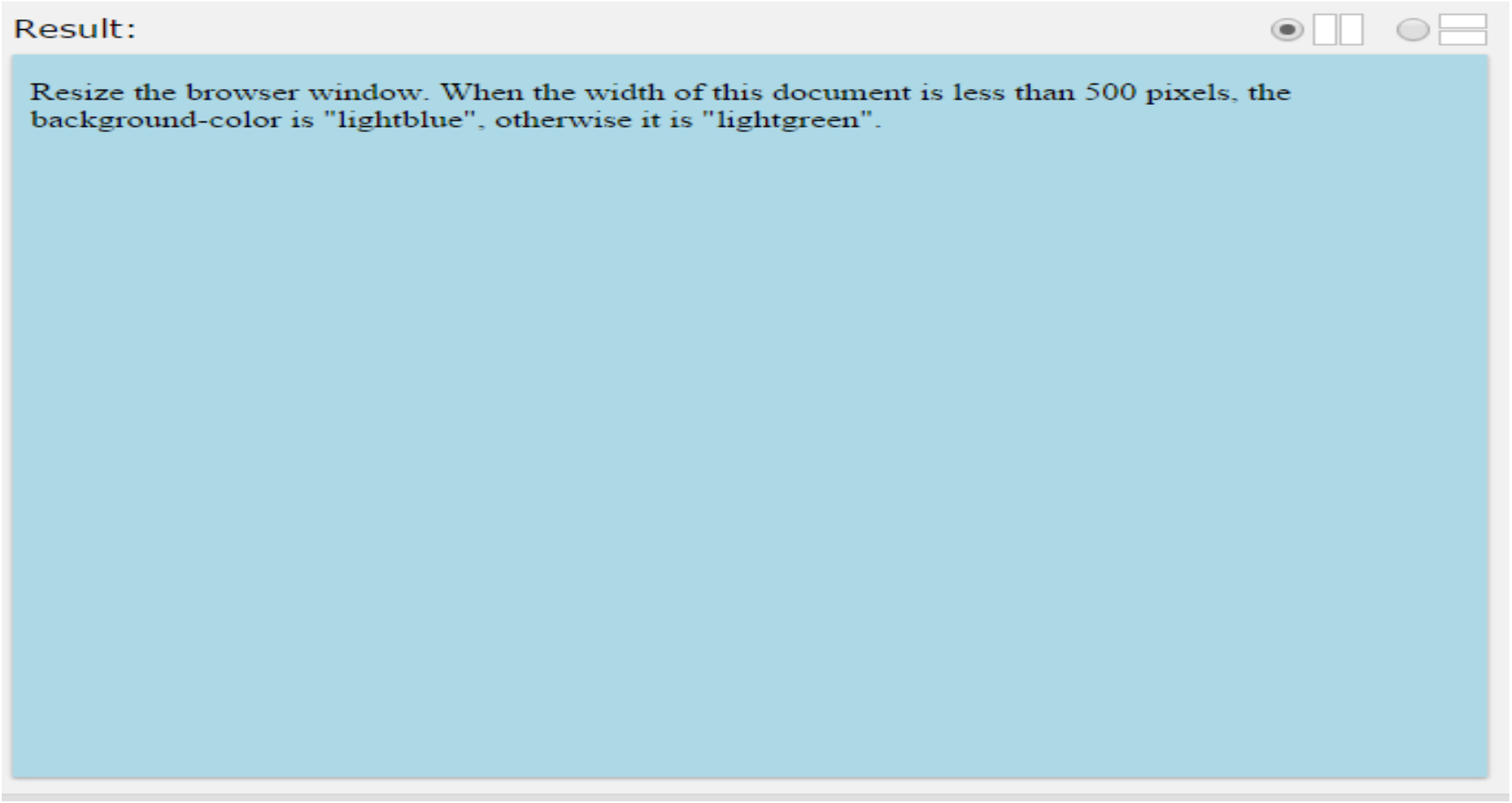
```
{
    body {
        background-color: lightblue;
    }
}
```

Media Queries Examples (Example 1)



Media Queries Examples (Example 1)

Result:

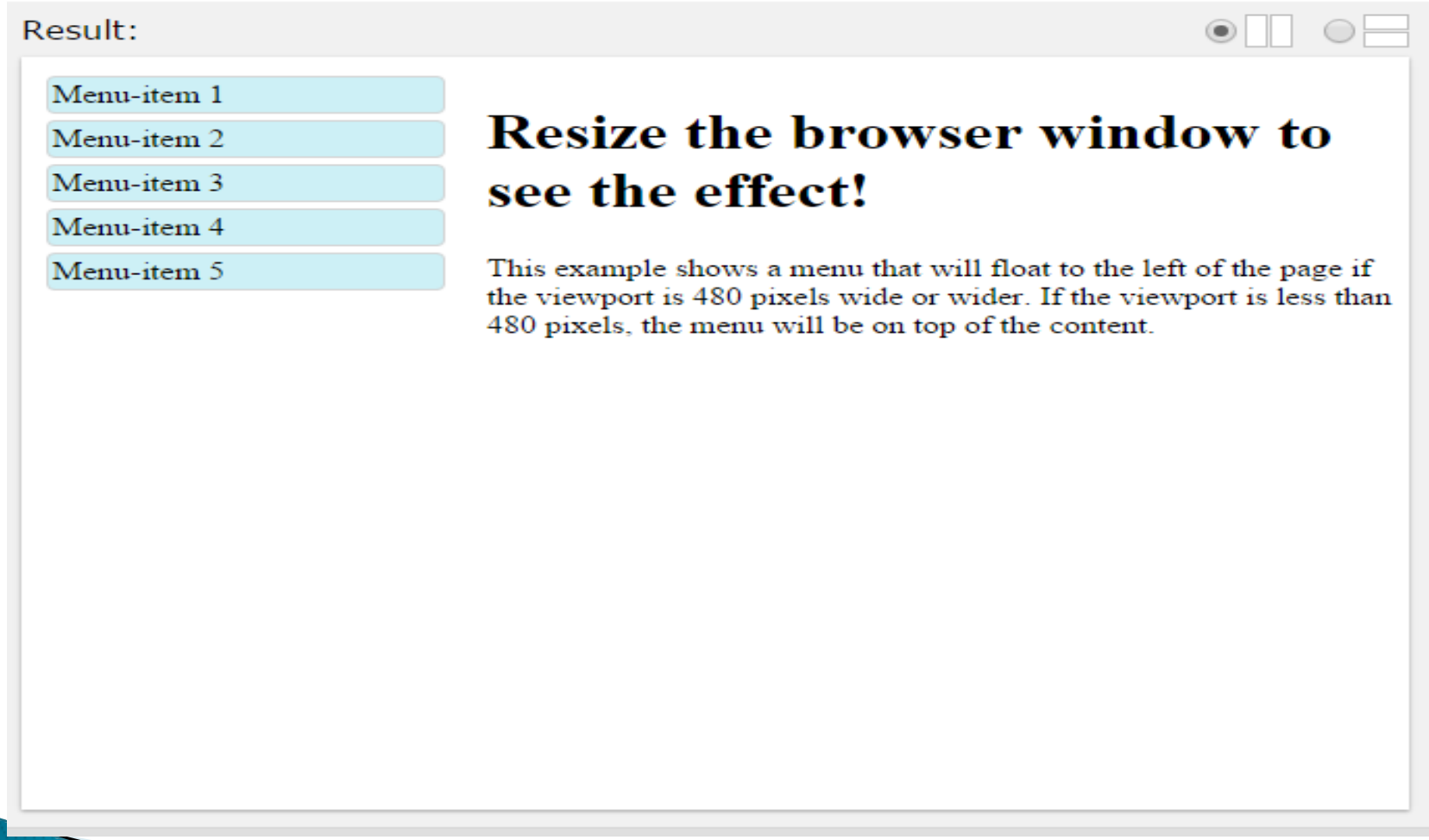


Media Queries Examples 2

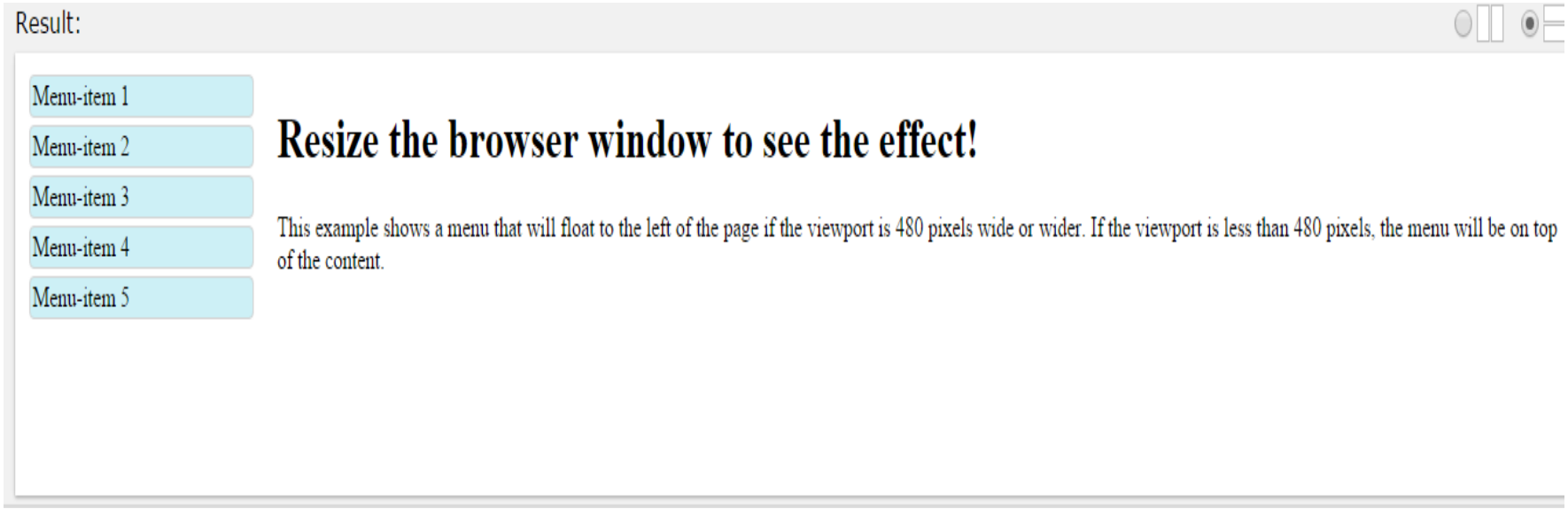
- **Example 2:**The following example shows a menu that will float to the left of the page if the viewport is 480 pixels wide or wider (if the viewport is less than 480 pixels, the menu will be on top of the content):

```
@media screen and (min-width: 480px) {
    #leftsidebar {width: 200px; float: left;}
    #main {margin-left:216px;}
}
```

Media Queries Examples 2



Media Queries Examples 2



Available Media Query Expression

Width	Orientation	Color
Height	Aspect-ratio	Color-index
Device-width	Device-aspect-ratio	Monochrome
Device-height	Grid	Resolution

Flexbox

- Flexible boxes, or flexbox, is a new layout mode in CSS3.
- Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.
- For many applications, the flexible box model provides an improvement over the block model in that it does not use floats, nor do the flex container's margins collapse with the margins of its contents.

Flexbox (Example)

Flexbox consists of flex containers and flex items.

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
    display: -webkit-flex;
    display: flex;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}
```

Flexbox (Example continue)

```
.flex-item {
    background-color: cornflowerblue;
    width: 100px;
    height: 100px;
    margin: 10px;
}
</style>
</head>
<body>

<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>

</body>
</html>
```

Flexbox (Example out put)

Flexbox Properties

Property	Description
display	Specifies the type of box used for an HTML element
flex-direction	Specifies the direction of the flexible items inside a flex container
justify-content	Horizontally aligns the flex items when the items do not use all available space on the main-axis
align-items	Vertically aligns the flex items when the items do not use all available space on the cross-axis
flex-wrap	Specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line
align-content	Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
flex-flow	A shorthand propert for flex-direction and flex-wrap
order	Specifies the order of a flexible item relative to the rest of the flex items inside the same container

The Device Pixel Ratio

The device pixel ratio is the ratio between physical and logical pixels

What is Pixel

- A pixel is simply the individual point of color on a digital image. A pixel doesn't have a particular size. It is an abstract representation of a specific coordinate, like a point on a map.

- Pixel is used to describe the number of discrete points that can be captured by digital cameras, and because most cameras can actually recognize millions of points, the term Megapixel is used to shorten the number of zeros needed.

Difference Between Physical and Logical Pixel

- Physical pixel :

  The actual number of pixels that mobile or computer device support e.g. physical pixel size of iphone6 plus is 1080*1920 pixels.

- logical pixel:

  Device independent pixels are called logical pixel e.g. 1 Logical pixel may be equal to 2 or more physical pixels in digital image.

The Device Pixel Ratio

- The device pixel ratio is the ratio between physical pixels and logical pixels. For instance, the iPhone 4 and iPhone 4S report a device pixel ratio of 2, because the physical linear resolution is double the logical resolution.

  Physical resolution: 960 x 640

  Logical resolution: 480 x 320

- Other devices report different device pixel ratios, including non-integer ones. For example, the Nokia Lumia 1020 reports 1.6667, the Samsumg Galaxy S4 reports 3, and the Apple iPhone 6 Plus reports 2.46 (source: dpilove). But this does not change anything in principle, as you should never design for any one specific device.

The Device Pixel Ratio

}

800 pixels

Nexus 7 Table

1080 pixels

GalaxyS4

The Device Pixel Ratio

Nexus 7
1280 x 800
1.3x
601 x 906

iPhone 5
640 x 1136
2x
320 x 240

Chromebook Pixel
2560 x 1700
2x
1280 x 850

Nexus 4
768 x 1280
2x
384 x 592

Galaxy S IV
1080 x 1920
3x
360 x 640

Mobile Device Pixels

- Mobile devices not only have different numbers of actual pixels, but they fit very different numbers of pixels into each inch of the screen real estate

The Device Pixel Ratio

The Device Pixel Ratio

Nexus 7
1280 x 800
1.3x
601 x 906

iPhone 5
640 x 1136
2x
320 x 240

Chromebook Pixel
2560 x 1700
2x
1280 x 850

Nexus 4
768 x 1280
2x
384 x 592

Galaxy S IV
1080 x 1920
3x
360 x 640

Image Quality

- Web applications need to deliver good image quality across a broad range of devices, but they need to do so at a minimum cost.

Removing the Default Zoom

- If you have used to browsing the Web on a smartphone you'll notice how websites are scaled out to fully display within the screen.

- When you get into building a responsive mobile design, the auto-zoom can really mess up your layout elements. Specifically, images and navigation content may appear small or too large in your layout.

Removing the Default Zoom

- There is a special meta tag you can append into the document header which resets this in most Android and iPhone devices.

```
<meta name="viewport" content="width=device-width; initial-scale=1.0; maximum-scale=1.0; user-scalable=0;" />
```

- This is known as the viewport meta tag which sets up some custom variables within the content. Apple has a documentation page regarding a few other meta tags you should look into, although these are geared specifically toward websites on iOS. The initial-scale value is important as this defaults your website to a full 100% zoom.

Difference in image Quality

Images are important facet of practically every website. Mobile users may not be looking to stream videos, but photos are a whole different story. These are also the biggest problem when it comes to layouts breaking out of the box model.

Difference in image Quality

```
img { max-width: 100%; }
```

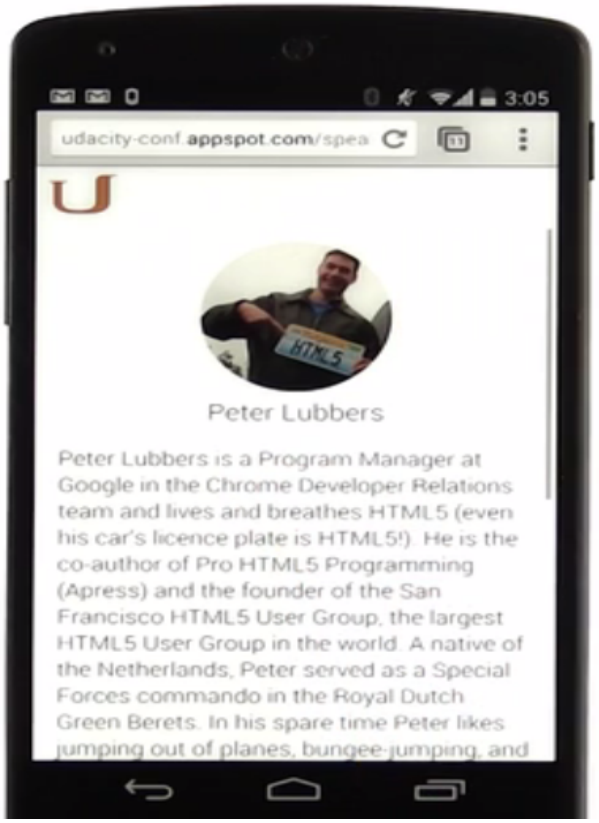
- The standard rule for CSS is to apply a max-width property to all images. Since they'll always be set at 100% you will never notice distortions. When the user re-sizes their browser window smaller than your image. it'll automatically re-adjust to 100% width scaled down. The problem is that Internet Explorer cannot understand this property, so you'll need to put together an IE-specific stylesheet using width: 100%;

Difference in image Quality

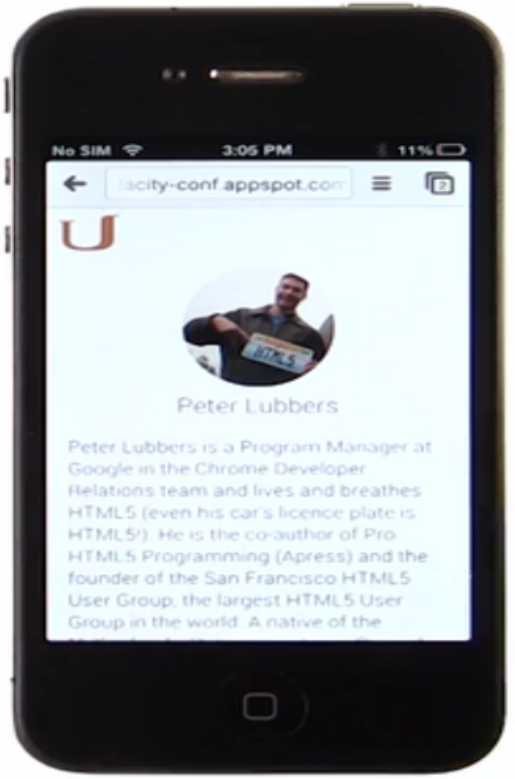
When we render images to the screen. A pixel layout dimensions may well not equal to pixel in real device terms.

Difference in image Quality

3x ~300px



2x ~200px



Beautiful images, low bandwidth

There should be our goal to make image as beautiful as they can using as little bandwidth as possible.

Beautiful images, low bandwidth

- Chromebook pixel screen: 4.1 mPx
- Nexus 4 screen : 0.9 mPx
- Iphone 5 screen :0.7 mPx

Producing Different Quality Content

Different Devices have different display densities. Some of them can make use of pixels and some of them can’t .

Producing different quality content

- What is SVG?

SVG stands for Scalable Vector Graphics.SVG is used to define graphics for the Web.

- The HTML <svg> Element

The HTML <svg> element (introduced in HTML5) is a container for SVG graphics.

SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

Producing different quality content

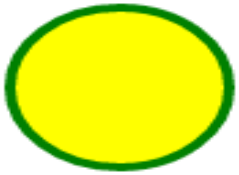
```
<!DOCTYPE html>
<html>
<body>

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stro
ke-width="4" fill="yellow" />
</svg>

</body>
</html>
```

Producing different quality content

Result:



Show all downloads...

Producing different quality content

- What is HTML Canvas?

The HTML <canvas> element is used to draw graphics, on the fly, via scripting (usually JavaScript).

The <canvas> element is only a container for graphics. You must use a script to actually draw the graphics.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

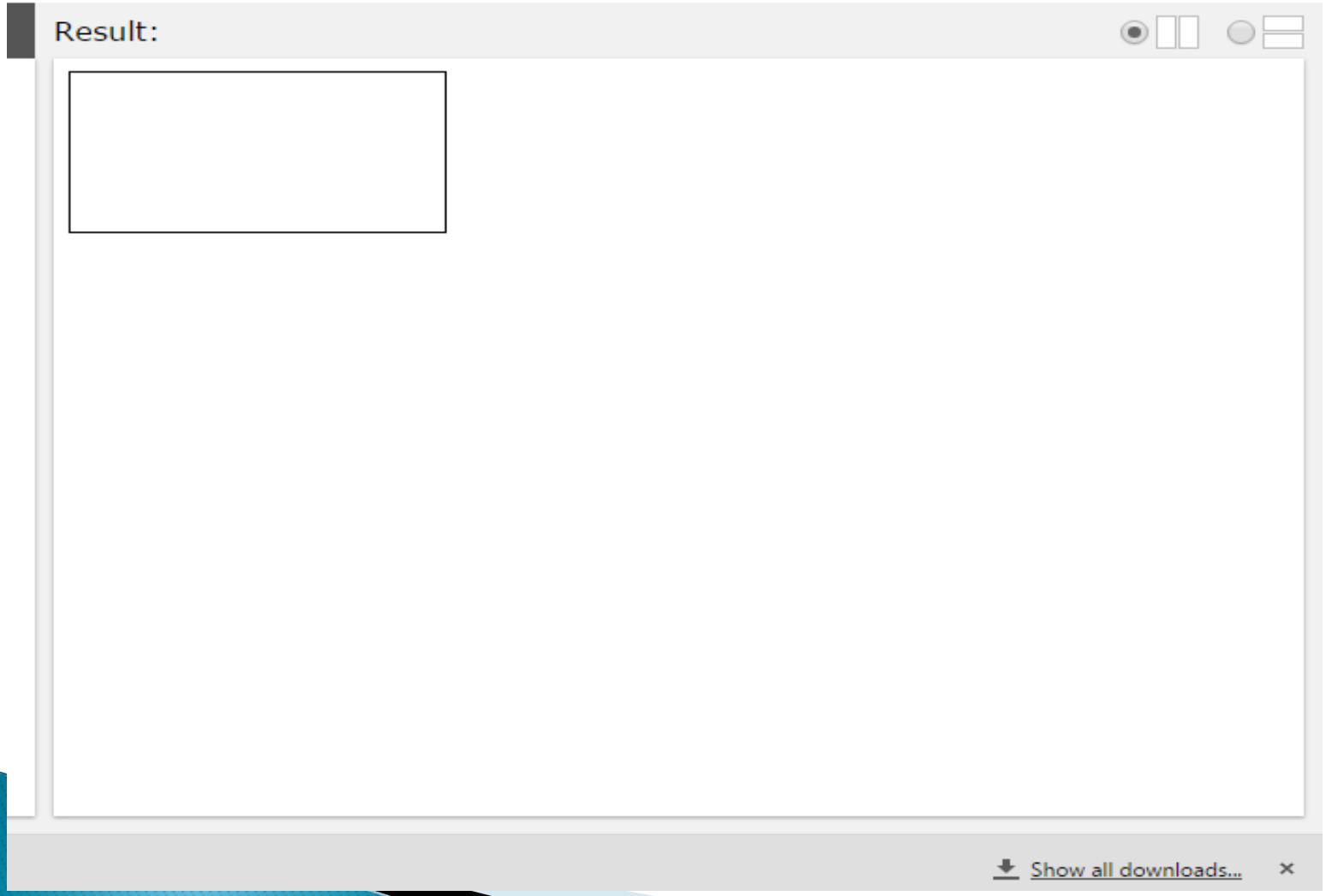
Producing different quality content

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #000000;">
Your browser does not support the HTML5 canvas tag.
</canvas>

</body>
</html>
```

Producing different quality content



Comparison of SVG and Canvas

Canvas	SVG
•Resolution dependent •No support for event handlers •Poor text rendering capabilities •You can save the resulting image as .png or .jpg •Well suited for graphic-intensive games	•Resolution independent •Support for event handlers •Best suited for applications with large rendering areas (Google Maps) •Slow rendering if complex (anything that uses the DOM a lot will be slow) •Not suited for game applications

Large area,low quality images

Let’s presume, we really need a raster image because some images can’t be represented well by vector drawing.

Large area, low quality images

Large area,low quality images

Deliver high resolution but highly compressed images

Do the right selection of quality image

} Using a higher resolution image to every device is not proper solution.

- Should we use a low quality source image ,replace it with JavaScript?
- Should we expect the server to give us the proper pixel density image magically?
- Should we make several copies of the image and conditionally request the right one from the client?

Do the right selection of quality image

- Should we use a low quality source image ,replace it with JavaScript?

Do the right selection of quality image

- Should we expect the server to give us the proper pixel density image magically?

HTTP Client Hints

This specification defines a set of HTTP request header fields, known as Client Hints, that are intended to be used as input to proactive content negotiation; just as the accept header allows clients to indicate what formats they prefer, Client Hints allow clients to indicate a list of device and agent specific preferences.

HTTP Client Hints

- **Available hints**

Current list includes DPR (device pixel ratio), Width (resource width), Viewport-Width (layout viewport width), and Downlink(maximum downlink speed) request headers, and Content-DPR response header that is used to confirm the DPR of selected image resources.

HTTP Client Hints

- **Delivering DPR-aware images**

  DPR hint automates device-pixel-ratio-based selection and enables delivery of optimal image variant without any changes in markup. For example, given the following HTML markup:

  <img src="img.jpg" alt="I'm a DPR-aware image!">

HTTP Client Hints

GET /img.jpg HTTP/1.1
User-Agent: Awesome Browser Accept: image/webp, image/jpg
DPR: 2.0
CH-RW: 160

HTTP/1.1 200 OK
Server: Awesome Server
Content-Type: image/jpg
Content-Length: 124523
Vary: CHoDPR, CH-RW
DPR: 2.0

Do the right selection of quality image

- Should we make several copies of the image and conditionally request the right one from the client?

Responsive Images

Responsive images is a method for providing the browser with multiple image sources depending on display density, size of the image element in the page, or any number of other factors to make possible flexible approach to images.

Responsive Images

Graphics are often a problem in responsive layouts designed to adapt automatically to different device sizes from desktop monitors to hand-held smartphones.

Traditionally, graphics have fixed dimensions that break responsive layouts on screens that are smaller than the graphic.

Length is not a problem, because pages can always scroll vertically. However, you never want the page to have to scroll horizontally on mobile devices with fluid layouts.

Problems in Responsive Images

- **What Are The Problems?**

One major factor in the need for responsive images is overall website size - even today, a huge percentage of mobile-version sites are as large (or even larger) than their desktop versions, thereby affecting performance negatively

Problems in Responsive Images

- **Problem 1 - Semantics**

Making responsive images work cleanly and reliably across multiple platforms sometimes involves using techniques that aren't ‘semantic'. Why? Because when the source of an image points to a real image, with an alt text to describe it, even if the source is the smallest image possible, it means you're downloading unnecessary data.

Problems in Responsive Images

- **Problem 2 - Art Direction**

A picture tells a thousand words ... most of the time. Many responsive image techniques enable you to provide several resolution versions of an image, which can then be used accordingly for suit a given platform. However, this can sometimes have a negative effect where the message of the image is diluted or lost altogether. What's the minimum size for an image? The answer, of course, depends on context.

Problems in Responsive Images

- **Problem 3 - Validity**

If you are customizing markup, checking markup validity using a validation service such as W3C ensures that a construct follows the correct syntax of whatever language it works with (for example, HTML). While most of the time browsers are able to think ahead to present Web pages that aren't strictly ‘valid' without any problems, these results can sometimes vary between one browser and another. In rare cases, it can cause confused layouts or even crashes. ‘Validating' every page is a bulletproof way of identifying these problems before they occur.

Problems in Responsive Images

Problem 4 - Bandwidth Capabilities

Much as we'd all like to say otherwise, a world of super-speed connections and lighting-fast load times for everyone is still some way off. That's why in many developers' opinions, responsive images ought to consider the bandwidth available on any given device, and adjust the image size accordingly to cut out excess downloads.

Problems in Responsive Images

- **Problem 5 - Server Side Components**

  The bulk of Adaptive Images' work is done through .htaccess and PHP 5.x, which offers a great solution without complete dependency on JavaScript. The problem with .htaccess is that it assumes your website is running on an Apache server. If you're using something else, such as Nginx, you can port the .htaccess over to Nginx syntax - which is similar to Apache's but this still involves some work. Additionally, adding PHP files to your site's root directory may not be possible if you're running on other technologies, such as Ruby or Python. Clearly this isn't ideal for everyone, so be sure to check first before you dive in.

Problems in Responsive Images

- **Problem 6 - JavaScript**

As you've probably noticed, many of the responsive image techniques above require some JavaScript. If you'd prefer to keep things as simple as possible and want to avoid using JavaScript you can use the third-party Sencha.io solution.

Responsive images

Responsive images is a method for providing the browser with multiple image sources depending on display density, size of the image element in the page, or any number of other factors to make possible flexible approach to images.

Responsive Images using CSS3

Using The width Property:

If the width property is set to 100%, the image will be responsive and scale up and down:

```
img {
    width: 100%;
    height: auto;
}
```

Responsive Images using CSS3

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
img {
    width: 100%;
    height: auto;
}
</style>
</head>
<body>
<img src="img_chania.jpg" width="460" height="345">
<p>Resize the browser window to see how the image will scale.</p>
</body>
</html>
```

Responsive Images using CSS3

Resize the browser window to see how the image will scale.

Responsive Images using CSS3

- **Using The max-width Property**

If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

```
img {

    max-width: 100%;
    height: auto;

}
```

Responsive Images using CSS3

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
img {
    max-width: 100%;
    height: auto;
}
</style>
</head>
<body>
<img src="img_chania.jpg" width="460" height="345">
<p>Resize the browser window to see how the image wil scale when the width is less than 460px.</p>
</body>
</html>
```

Responsive Images using CSS3

Resize the browser window to see how the image wil scale when the width is less than 460px.

Use of high quality and different sized copies of image

In order to do responsive images , we need to request different images based on the device pixel ratio, but also possibly based on the layout width.

These images may be the same image transcoded to different sizes.

They may be actually different images depending on the layout size.

Use of high quality and different sized copies of image

Use of high quality and different sized copies of image

We shall need to provide multiple copies of this image so that low pixel count devices can save bandwidth or high pixel count devices can get a crisp image.

It is possible to manage on server side in some cases.

In fact, some content systems let you simply specify what size you want a given source image app.

Use of high quality and different sized copies of image

If you want the images to actually represent different art, different crops for example you'll have to do that manually.

For each of these images though you should remember to tune the quality parameters to minimize file size while maximizing quality.

We'll need at least 1x and 2x versions of images as 1x or 2x for device pixel ratio purpose.

For responsive images, We may request different images based on available layout width.

Request Right Image

- Client should request for the “right” image.
- We need to get the client to ask for the right image to be downloaded.

Request Right Image

- It needs to know what device pixel ratio the device is using and potentially the layout size too, if it's responsive to the layout width and then request only the appropriate image.

Request Right Image

- This last part is the hard part, because the web browser tries to download image as early as possible which of course is a good thing for getting content on the screen quickly, but bad if the client asks for more than one version particularly on a mobile device with limited bandwidth.

Request Right Image

So we should try to find the solution for the problem of right image at the right time.

Ideal Image Solution

- What we really need is the ability for our client to selectively choose what image to use.

Ideal Image Solution

- Ideal solution would be semantic and validate properly.It would only load one copy of the image and work and be accessible across all browsers.

Ideal Image Solution

- Unfortunately that solution does not exist.

Background Image

The CSS background properties are used to define the background effects for elements.

We shall discuss following properties of background images:

- Background Image - Repeat Horizontally or Vertically
- Background Image - Set position and no-repeat
- Background Image -background-position:

Background Image

- Background Image:

The background-image property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element.

Background Image

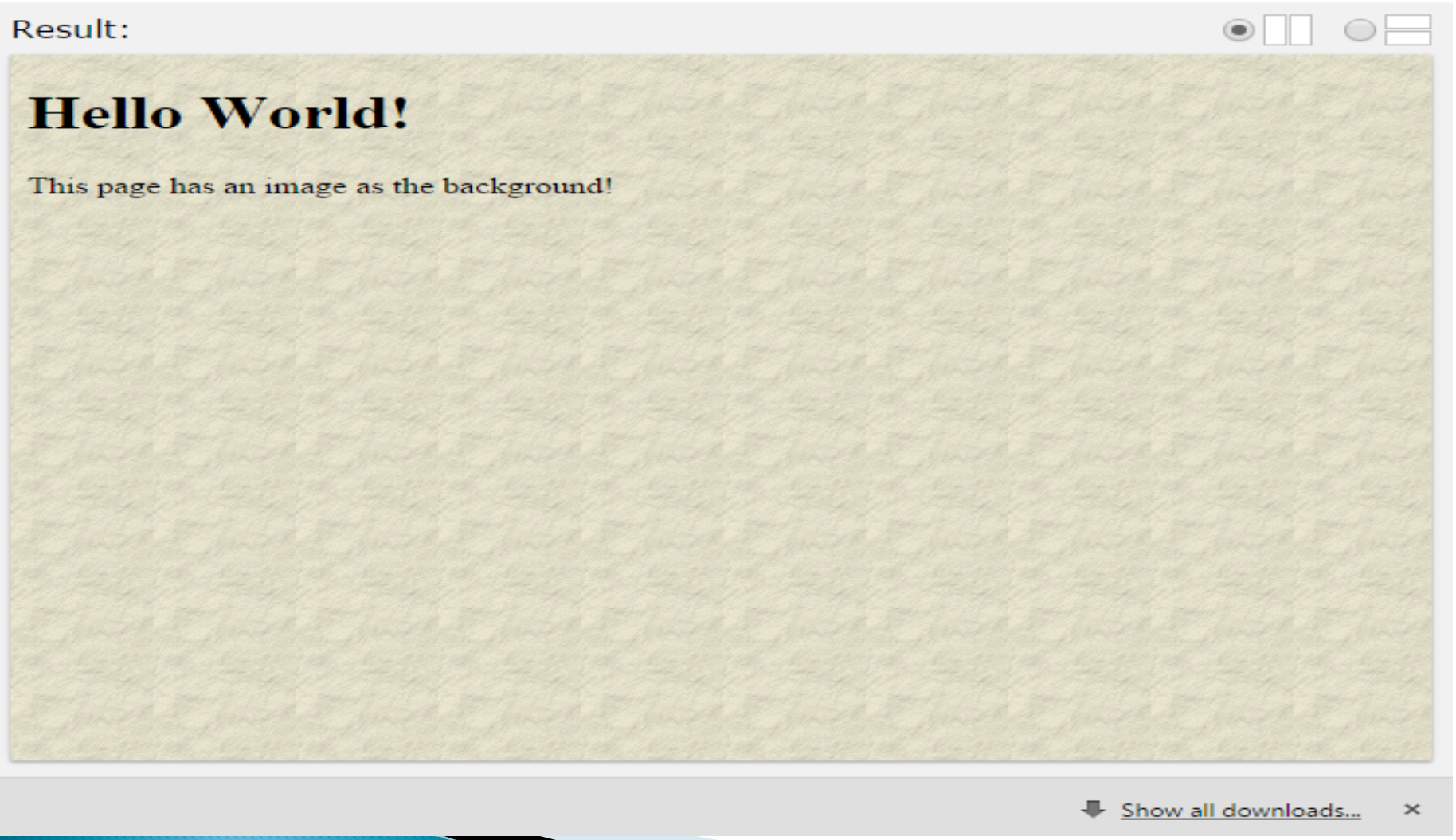
Example:

```
<!DOCTYPE html>
<html>
<head>
<style>

body    {
          background-image: url ("paper.gif");
        }
</style>
</head>
<body>

<h1>Hello World!</h1>
<p>This page has an image as the background!</p>
</body>
</html>
```

Background Image



Background Image

- Background Image – Repeat Horizontally or Vertically

  By default, the background-image property repeats an image both horizontally and vertically.Some images should be repeated only horizontally or vertically, or they will look strange, like this:

Background Image

- **Background Image - Repeat Horizontally or Vertically:**

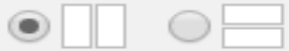
```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("gradient_bg.png");
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<p>Strange background image...</p>

</body>
</html>
```

Background Image

Result:



Hello World!

Strange background image...

Background Image

- **Background Image - Set position and no-repeat:**

Background Image

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body
{
    background-image: url("img_tree.png");
    background-repeat: no-repeat;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<p>Background image example.</p>
<p>The background image is only showing once, but it is disturbing the reader!</p>
</body>
</html>
```

Background Image

Result:

Background Image

background-position:

In the example above, the background image is shown in the same place as the text. We want to change the position of the image, so that it does not disturb the text too much.

The position of the image is specified by the background-position property:

Example

```
body {
    background-image: url("img_tree.png");
    background-repeat: no-repeat;
    background-position: right top;
}
```

Background Image

Result:

Hello World!

Background no-repeat, set position example.

Now the background image is only shown once, and positioned away from the text.

In this example we have also added a margin on the right side, so the background image will never disturb the text.

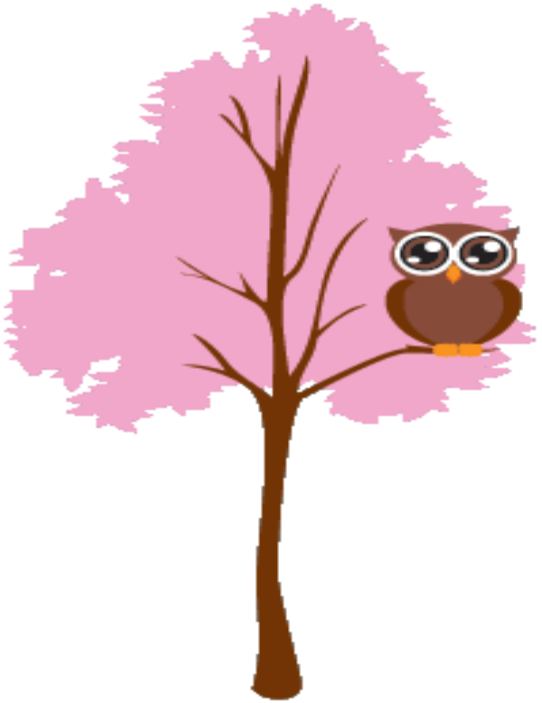


Image-set

Why we use image-set instead of media queries?

I. The problem with media queries for higher resolution displays is they don't give the browser any discretion. They say explicitly that if the pixel density equals 1 or 2 or whatever, that the browser must use a particular image source.

II. The various resolutions of the image are all in the same place in css. When we use media queries to specify different image resolutions, we can end up with the image sources separated by dozens of lines of css which makes tracking down the different image sources more challenging.

Image-set

- The emerging trend on HD screens for computer devices has changed the way we build websites, including how we add images. In the past, we can simply use background-image and set the image URL.

- Today, this is no longer relevant, since we also need to provide an image that is optimized for higher resolution displays or the image will look pixelated or blurred. So, technically we need two sets of image, one in regular size and another twice the size.

Image-set

Image-set

- **CSS Image Set Function:**

  By using this function, we can insert multiple images which will be set for normal and high-res display.

  Apart from that, this function is also trying to deliver the most appropriate image resolution based on the connection speed. So, regardless of the screen resolution, if the user accesses the image through a slow Internet connection.

  CSS Image Set is experimental. It is only supported in Safari 6 and Google Chrome 21 and is prefixed — -webkit-image-set().ection, the smaller-sized image will be delivered.

Image-set

- This function is declared within the background-image property, while the background URL is added within the function followed by the resolution parameter (1x for normal display and 2x is for high-res display), like so.

Image-set

```
.selector {
    background-image: -webkit-image-set(url('../img/image-1x.jpg') 1x, url('../img/image-2x.jpg') 2x);
}
```

Image-set

```
body {
	background-image: url('../img/bg.png');
}
.demo-wrapper {
	width: 600px;
	margin: 0 auto;
}
.demo-wrapper .theimage {
	margin-top: 100px;
	border: solid 8px #f3f3f3;
	width: 100%;
	height: 360px;
	background-size: 100% auto;
	background-repeat: no-repeat;
	background-position: center center;
	background-image: url('../img/image-1x.jpg');
	background-image: -webkit-image-set(url('../img/image-1x.jpg') 1x, url('../img/image-2x.jpg') 2x);
}
```

Image-set

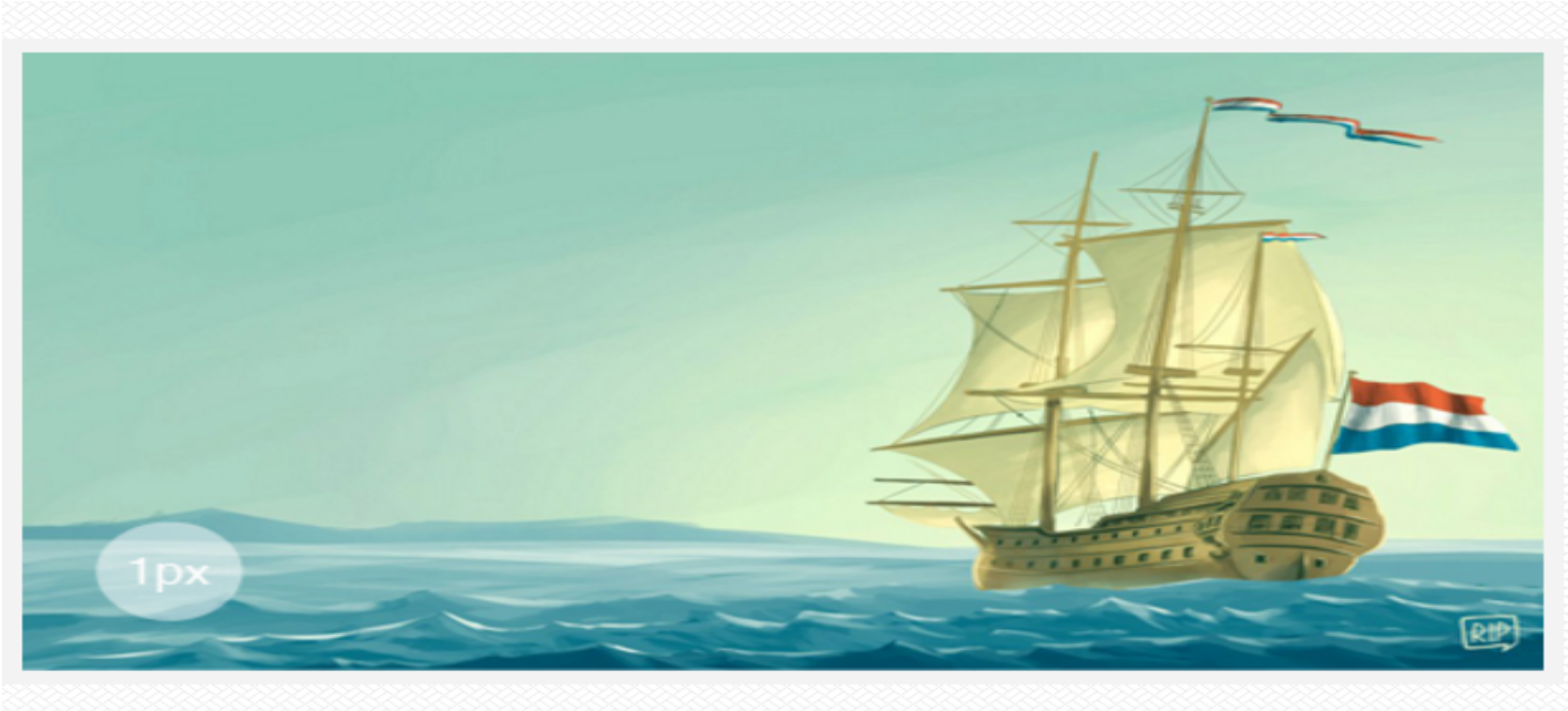


Image-set

- **Finally:**

image-set() function seems to be a promising solution for serving images in high-res. But, with the current limited browser support, image-set() is not ready to be implemented in a live site. In addition, the implementation could be changed along the way in the near future.