Optimizing sites for

MOBILE DEVICES

james williamson | lynda.com

D²W

designer/developer
workflow conference

Howdy again y'all

james williamson

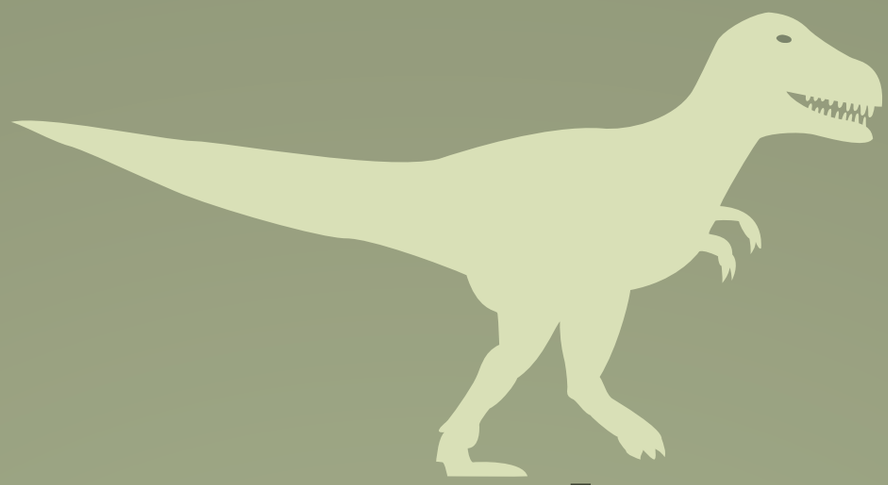
lynda.com | senior author

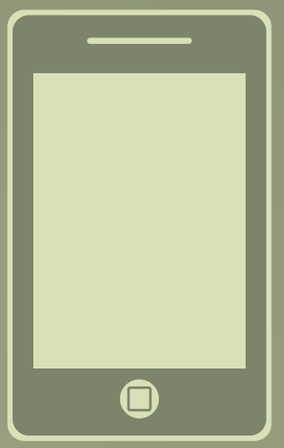
@jameswillweb on the Twitter

Today I want to talk about techniques you can use to plan and develop sites so they perform well on mobile devices.

(it’s kind of a big deal)

We are now in the age of mobile





500 million

Cambrian

180 million

Jurassic

present

Cupertino

Or are we?

It might be more accurate to call it
“The Age of Constant Connectivity”

This means that over time we need to change the way we approach design.

We need a fundamental shift in content planning and site architecture to help us design always-connected user experiences

...but that's an entirely different presentation

(maybe next year)

However, it's now critical for you to have a strategy in place for how your site looks and performs on mobile devices.

So what does that entail?

Changing the way we plan sites to include mobile

Using responsive design techniques that allow flexible layouts

Managing resources responsibly

Taking advantage of device capabilities

Planning for mobile

Mobile needs to be an equal partner, if not the primary platform you design towards

Check out Mobile First*

Popularized by Luke Wroblewski, it emphasizes starting the design process with mobile in mind.

*http://www.lukew.com/presos/preso.asp?26

Remember contexts when designing

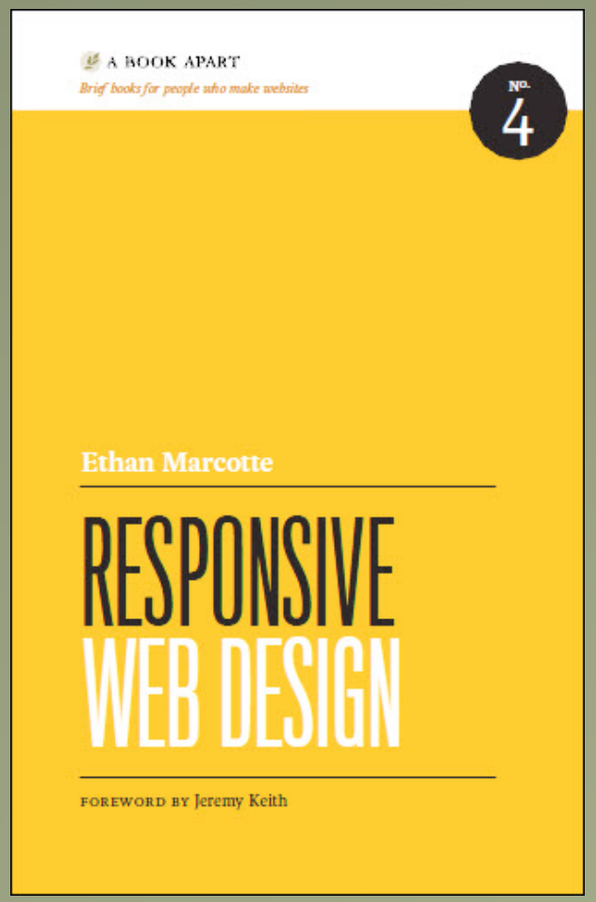
Mobile devices are always on and always there

Mobile users are not distracted, not on-the-go, and not disengaged

Don’t block access because someone’s on a mobile device

Mobile capabilities increase what you can do, not limit you!

Use responsive design techniques



Fluid grids

Media Queries

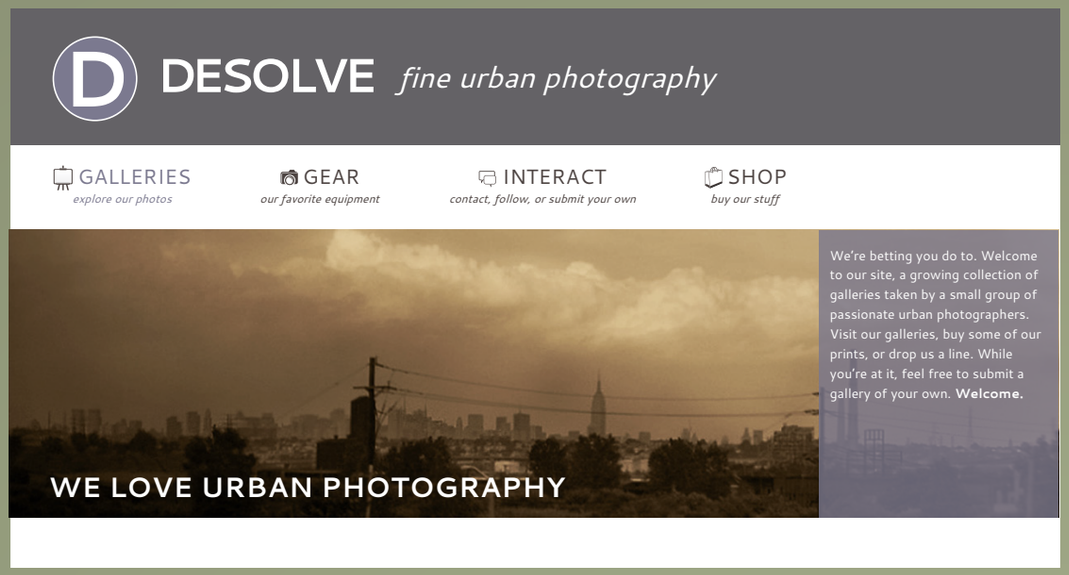
Responsive Images

Responsive Web Design by Ethan Marcotte

http://www.abookapart.com/products/responsive-web-design

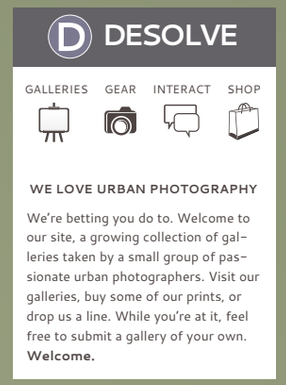
Use fluid layouts with breakpoints

desktop



tablet

mobile



Don't lock breakpoints into fixed sizes (960px, 720px, 320px), keep layouts fluid within your breakpoints to account for multiple devices and screen sizes

Calculating percentage-based element widths

For each breakpoint, determine an “ideal” target size
(For example 960px for desktop)

Determine percentages based off this target size

Margin and padding values are calculated as percentages of the containing block (the parent element)

It's not always going to be this hard

Flexbox is coming! (http://www.w3.org/TR/css3-flexbox/)

Other layout modules are being designed with mobile in mind
Grid template, CSS Regions, Multi-column Layout, etc.
(check out www.w3.org/Style/CSS/current-work.en.html for more info)

Using Media Queries

CSS Media Queries allow you to control the application of styles based on the presence or absence of specific media features

```
@media only screen and (max-width:480px) { ... }
```

like viewport width!

Media Query syntax

```
link rel="stylesheet" href="desktop.css"
media="[not | only ] screen [and] (expression)"
```

or

```
@media [not | only ] screen [and] (expression) {
}
```

The keywords “not”, “only”, and “and” can be used to filter results, while expressions can be used to check for specific media features

Media Query syntax

Media Features

`width*`	`color*`
`height*`	`color-index*`
`device-width*`	`monochrome*`
`device-height*`	`resolution*`
`orientation`	`scan`
`aspect-ratio*`	`grid`
`device-aspect-ratio*`	

*accepts “min-” and “max-” prefixes

Meta Viewport tags

Meta viewport tags can be combined with media queries to help ensure consistent experiences.

CSS @viewport may eventually replace

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
```

for example, this meta viewport sets the viewport to the device width and sets the initial zoom to 100%.

Strategies for using media queries

Use inline syntax to reduce http requests

Use width instead of device-width to target viewports

Combine them with global styles for more maintainable code

Responsive Images

Yeah, that.

Responsive Image challenges

Desktop and mobile layouts usually require different image sizes

Larger images add unnecessary overhead for mobile devices

Screen density differences now allow the use of higher resolution images for some screens

Responsive Image techniques

Use Background Images

Through media queries, you can change image requests based on layout (Android 2.3 > still downloads ALL background-images)

Larger decorative background images will clip if inside a fluid parent element

Since no img element is used, images are non-semantic

Responsive Image techniques

Make images fluid

By setting width or max-width to percentage values, you can make images scale along with layout

Still forces mobile devices to download large image sizes

Image scaling can harm performance on mobile devices

Responsive Image techniques

Use javascript to replace images

Use javascript to determine which image is appropriate and then serve that image

Multiple libraries have been created to enable this

Can be tricky to ensure that only the image you want is downloaded

If the technique requires you to remove the src attribute, your code become non-semantic

WHATWG to the rescue?!

Recently, the WHATWG added the srcset attribute to the img tag

This allows you to pass a comma separated list of multiple images and the conditions (width, height, resolution) that trigger their use

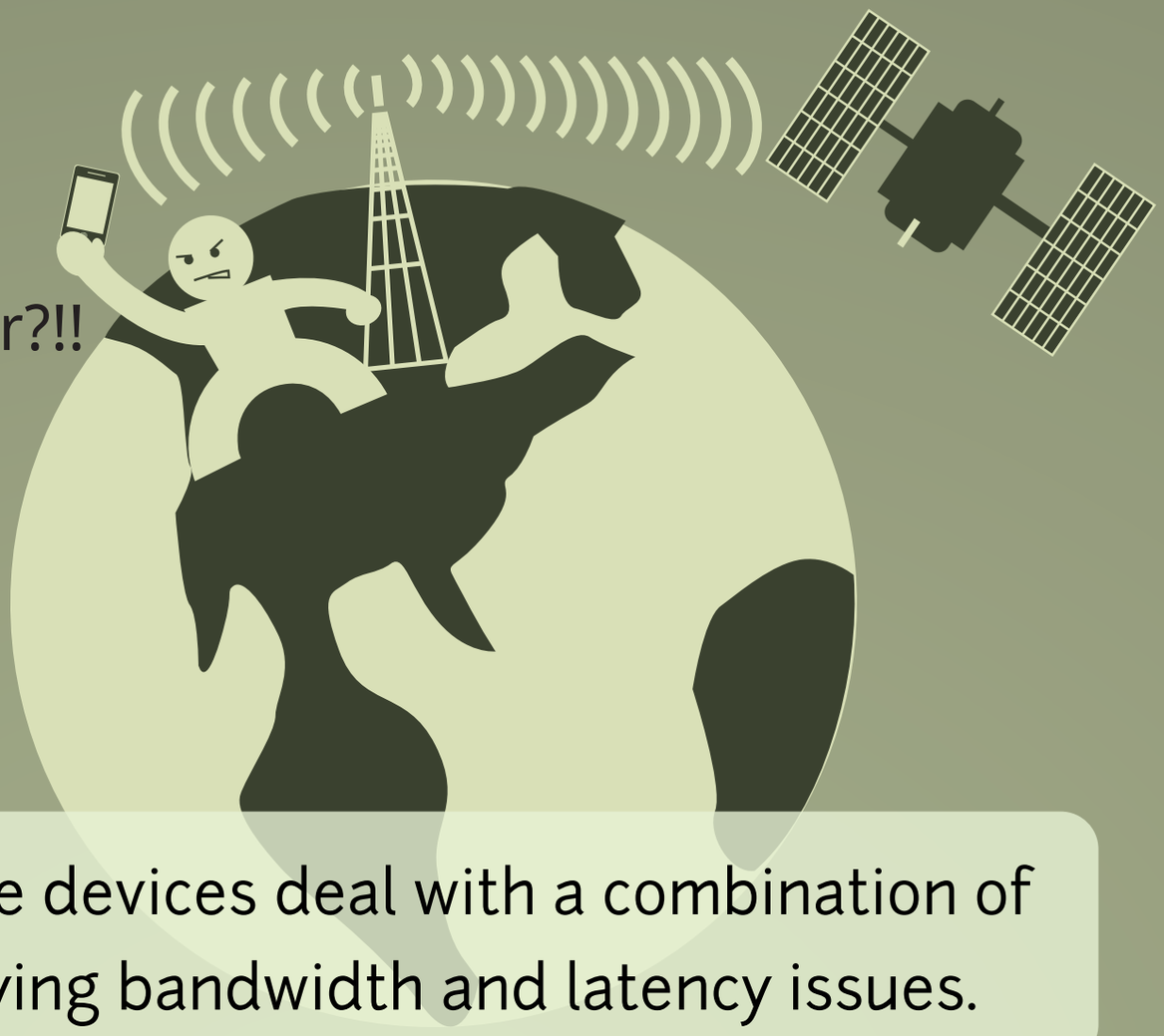
Current unsupported

Ignored the recommendation of the Working Group that proposed the picture element

Is a big mess

Managing resources

Why the hell
isn’t this faster?!!



Mobile devices deal with a combination of varying bandwidth and latency issues.

You can ease your users pain by managing resources wisely.

Fewer server requests are better

Condense multiple CSS and javascript files into single files when possible

Minify scripts and files and use gzip compression to reduce file size

Use CSS Sprites to limit image requests

Use data-URLs in place of images when feasible

In many instances CSS can be used in place of images

Only load resources if you need them

Often, external scripts won't be used on pages or on mobile devices at all

Use asynchronous loading and detection scripts to serve only what each context needs

There are multiple options available, but Modernizr 2.0 combines feature detection, media query testing, and yepnope.js to give you a robust set of options (http://modernizr.com)

Take advantage of mobile capabilities

Mobile device capabilities

Geolocation

Orientation / Accelerometers

SMS

Touch / Gesture support

Audio & Video capture

OMG! THEY CAN MAKE PHONE CALLS (some of them)

Thinking about these capabilities during the planning process can result in more fulfilling user experiences

HTML5 and standards can help

APIs for Geolocation, Touch Events, and SMS can help make it easy to add these capabilities to your site

HTML5 form elements are well supported in mobile browsers

HTML5 audio/video formats are widely supported

Testing...testing...testing

Always test your site on as many browsers and devices as you can

Emulators are good, but actual devices are better

Pay as close attention to loading times and server requests as you do page layout or functionality

Testing...testing...testing

Web Inspector Remote (weinre) | remote debugging
http://people.apache.org/~pmuellr/weinre/

Webkit Remote Debugging
http://www.webkit.org/blog/1620/webkit-remote-debugging/

Blaze mobile testing | Mobile performance testing
http://www.blaze.io/mobile

A few things to remember....

Make mobile an equal partner when planning and designing your sites.

Use standards-compliant, clean, accessible code as the foundation for all your sites.

Create responsive designs that work across multiple screens

Mange resources carefully and limit server requests

Take advantage of device capabilities to enhance user experiences

THANK YOU

james williamson | lynda.com
jwilliamson@lynda.com
@jameswillweb on the Twitter
www.simpleprimate.com