Modern Web Development

mozilla

Today we're going to talk a bit about Modern Web Development which is already an arrogant thing to say. What was modern development in 1998 is laughable these days and no doubt we will look at what we do with contempt in a few years. So instead of showing you shiny demos and tools of the trade of today let's quickly talk about approaching development in a way that will keep you safe from making mistakes today **and** tomorrow.

A new world...

- HTML5 != Desktop
- Exciting new opportunities
- New browsers with one goal
- Taking this show on the road

mozilla

We live in a new world when it comes to web development. We build for lots more environments than the desktop, mobile devices and desktop browsers offer us much more functionality than we had before, browsers are agreeing on a standard to support instead of one-upping another with proprietary technology and our web technologies are used in all kind of cool new and mobile devices. This rocks and means we are on to something good here.

Demo time:
HTML5 magic built-in

Show the form.html and magic.html demo to show that browsers come with great in-built functionality and that they are very forgiving which means we don't break the web of old. Refer to the README how to present them.

Memo time:

- **HTML5 is backwards compatible and robust by design**
- **That doesn't mean we should rely on that - it just means we can move forward safely**
- **HTML5 has a lot of in-built client side functionality we do not need to write in JavaScript ourselves.**

mozilla

You can find more instructions how we reached these conclusions in the README of the code examples.

Developing for the unknown

http://www.flickr.com/photos/smemon/5547552978/

Web development, by definition, is developing for the unknown. You can not force users to use a certain browser or have a certain technology enabled and you can not expect any of that. So how can you still develop without creating broken experiences? How do you prepare for what is beyond the next bend?

Progressive enhancement

One way to approach the unknown is progressive enhancement. This means you take something that works and enhance it to be even better when a certain technology is available. The best example for this is escalators - they are progressively enhanced stairs. When there is no electricity or some other default people can simply use them as stairs - you don’t need to have some fallback way of going up and redirect people to that one - error handling is already build in.

Progressive Enhancement

- Start with something that works
- Then hijack it to become better
- Test for support, then add on
- No broken promises

Graceful degradation

The other way to deal with the unknown is Graceful degradation. Lifts work like that - they are great to reach another level fast and take up much less space than escalators. When they break down though, then you need to have stairs as a fallback somewhere in the building or people will be stuck. You actually need to both build a lift and stairs because of fire regulations. Graceful degradation can seem much easier and less work but will always mean that you need to create and maintain a fallback. It also leads to broken promises as you may show UI elements that don’t work - like these buttons which can’t be pressed.

Graceful degradation

- Start with a complex matter
- Develop a fallback for other environments
- Maintain both
- Broken promises

Demo time:
Progressive Enhancement vs. Graceful Degradation

mozilla

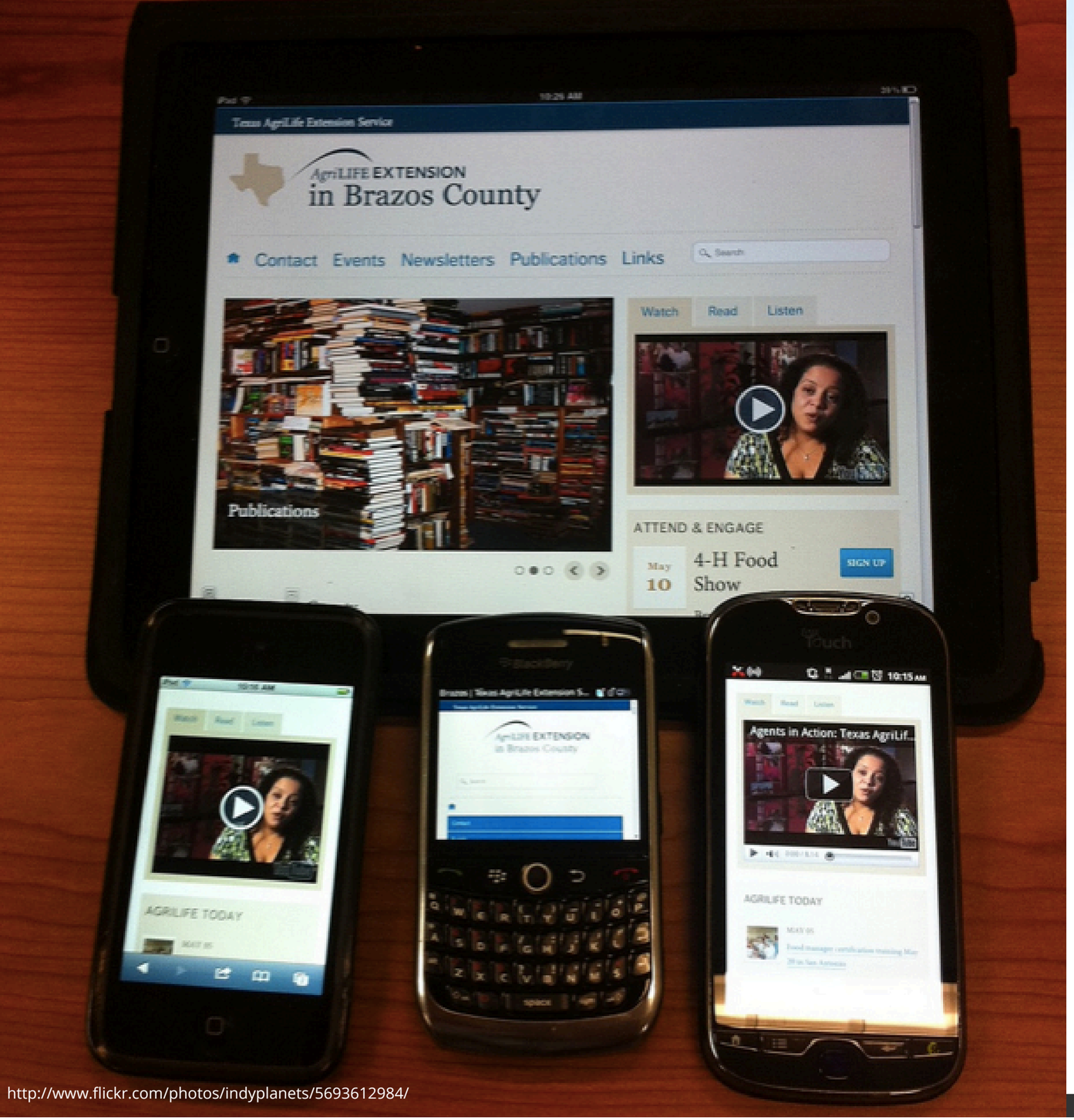
Show the gradients.html, /vintagecafe and the /contextmenu demos.
Refer to the README how to present them.

Memo time:

- **Be flexible and nothing will cause you grief in the future**
- **Don’t make end users suffer for your mistakes**
- **Understand the basics**
- **Don’t bother with PE when you are in a fixed environment**

mozilla

You can find more instructions how we reached these conclusions in the README of the code examples.



http://www.flickr.com/photos/indyplanets/5693612984/

Responsive Design

mozilla

Responsive design means that instead of building a special version of a web document or app for each device - desktop, mobile, tablet and so on you build one version of the product and use CSS to deliver different experiences to each of them depending on their resolution or orientation. It is a very “future friendly” approach to design. Instead of trying to predict all the use cases of your product or limiting yourself to a few ones you allow your design to breathe and change.

Mediaqueries

- Apply CSS for certain scenarios
- Also available in JavaScript
- One design, many views

Responsive design uses mediaqueries which is a CSS feature that allows you to define styles that only get applied when the displaying browser fulfils certain rules you define - like a certain resolution, orientation or pixel density. They are also available in JavaScript to apply functionality only when and if it makes sense.

Demo time:
Responsive design demos

mozilla

Show the responsive.html demo or some example on mediaqueri.es.
Refer to the README how to present them.

Memo time:

- Mediaqueries allow for granular design changes
- No need for JS sniffing
- Possible performance issues :(
- matchMedia() can help
- Queries for connection speed needed

mozilla

You can find more instructions how we reached these conclusions in the README of the code examples.

What about legacy browsers?

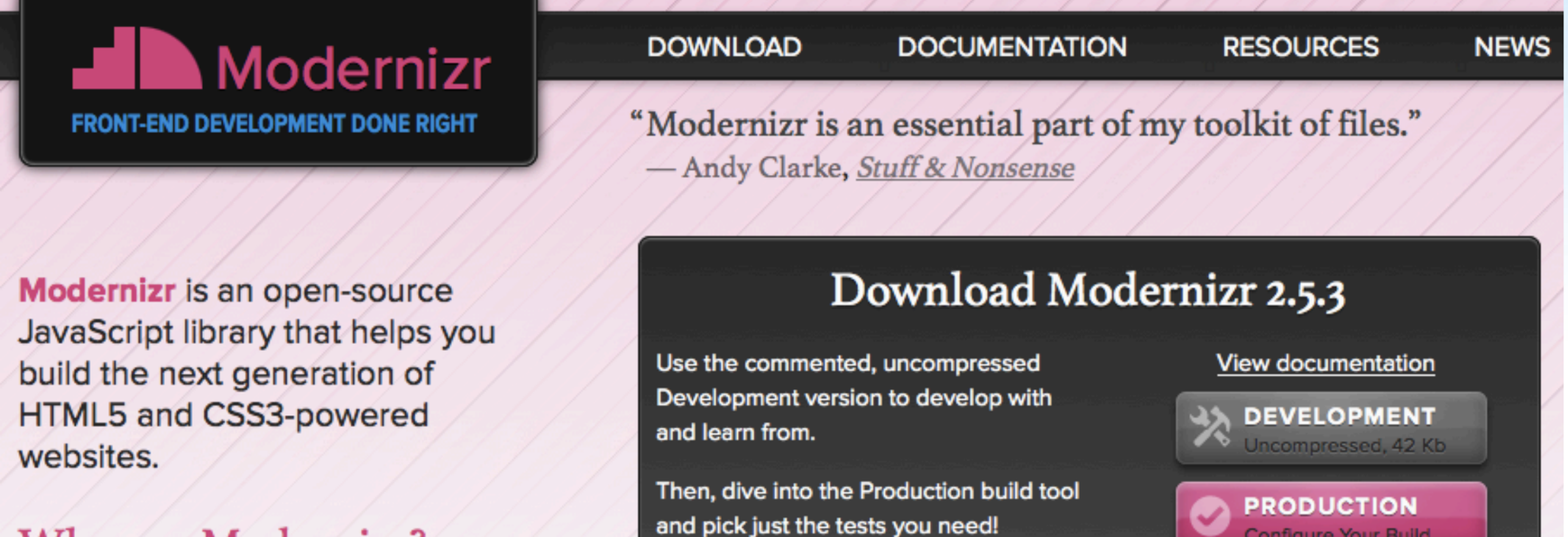
http://www.flickr.com/photos/tallkev/256810217/

When showing all these features a constant nagging is what we could do with legacy browsers. The basics of HTML5 are backwards compatible and by using progressive enhancement legacy browsers still get something that works, but isn't as shiny. But what if that is not enough? Can we make old browsers get what we are trying to do here?

Polyfills

http://www.flickr.com/photos/bramus/7255025512/

Polyfills are JavaScripts that you can include in a page to give functionality that is defined in the standard to browsers that don't support them. By their very nature they have the same APIs as the real thing - simulated. This is great to reach parity amongst new browsers and it can be used to give legacy browsers new features. The danger of polyfills is that you bring intensive functionality to old technology - if you make IE6 animate things it is also up to you to make this perform well and you add IE6 to the browsers you need to test heavily.



http://modernizr.com/

mozilla

Modernizr.com is a JavaScript library that allows you to test support for a certain technology in the browser your users have in a very simple way. It makes it easy to apply intensive functionality only to browsers that can deal with it and it takes the pain of testing for features on for you. It is written by a team of very dedicated developers and used by a lot of big and heavily trafficked sites.

mozilla

HTML5please.com shows you features of HTML5 (and friends) and how safe it is to use them right now and in the environment your users have. You get information how to use the technology with a safe fallback, you get information when to use caution and you also get warnings if something is just not ready yet to use in production code.

HTML5 ★ BOILERPLATE

A rock-solid default for HTML5 awesome.

▸ I'm new here: plz explain why it's good, first.

DOWNLOAD BOILERPLATE 3.0.2 UPDATED FEB 19TH

DOWNLOAD BOILERPLATE | OR | DOWNLOAD BOILERPLATE "STRIPPED" | OR | CUSTOMIZE BOILERPLATE

KEEP THE HINTS AND LINKS | NO COMMENTS, JUST THE BIZNISS. | 100% HIPSTER.

READ THE DOCS ★ CONTRIBUTE ON GITHUB ★ FOLLOW ON TWITTER

HTML5 BOILERPLATE IS 3.0!

MONDAY, FEB 6TH, 2012

The key feature of this update is making Boilerplate smaller. Most of the changes have been working towards that. We have significant work done to our build script, so much so that we thought it deserved its own repository. Meanwhile, here is what's new.

CHANGELOG

3.0 gets some major updates Here are all the commits since last release.

Should you upgrade existing sites? Short answer: nah, you're good.

SIGNIFICANT CHANGES

http://html5boilerplate.com/

mozilla

HTML5boilerplate.com is a zip with a index.html that has all the safeguards you need to create a working HTML5 document that is progressively enhanced for legacy browsers.

320 and Up

The ‘tiny screen first’ responsive boilerplate

320 and Up on Github | Download

This is the new ‘320 and Up’

A lot’s changed since I wrote the original ‘320 and Up’, my ‘tiny screen first’ responsive web design boilerplate. Back then we were just getting started with responsive web design and many sites, including mine, and frameworks and boilerplates like HTML5 Boilerplate, structured their CSS3 Media Queries from the desktop down, rather than for small screens up.

(Oh how we laughed when we realised our mistake.)

To put things right, I wrote ‘320 and Up’. It worked as an extension to HTML5 Boilerplate or a set of standalone files. ‘320 and Up’ has been used by designers and developers all over the web and I’ve used versions of it on every website I’ve worked on since I wrote it.

What’s in the new ‘320 and Up’?

Five CSS3 Media Query increments: 480, 600, 768, 992 and 1382px

http://stuffandnonsense.co.uk/projects/320andup/

mozilla

320 and up is just one of many CSS frameworks that can get you started with responsive design.

Rule of thumb:

Giving legacy browsers a working solution is better than a full one that means a lot of work and doesn’t perform.

Be future friendly!

mozilla

As web developers it is up to us to forge the web of tomorrow. Let’s embrace new technologies instead of feeling the weight of legacy browsers and making them stop us to use great features our users can benefit from.

easy readers *presents...* http://easy-readers.net/books/adaptive-web-design/

Adaptive Web Design

Crafting Rich Experiences with Progressive Enhancement

by Aaron Gustafson

Foreword by Jeffrey Zeldman

> “Adaptive Web Design *not only provides the clearest, most beautiful explanation of progressive enhancement I've ever read, it's also packed full of practical know-how pumped directly into your neocortex through Aaron's warm and friendly writing style. If you aren't already using progressive enhancement to build websites, you soon will be.*
>
> — Jeremy Keith, Author, *HTML5 for Web Designers*

Don't believe Jeremy?
Read a sample for yourself

FREE
DOWNLOAD CHAPTER 1

If you want to know more about Progressive Enhancement I can recommend the book Adaptive Web Design by Aaron Gustafson. It is a quick read full of good information and it comes with a chameleon!

FUTURE ☆ FRIENDLY

In today's incredibly exciting yet overwhelming world of connected digital devices, these are the truths we hold to be self-evident:

Disruption will only accelerate. The quantity and diversity of connected devices—many of which we haven't imagined yet—will explode, as will the quantity and diversity of the people around the world who use them. **Our existing standards, workflows, and infrastructure won't hold up.** Today's onslaught of devices is already pushing them to the breaking point. They can't withstand what's ahead. **Proprietary solutions**

http://futurefriend.ly

mozilla

If you like the idea of embracing the future by letting go of some outdated constraints and without blocking people out, check out Future Friendly which has some very good ideas on how to embrace the technologies and environments to come by changing our approach and workflows.