

An Event Apart

Boston 2010

Disclaimer

by Fred LeBlanc (the author)

These notes were typed completely on an iPad's on-screen keyboard. I was having a couple issues with certain words "autocorrecting" to other words. If you see the word "Ike," it was probably supposed to be like. "Intent" is "content," "e" is "the," and "hints" may be "things."

Also, these were typed on an "as I experienced it" basis. There's some paraphrasing, some quoting, and a bunch of things that might not have been as important as they seemed when I first started typing them. Bear with me.

Hope these help you. 'Twas an awesome time.

Put Your Worst Foot Forward

by Jeffrey Zeldman

Zeldman has learned a thing or two about how to remove the stress from making websites.

Don't: Ignore Your Instinct

The client isn't just interviewing you, you're interviewing them. Same with employers. It's just like dating. You'll know right away if you're compatible or not.

Quick story: a guy in New York makes great nutritional bars, Zeldman was looking to work with him to learn about his brand and marketing. Every time they'd meet, a long talk with no decisions. Gut instinct: this guy is indecisive and felt like a micromanager. By the time the account was won, he should have known not to accept it. He always wanted more. Whenever the client had a question, he thought they were unprepared.

I don't know is a good answer.

The result: ended up firing the client.

Parrot quips: things the parrot can say on each page.

Don't: Dive Right In

Having a boilerplate contract is not a good idea. Gather requirements before accepting the job, and set client expectations beforehand. One template in the CSS doesn't mean one task. There could be many, many variations of that one template.

Communication problems can be typical. We all use the word template, but it means something different to each person in the process. Same when you work in house, your boss will have different ideas of what a "template" is. Be clear about your terms.

Don't: Anything Goes

Remember: liquid.

This is where you report to many people. A good idea in contracts, have one point of contact. They can report to a committee, but you shouldn't.

Zeldman's worst client story, happened just before the train bombings.

A British game company wanted a website, had a billionaire owner, and the billionaire's deputy. Did all the work, nearly done with things when the deputy is stepping aside. What is this? The billionaire sends in a drawing. He decided that he had a vision, but didn't tell anyone.

Response: well, some agencies are good at taking the client vision and making it, but that's not what Zeldman did. In order to finish, they had to throw out work, and were unhappy about it. What he wanted: a photo realistic desk with a pen, a glove, a blood stain; like a Flash website, but in HTML.

They did it, and the client said, "it's not liquid." Billionaires be crazy.

The client didn't like it, and refused to pay. He then demanded his money back. The next day was the London bombing. Zeldman kinda half-hoped that the guy was on it.

Change of leadership is a bad thing.

The project manager should be the one holding the conversation about the change of leadership, expressing concerns from what was in the plan and what was in the contract. The project manager separates business from personal and emotion. The creative director is always too attached.

Walking the halls and smoking cigarettes at 3 in the morning is not web design.

What you *can* do:

Plan for change. Add 30-40% on top, because you're going to spend it anyway. Never budget for exact what is needed. In-house: if there's respect for your job, you can convince your boss that you need more time.

Impressions that you make don't change. Once the trust is broken, it's broken.

When you have bad news, it's tough to tell the client, but it's necessary. Tell it like it is, but use psychology. Put it in terms from their point of view. Say something the client will care about. Give the person a reason to buy into your problem.

Sometimes you just can't fix it. Sometimes, people just want too much information. There needs to be trust.

How to fire your client: it's not you, it's me. This works. A story about Zeldman's first Happy Cog client, a bad client, the only client. "I feel like you have this great vision, and I'd love to be part of it, but I don't think my background fits what you're looking for."

Or: raise your rates.

Zeldman would get hired to do last-minute promotions. The client would come up with a ton of bad ideas and email them over, like animated GIFs and stuff. He would use whatever tool he had, and it was fine and money was coming in. Later on, there are better clients with better projects and more money. You don't want to lose your first client, but you want to change the way you're doing business.

He sent an email about having more lead time, to guaranteed things got done on time. But he didn't change, and kept sending things last minute. So he doubled his rates and never heard from the client again. Someone once said, "you should raise (double) your rates every two years."

One more story: company in France, transcontinental phone calls, lots of work, never heard from the client. Forgot about it. A year and a half later, the client calls saying they're ready to go. After talking, the new rates were double and the client was confused. A couple-hundred-grand-type-of-money is nothing to big corporations.

They asked why, he said, "we've raised our rates." Clients said OK.

Summary

1. Know before you go.
2. Keep expectations on track and in sync.
3. Constantly course-correct.
4. Tell the truth.

5. Phrase it from the client's/boss's point of view.
6. Report bad news before the client/boss notices it.
7. Have a recovery plan.
8. Apologize — but never denigrate yourself or your team.
9. Have an exit strategy.
10. We all make mistakes.

How to fix the change of leadership: have meetings in person. You can head that off really early. Have a kickoff meeting that makes participants of the client. Do games or interactive bits so that the client feels a part of the process (and not just the client). Sometimes, everyone in the room on the client's side isn't on the same page or even know one-another.

Or, you can do a "previously on LOST" type thing. Remind the client what everyone was talking about last time. What went wrong with the billionaire: the boss wasn't involved, the type of person that says they're deputizing someone, but isn't really.

Object Oriented CSS

by Nicole Sullivan

Nicole is a CSS geek.

All of the code is on github.

<http://wiki.github.com/stubbornella/occss/>

Why Bother With Performance?

It isn't just for the gigantic sites. Some stats.

Yahoo!: 400ms drop was a 5-9% drop in full-page traffic.

Shopzilla cut 3.5 seconds (which is huge) and it really paid off. Conversion rate went up 7-12%, page views went up 25%.

AOL: did a bunch of experiments, fastest page load times lead to highest amount of page views. There's a sharp drop-off when things are slow.

Bing & Google: did the same tests and compared results. Bing found as the user experience is worse (time-wise), people stop clicking. As you slow down the page, the user loses their concentration.

Adding 200ms resulted in a 500ms delay in time-to-click.

Google found that over time if you slow down your user experience, people will use your site less.

You have to build performance into your development.

Abandonment, searches, page views and conversions are impacted by performance. Active users are most affected. Your users care about your performance, they may not be able to express it in words, but they express it in their behavior.

The biggest fixes we can make: file size and http requests. CSS is one of the biggest factors in both of these. Just like at airports, you can only have so many bags, and they can only weigh so much.

Back-end optimizations are great, but the front-end is where it matters.

"JavaScript doesn't suck. You're just doing it wrong." — Douglass Crockford
It's the same with CSS.

Things that are wrong with CSS and how we write it:

1. The code is just too fragile.
2. It requires expert ability just to get started.
3. Code re-use is almost non-existent
4. File size keeps getting larger

Understanding The Cascade

Browsers have different default stylesheets. They renders things differently. Invalid rules are skipped.

The order of classes makes no difference. The order of properties does make sense. Same as the order of selector definitions. Same as the order of the stylesheets. Rules are inherited from parent tags.

Specificity

ID vs. Class: the ID will always win.
Inline styles are stronger than almost anything in a stylesheet.
Using !important is stronger than inline styles.

Her rule: avoid ID, inline and important to allow cascade ordering to determine the winner.

The Aggregation of Rules

For example: "message help" and "message error", this is good practice. *Don't* do things like ".message.help" as it leads to rats' nests.

Object Oriented CSS

A philosophy, framework and tool; an evolution of the language.

We've always focused on what's inside the curly braces. But now, we need to focus on what's on the outside. Also, terminology can be confusing, so forget a 1:1 mapping between OOP and CSS.

On big sites, like Facebook, each module on a page may come from different departments. Sort of works the same when working on your own site, you versus you-6-months-ago. Need to find ways to work together.

If you add in required dependencies in the Facebook experience, the system will automatically build packages. This is good and bad, as there are a lot of weird requirements happening at once. Make sure the modules all play nice together. Page per CSS load was down, but there was no need to work together.

Facebook at one point: 706 CSS files totaling 1.9mb.

How to Fix This:

1. Analysis — How bad is the duplication? What solutions are required? Do grep on your stylesheets to see if you can find repetition.
2. Find a Simple Solution — Problems often seem complicated, but that doesn't mean that they are. Ask: what do we know? Can the width vary? Is there unknown content? Can these be nested? Must this clearfix?

Nicole made a "Lego" solution, it affected nearly everything on the homepage. HTML size was reduced by 50%.

Three Goals for Stylesheet Sanity

1. Reduce duplication
2. Improve predictability
3. Avoid CSS kudzu

Things We Can Do

Use hacks sparingly, and when we do use them, they shouldn't affect specificity.

Avoid styling IDs, they also impact specificity and they cannot be reused.

Avoid styling elements directly.

Avoid !important whenever possible.

Style classes rather than elements. (So, not `div.error` but `.error`)

Give rules the same strength.

Layer Selectors

Reset element styles and then stop it

Duplication is worse than stale rules

Reusing elements makes them performance freebies — how do we give great value and design with little CSS

Some clients are good about having their CSS fixed, others make a yearly call to Nicole.

Essentially, developers need to help define the styles used and how future design expands upon them, otherwise you get a CSS mess that people don't trust.

How do you let people know what is available? You build libraries of components for people, possibly stylesheet style guides. You can have the rendered code, a code snippet and maybe an explanation of when you would use the components.

It's worth it to compress your CSS files when you're done.

Using CSS sprites is a great way to reduce HTTP requests. You can analyze your user flows and anticipate the images to be used and preload those when possible.

Don't put JavaScript in your CSS!

The CSS3 Experience

by Dan Cedarholm

Dan thinks people landed on the moon! And that they left a lot of stuff there!

Even when we blast things into space, we leave time capsule type things on them in case someone or something finds them. All of these things create the user experience with the best technology possible.

When we create a message for the web, we're essentially creating a message in a bottle. We don't know who it will be or how best they can use things. We want to add to the store and enrich the message.

Progressive enhancement. We let the capability of the device determine user experience. Think of a black and white TV versus a color HD one. Same content, different devices.

Websites do not need to be experienced the same in every browser.

Sites are three layers: markup on the bottom, style in the middle new experience on top. CSS3 adds to the experience.

<http://css3exp.com/moon>

There are critical and non-critical experience factors. Critical is about the brand, non-critical is where we use CSS3. There is stuff that works and stuff that doesn't work.

Things like transforms, transitions and animation.

Hover

These effects are surprising and delighting. Small simple things that enhance the user experience. Specify a back-up color before using things like rgba.

When defining vendor-specific CSS, put the non-prefixed version last so that when those vendors *do* support things, the generic one trumps all.

Transitions don't go on the pseudo-classes, but the base class.

Opacity

Using transitions and half-opacity is a great effect, changing opacity on hover. Try fully white or black transparent PNGs and set an opacity less than 1 to absorb the background colors.

But opacity doesn't work in IE, so what can you do? You could use IE-proprietary filter option if you had to.

Multiple Backgrounds

You can do a parallax type thing. Remember Silverback? (<http://silverbackapp.com>) This, like most hints, doesn't work in IE, but you should list critical images first in their own background declarations.

Forms

A great place to enrich the experience. Using slight gradients in forms. Check out <http://www.westciv.com/tools/gradients/index.html> for an interactive gradient creation tool.

Buttons

With no images, a lot of great design can be applied. Use background gradients, text-shadow, box-shadow to achieve a nice effect.

On the focus pseudo-class, set a box-shadow on the form elements. You can even make it pulse. In Webkit.

CSS Animations are defined rules, you set them to a name. You can then trigger those named rules with "-webkit-animation" (and the generic "animation.") These animations can also happen on buttons (on hover, for example). But what about other browsers?

It's still a useable form, it just doesn't look as fancy.

With a mix of transforms and transitions (and animations), developers can easily scale, rotate and change opacity of elements in a way that isn't jarring and is visually pleasant.

Rotation

There are appropriate uses of rotate, and then there are most certainly inappropriate uses of rotate. Be subtle.

A good idea: set a transition declaration on a class so that all transitions are exactly the same. Plays off of Nicole's idea of doing things simply and once.

Can you use CSS3 now? Sure, but focus on the things that work now. Just do it, don't tell your boss.

Mobile First!

by Luke Wroblewski

Luke makes many versions of forms, sometimes for Yahoo!

Web products should be designed for mobile first, even if no mobile version is planned.

Google is also doing this, they say they're better and that's what the engineers want to build.

Growth = Opportunity

The mobile web is out-pacing the desktop web by a factor of 8. Next year we'll see smartphone devices will outsell desktop devices. People that use Facebook mobile are 50% more active on the desktop version.

AT&T's mobile data traffic increased 50x in three years.
Mobile may make up as much as 90% of all web traffic by 2014.

Don't just port and rethink things for a small screen.

Constraints = Focus

Focus on core actions, know your users, and use scalable design.

Screen sizes: on good mobile devices, the screen is 80% smaller. For a good example, check out Southwest Airlines sites. The mobile version is driven by the tasks that people need to do.

Some things make more sense on mobile than on the desktop. Luke isn't saying that they're equal, just that the two are equally important.

For Flickr, there are a lot of options (60!) on their standard website, but the mobile version only has 8 things to choose from.

You need to know your audience. Designing for the mobile web isn't just for homepages, but for content pages too. A good example is on Expedia. Had they focused on their mobile version to start, they may have ended up on a better desktop design.

All these examples are on the iPhone, here's why:

- It takes 30 times the number of devices to get the web traffic 1 iPhone delivers on average.
- Design matters on the iPhone.
- What you can do is all on the screen, you have to put everything on the screen.
- Not having menus makes the content the interaction point.

However, if you're doing a mobile app, you need to think of everyone, and pick your target devices. When you decide, you should think about not just screen size but screen densities.

The iPhone has 164ppi. The Palm Pre has 186ppi. The Pixie has 194pp. Android and others are well over 200ppi. The standard pixel density on something such as a cinema display is around 96ppi by comparison.

How do you deal with these?

1. Define device groups.
2. Create a default reference design.
3. Define rules for content and design adaptation.
4. Opt for web standards and a flexible layout.

Speed

Keep performance top of mind

Take advantage of HTML5

Eliminate redirects, use CSS sprites to serve multiple images, consolidate CSS & JavaScript into a single file, minify your code. Load contents lazily, use application cache for local content storage. Use CSS3 and canvas tag instead of images where possible.

Content

Quick bursts... everywhere

One-handed

Your mobile is with you all of the time, so designing for mobile is designing for something that can be used all of the time.

The most popular apps on mobile are things like Twitter apps and Facebook, both support quick checkins.

Across 640,000 URLs, 25% of all documents were displayed for less than 4 seconds.

Capabilities = Innovation

Touch

Simplify your user interface

Don't count on hovers

Ubuntu just released touch interaction guidelines. Thinking about mobile first forces you to think about a touch interface. Nokia is projecting the majority of their phones are headed towards touch-enabled (away from keypad and/or qwerty only).

When designing for touch:

Touch targets: Apple recommends a minimum target size of 29px x 44px. Because our fingers are fat. Microsoft says 34px, 26px minimum. Minimum spacing between elements is 8px. Visual size is 60-100% of the touch target size.

All of these are based on an MIT study about the general fingertip size.

Also, the target for touching a button might be the size of the button, but the area of action while touching the button is about 160% of the size.

There are many different platforms and suggestions for things, the following are the core gestures that are used:

- Tap
- Double Tap
- Drag
- Flick
- Pinch
- Spread

- Press
- Press & Tap
- Press & Drag
- Rotate (in both directions)

Microsoft did an experiment with their Surface and asked people what type of gestures they would do for things. Results are people made question marks for questions, and check marks for accept.

Drawing is making it's way into interface, for example circling things that you want.

Remember: there is no hover. If it's important, put it on the page. A hover is not intentional, a click is intentional.

Location
Positioning, filtering

Yelp is confusing on the desktop, but the mobile version is great. It uses location to help hone your searches by default.

GPS is cool, but drains your battery. Wifi location is pretty accurate, is accurate enough.

So many more capabilities:

- Multiple orientations (not just portrait/landscape, but your direction)
- Voice as input

Google Goggles uses many web capabilities in concert to tackle things like visual and audio input.

It's apparently Luke's birthday. Everyone get ready to sing.

Learning to Love Humans — Emotional Interface Design

by Aarron Walter

Aarron loves MailChimp.

Who are we as humans?

People are getting more formal on the web, and this is probably because of the way that we interact with one another on the web these days. Things are getting more public each day.

We need to understand core needs.

Maslow

A pyramid of human needs: eat, sleep, bathroom, love, belonging, safety, etc.

In our interfaces and how we use computers can be very similar. We red things to be pleasurable, usable, reliable and functional. We want things to be a delight to use, and that's the piece that often gets overlooked.

The bar is currently low.

Saying that you want something to create something at is usable is like a chef saying they want to create something that is edible. Spam is edible, but is it pleasurable?

Basecamp: Usable, but not pleasurable. That's not a slight at it, but there are things out there that can do all of that and more. Wufoo, on the other hand, has personality.

Personality is a platform for emotion.

(Aaron's slides are hilarious.)

Gestalt — a German word meaning scanning for a pattern, and finding something that stands out. We ask "is this good for me or bad for me?" We do this in business, we do this in dating. It's contrast.

Contrast is built into the animal kingdom. Zebras have stripes and when in groups, moving around makes it confusing for predators. Aaron also speeds on the highway.

Bringing up Boarding Pass/Fail, a site of designers redesigning boarding passes.

Why do we feel emotion? It's built in, an evolutionary measure to keep us alive. For example, helps babies alert parents of hunger, fatigue, discomfort, etc.

Emotional Design Principal

People will forgive shortcomings, follow your lead, and sing your praises if you reward them with positive emotion.

The Volkswagen Bug is one of the most popular cars because it has personality. On the iPhone, Tapbots is doing interesting work with their apps because they use personality. Personification is important.

Software shouldn't be a thing you have to use or make, it should be a fun time while still accomplishing tasks.

Another great example: Feathers on the iPhone.

Aarron's mom used to say, "you can't polish a turd."

Wufoo attempts to be the opposite of all of their competitors by introducing fun into things. MailChimp uses the chimp in the top to entertain people. There's also "party pooper" mode, when the box is checked, the fun goes away.

The fun of the open system is that it's customizable and personable, like the Magic 8-Ball. You project yourself onto the experience.

Treats are fun and special because of the discoverability.

Also check out:

<http://www.photojojo.com>

Great balloon to learn more, cart trick.

Humans have doubts. We can't help it, we're skeptics. It goes back to the question, "is this good for me or bad for me?" A site like Mint makes financial stuff fun. It's not cute, it's elegant; people need to trust it. People need a compelling reason to trust it, and they do because it's beautiful.

As humans, we make mistakes. Example: Flickr, made their down page a contest to win a pro account while they worked to fix things. People ran with it, it made a bad situation a triumph.

Humans have limits. There is only so much attention that someone can give to something. When you add something in, you should consider taking something away. We have limited bandwidth.

CSSDrive is interesting, but it's overwhelming. So we should fall on Occam's Razor. We say people are lazy but the real truth is that we just want the path of least resistance. Also, sometimes you need to bribe people. For example, Dropbox has a great introduction screen. Gives you a fun checklist to get set up, and if you fill up the box, you get 250mb of free storage.

CoffeeCup makes software in Atlanta. Had a contest to find eggs around their website giving away \$18,000 in free stuff. Page views went up 600%; 300% increase in traffic, 217% increase in Facebook fans; 170% increase in Twitter followers.

While they look around the site looking for these eggs, they're learning too. It's turning non-customers into customers.

Are we changing with the web? Maybe, maybe not, but we're definitely becoming more connected, and more connected means emotional.

Anatomy of a Design Decision

by Jared Spool

The world loves Jared Spool.

Let's talk design decisions. Let's talk about Grey's Anatomy (the medical text). It allowed folks to study the structure of things. What are the internal workings of design decisions?

Design decisions can go either way, good or bad. NYTimes.com versus HavenWorks.com. Etsy.com versus Arngren.net

By the way, HavenWorks validates.

37 Signals versus Don Norman, a celebrity death match. 37 Signals says that they don't design for others, they design for themselves. Don Norman thought that Basecamp was OK, but didn't like it. Said developers were being arrogant. Why was Don so angry?

37 Signals' choices are all design decisions, it's called "Self Design." Don probably doesn't like it because he doesn't do what 37 Signals does. It works if you have enough people doing what you do, but it's not the only way to do things.

Sheraton makes some crazy design decisions, their welcome message is behind a button. American Airlines has bad error messages.

There is "Unintentional Design." This is what happens when we don't think about the user. There is also "Genius Design," or designing for other people.

NewCityMedia makes educational websites. They talk to parents and students and schools and are finding that a lot of data is showing patterns. They take this data and creates websites with it. this is Genius Design.

There is "Activity-Focused Design." The design solutions are focused on accomplishing the activities required by the app.

There is actually one more type of design, "Experience-Focused Designed." Six Flags focuses on the rides, whereas Disney focuses on what happens between the rides. This is the experience. Disney also makes animals out of your towels when you're not looking.

It's possible to design a usable set of experiences and still have a bad website.

For example, the Neonatal Intensive Care Unit (NICU), the app for ordering eats must be simple and must include everyone. You have to do a lot more research on a lot more experiences to create a good design.

Rule-Based Decisions versus Informed Decisions

Rule based sounded more natural, but informed decisions are better.

People like to put things under trees. But the University of Penn didn't fall for it. Except that none of their websites look the same. They needed guidelines to help unify things. The guidelines all were expensive to produce, people never followed them and they all failed. Don't make design decisions because of rules, it prevents thinking (by the designer). However, informed decisions require thought.

Also, rules-based designs fail on any exception cases. Informed design succeeds during normal and exception cases.

Many companies were interviewed, some that succeed a lot, others than often fail. To get work done, you perform a process. On one side, there is a Methodology. These can be formal or informal. Beyond Methodology is Dogma.

Jared flies. A lot. And he is trained to observe things, like pretty women. The TSA required that she put her lotion in a plastic bag. A ridiculous rule, but Jared was there to provide a plastic bag. His bag. They went through the security point, but that wasn't good enough. Why? Because of the spot checks?

Dogma is an unquestioned faith independent of any supporting evidence.

On the other side of process, there are Techniques. These make up the building blocks of a process. Techniques are things that we can practice to go into the process. Beyond Techniques are Tricks. Tricks are Techniques that we don't use correctly.

Sometimes it takes too long to get the right tool than to just use what's in front of us. Rule-based decisions used Methodologies and Dogma, informed-based decisions use Techniques and Tricks.

Design patterns (which are pro-active) are better than style guides (which are reactive).

From Unintentional Design to Self Design is eating your own dog food.

From there to Genius Design is usability testing.

From there to Activity-Focused Design is field studies.

From there to Experience-Focused Design is personas and patterns.

- Every style has a purpose. It's OK to use Unintentional Design, just admit that you're using it.
- Great designers know which style they were using.
- Great designers use the same style for the entire project.
- Great teams ensure everyone uses the same style.
- The more advanced the style, the more expensive. Agencies don't go beyond genius design.

- The more advanced the style, the better the design.

This is not rocket science, Jared knows because they're one of their clients. It's not brain surgery either.

What kind of designer do you want to be? You need to know what decision style you use. Encourage informed decisions and avoid rule-based decisions. Focus on your techniques and tricks.

For more, <http://www.uie.com>

Everything Old Is New Again

by Eric Meyer

Eric co-hosts An Event Apart and has finally cut his hair.

We're not going to talk about web fonts because everyone else is doing that. Also, Eric isn't great at typography.

The Box Model

Some struggle with this because it's specified in a very interesting way. Padding isn't included in the width that you specify in the CSS for an element. This hurts when you are trying to make columns.

With CSS3, there's the box-sizing property, like `box-sizing: border-box`. Now when you do three 33% columns, the float drop doesn't happen. That's pretty awesome, it's getting rid of nested divs. If you're going to use it now, you'll have to use vendor-specific prefixes.

Fewer elements makes Eric Meyer happy.

Media Queries

It would be nice to change the layout when the screen size changes (but without JavaScript). Also, people come in with many different sized windows.

```
@media all and (min-width:500px) and (max-width:799px) {  
    /* your styles */  
}
```

```
@media all and (max-width:499px) {  
    /* other styles */  
}
```

You can do pixels or ems in these declarations. This is pretty fantastic. Ethan Marcotte just wrote about this on A List Apart today.

Rounded Corners

There has always been a lot of hacking around this. You can do the sliding doors technique, or maybe other non-sensible options. It's a lot easier now.

```
border-radius: 20px;
```

Plus all of the vendor-specific options. When you want to change the radii of the corners, webkit does (or used to) be more implicated than the typical shortcut. Why do we need the vendor-specific prefixes? Because the definition for this property is not finished yet, and as such, there's no guarantee that all of the vendor versions are the same. The main confusion: how should the browser handle rounded corners when two different size lines meet?

So, vendor prefixes: annoying but necessary.

Sliding Doors

Where you have a couple of different nested tabs with background images on different sides that will style opposite sides of an element. With border-radius, this becomes a lot simpler. But what about when they get fancier?

We could use a sprite, but that's messy and complicated. Instead, with CSS3 we can use multiple background images and position them around the edges of the box. Use translucent PNGs so that when you change the background color, it shines through and nothing is affected.

Accessibility-wise, having many consecutive a elements is a loss. You may consider putting them into lists.

Anything you could do with sliding doors you can do with multiple background images, but Eric isn't sure that you'd want to.

Translucent Colors

With rgba, we don't have to mess with box opacity. Except when you take the blue that you want and change the opacity on top of black the blue gets darker. After lots of numbers, he lightened the blue and everything worked out.

Things are still a little different. And things won't be exactly the same. Also, ripples are the future.

There's also hsl and hsla that can be used. Eric doesn't fully process things that way, but it can be helpful.

Child Selection

When you add a bullet after every a element in a navigation, for example, that's great until you get to the last link. You used to add classes like "first" and "last," but now there is last-child.

You can also avoid putting the "odd" class in tables by using nth-child(2n-1). This syntax can be confusing. When you use nth-child, it's not nth-child of that element, it's of all children. However, only the specified children will be styled.

Where this can come in handy, when you have a bunch of images floated left, but some are taller than others. You can then find specific images and add a clear: left; onto them. You can quickly change the number of images wide in the CSS. Combine this with media queries and you have something pretty powerful.

But big deal, they're images. They're made of pixels. But you can do this with anything, including divs of content. Amazing!

No mentions of browser compatibility. Why not? Because things are changing so fast. Since this talk 6 weeks ago, IE implemented media queries in IE 9. But more importantly, all of these can be done in the modern browsers (maybe with a jQuery plugin, etc.).

Oh, one more thing. Media queries fire when you resize the browser, not just on page load.

Paranormal Interactivity

by **Jeremy Keith**

Jeremy is the author of many fine books and many fine websites. He's from ClearLeft.

Communication is the key to interactivity.

We use iconography to communicate meanings, but they can also miscommunicate. Icons can go out of date, like the save icon being a 3.5" floppy disk.

We can use writing to be more specific, but the problem is that it's not universal. We speak different languages, and that makes it hard. Sometimes we need to use words as there just isn't any way to communicate things with an icon clearly.

This communication is interaction.

We like to relate to things as if they're another being we can interact with, and not just living things. For example, MOO uses the little MOO robot, but eventually we're going to get tired of the happy, friendly messaging.

Calling people by name: easy to do on the web, hard to do on chalkboards.

Choose your own adventure was essentially the first version of hypertext. How we move between pages: links & forms. The `a` tag is what makes the web awesome, otherwise, the web would just be a stack of documents you could randomly read.

Just because you can do things on hover, doesn't mean you should. Especially in navigation. Mobile devices don't even have hover. But hover isn't evil, it can be used to nicely enhance the interface.

For example, on Huffduffer, the main logo presents a hand pointing to the left when you hover over it. If you miss it, no big deal. Hover shouldn't be used for vital tasks. And when you're using hover, you should also do focus. There should be the same number of hovers and focuses in your CSS file.

We can now do things with HTML and CSS that it used to require JavaScript to do. What used to be mouseover is now available via hover. This is moving from procedural to declarative.

On forms: they're awful. They're associated with bad things, like taxes. A lot of times, companies and organizations will take their paper forms and translate them right to the web. This can be dreadful.

Huffduffer, on the other hand, presents its sign-up form in paragraph format. Not particularly difficult or crazy, but people loved it. Other people started doing this too. But let's not put all of our forms into paragraphs.

Form validation and UI elements are also moving from procedural to declarative with HTML5. There are many new type declarations in HTML5 that will help, like: range, email, date, url, number, search.

Progressive disclosure is when more information is revealed. This shouldn't happen on hover, but on click.

At ClearLeft, they've also moved away from styling IDs, but they're still useful because they make things addressable. But your hrefs should always link somewhere, never just #.

Trick: have jQuery add a "hasJS" class to the body, then you can style things in CSS specifically for instances where JavaScript is around. You want to create safe defaults.

HTML + CSS + JavaScript

Separation of these parts makes things easy. Ajax is a server request that doesn't get back a full page, but only a part of a page. You should build a non-Ajax version first, using links and forms. After that's working, then go back and add in your Ajax interactions. You "hijack" the link clicks and form submissions. He calls it "Hijax."

Also, having a loading symbol that can appear when Ajax is doing it's thing is helpful. Is is called feedback.

Feedback

Even if you provide a complete experience that is fully safe, people won't feel comfortable unless there is sufficient feedback. Also, the feedback needs to be accurate for the level of action that is happening.

A great example is collecting coins in a game (it makes a sound) or scoring (small numbers fade in and out). Find inspiration in other forms of media, but don't emulate them.

Inspiration is good.
Emulation is bad.

An example of emulation: Rich Internet Applications. Why emulate the desktop? There are not hyperlinks on the desktop.

Other examples are Xbox vs. Wii; normal movies vs. 3D experiences.

Rich doesn't mean flashy graphics. It means a deep, fun, delightful experience. We've moved into a world of producers and consumers. Most of us are both.

We only need a word for interactivity because we've been living in a world dominated by non-interactive media.

Message and Medium: Better Content by Design by Kristina Halvorson

Kristina is brilliant and pixieish. And she loves talking at An Event Apart.

All About Your Teeth

We're told to keep our teeth clean and great, but the dentist is always a daunting experience. Especially finding a new dentist. The best way to find a new dentist seems to be asking family and friends for recommendations.

Or maybe you ask someone at a party. And when that doesn't work, you'll go to Google. You click a link, and get their content.

Page design is not always indicative of the content within.

Unintentional messaging occurs when you approach a site from what you need to say, not from the message that people should experience.

Content Strategy

Over the last 18-24 months this has become popular. People want content that is useful, usable, purposeful, productive, and profitable. It hopefully prevents your in-house team from destroying the sites you create between when you're done and when they put the content in.

Even if you're a not-for-profit or educational center, you still have a bottom line to worry about, so profitable always applies.

Content is text. It's page copy, articles, links, labels, Flash elements, alt attributes, error messages, task instructions, forms, search results, metadata, it's everything.

Strategy is not the big idea of what you're going to deliver, but the entire body of why you're doing what you're doing, and where it's going to go, and when it gets delivered. What happens after it's out there? Are you measuring it? Are you archiving it, deleting it?

We're not just launching content, we enter it into a lifecycle. Everyone needs to be aware of this, and how it works.

There are four core components: substance, structure, workflow and governance. Substance and structure are content components, workflow and governance are people components.

Today's focus: *substance*.

Messaging is not a mission statement, a brand promise or a tagline. Messaging is an internal tool used to prioritize content types and choices. It keeps content consistent over media and time. It guides design choices, it aligns content owners.

Get your mission statement off of your website! People don't care about this, this is a statement to direct your internal team, it's not for your customers.

The messaging pyramid: in the first second, people need to see the brand, feel comfortable and know that they've found what they're looking for. At 10 seconds, you need to have conveyed your primary message. At 2 minutes, we need to convey a secondary message.

All of those feed into a centralized call to action. It should always be there.

Messaging In Action

As an example, AwesomeCo has a great website, good page load but high bounce ages. They need a good primary message.

Primary message is who you are, what you deliver and what the user gets.

They also need a secondary message. These are supportive of and are born from the primary message.

Beyond secondary messages, there are details. These all point towards the call to action, either subtly or directly. Examples: product descriptions, about us, case studies, help content, etc.

Bad ways to work with the content people, give them a stack of templates that need to be filled in. A better idea: page tables, specific, wireframe-based content definitions. It bridges the gap between structure and message. It looks like a page template, but it also says what type of role the content is supposed to play. It has the objective of the page, the source materials available and a list of maintenance that will be needed in the future. It also has a list of key page messages that need to be conveyed and a priority of the content to be displayed.

Messaging Across Your Site

As an example: Ben & Jerry's. "We always craft our eclectic collection of euphoric ice cream flavors with love, all-natural ingredients, and a plan for global goodness." This message resonates within their content all across the site. Unfortunately, it seems that the content writer didn't get to work with the designers. The content sections seem out of place and poorly visually designed.

Their tone on their about us page and even the press release section all echo the brand and the content strategy. They could have gone boring in these areas, but they went the creative, on-target route.

On the contact us page: completely out of place. There's even an awful FAQs database which contains many questions that sound like they were typed by a robot. What happened?

Onto the activism section. While the message is important, there is probably a better way to say what they're doing. There seems to be a struggle or things written by the PR people and the content strategists. It's obvious where people were simply filling in a page template versus designing around content.

And onto the social media channels. People can see when you've logged in. You need to keep things up-to-date and get engaged. They probably hired an agency to build out a social media strategy, the agency built them a Facebook page and sent an invoice. You have to do more, just putting something up isn't good enough.

On Twitter: way too formal. Also, they just search for and retweeting people, and very now and then sticking in pointless messages.

Kristina also considers external site searches as important to content. Even the summary line on your Google search result. It all needs to follow your content strategy. And you should make sure that in your editorial calendar that when content comes down it comes down!

Ben & Jerry's has a mobile app. Instead of saying, "here's everything on our website," they kept it simple. Very image based. It has a lot of maps, and the ice-cream-on-shelves thing, but then when you get to the flavor descriptions, you get the desktop web copy. They should have involved a content strategist to say what would fit there better.

Messaging isn't something that we need to just consider on our homepage or about us pages. It needs to be everywhere, throughout development and design so that everything is one harmonious experience. The three need to work together so that nothing seems jarring or out of place.

Most spend a lot of time thinking about the 1 second in the pyramid, but it's time to start thinking about the rest. How does this affect my site? This doesn't take a ton of people to think about this and start doing something.

How can you begin to make content and content strategy something that is accessible but fun. It needs to shift from a science and study thing to an action. Work with content creators and find out how you can make a better product with them.

Hardboiled Web Design

by Andy Clarke

Andy is writing a new book and comes from England.

What is hardboiled?

Think of it like an old-school detective story. It's the heroes that matter, because they give everybody hope that things will be OK. They operate in such a way that they're not apart of the rules that we as a society follow ourselves.

We route for these heroes because we need them; because we can't do it ourselves. The web needs heroes.

Have all the questions about CSS been answered? Not yet.

People think that we can't use CSS3 because all of the browsers don't support it. Progressive enhancement is building for the least capable devices first, and then we add things on from there, including CSS3. People have been talking about progressive enhancement since 2003.

That's 7 years ago.

Most start at the bottom and enrich from there, but when you do that, there is only so far that you can go up. We need a better way.

Dan Cedarholm calls it *Progressive Enrichment*. Is he wrong? Maybe. We need to be creative and take things a step further. Some say that IE is holding back the web, but that's nonsense. The only thing holding us back is the way we create things.

The fact the websites should look the same in every browser is old thinking. We need to change people's perspective that the web is like print. It isn't.

If you're someone that says that you'll wait until you're told it's OK to use CSS3, you'll wait forever. CSS3 isn't even a standard, it's broken down to be modular so that browsers can implement them when their business needs allow.

The W3C is not an innovation body. They don't just make a standard and then everyone is required to implement it. This is because the working group is made up of many of the paid players, all suggesting ideas on how to implement each things. We need to rise above the politics and just get on with it. How do we do that? We get hardboiled about it.

Instead of just creating experiences that works in IE and going from there, let's work backwards. We should design for the best browsers available and then find ways to scale back for less capable browsers.

Hardboiled 101: The 404 Page

Keeping the markup simple and using things like web fonts. It makes something that looks great and is easy to go.

CSS3 has a whole range of new selectors.

Interestingly enough, it isn't the old browsers that are the problem, it's the less-capable ones. For example, Camino 2 (which is new) still uses the old Gecko rendering engine that doesn't yet support web fonts.

Check out things like `:nth-of-type` instead of `:nth-child`. You can do many transforms: translate, scale, rotate, skew. You can even move the transform origin.

Between hover and focus, and all of the transforms, there's a lot of possibilities. You can transform many, many properties as well.

It's possible to do things like 3-dimensional flips just in CSS with animations.

Are we going to wait until everyone catches up, or make awesome things now? Your things won't look the same everywhere, but they're all usable. Things should never look broken, but most people don't even know that there are different browsers.

What does browsers testing mean? It used to mean checking that things look the same in all browsers. So now what? Now it means that each browser needs to display content at a level that is appropriate for that browser.

A Little on Vendor Prefixes

They're better than the old hacks, so stop complaining. If you check, a lot of the stuff now can be done with crazy DHTML stuff from back in the day. But that is ridiculous, things don't need to look the same in every browser.

If we use CSS3 to push forward, browsers may be forced to jump on board if they don't implement the best technology that we want to use.

Check out the CSS3 library that attempts to bring CSS3 selectors to your CSS. The library plays well with other libraries and is only limited by the strength of the JavaScript library you're using with it.

Modernizr

This will test a browser for CSS3 capabilities and it will apply a list of classes on the HTML tag. This lets you specifically target the different capabilities of each browser. If a feature isn't found, it will add a no-feature class for the feature missing.

We need to design to the capabilities of the browser. We don't have to wait, we can start doing this stuff now. And not just visual sprinkles, but going all the way to the core. We need to stop working bottom-up and to top-down. We just need to make awesome looking things.

Ignorance is bliss.

When we get hired, we're not being paid to show up, we're being paid to lead.

A Dao of Flexibility

by **Ethan Marcotte**

Ethan makes cool stuff and writes books.

0.

Ethan accidentally deleted his keynote presentation.

There's a tension between the web as we know it and how the web is going to be. So much of how we describe the web is from print, but we also borrow terminology from painting on a canvas.

As web designers, we create a canvas and work from there. But technically speaking, Photoshop isn't our canvas. The web browser is. The browser window changes sizes, it's unpredictable. If we don't do a good job, sometimes the user gets penalized, but sometimes it's our content.

We need to accept the ebb and flow of things.

Users have so many more wireless access points than we ever expected.

If you look at the acceptance rates of things like AOL and Netscape, the iPhone and iPod Touch have outpaced those dramatically. It's crazy.

"I need an iPhone website." Ever hear this? How do we create tailored experiences to all of the different devices available to us today?

Responsive architecture is asking how a space can redefine itself to meet the needs of the people that occupy it.

Introducing: responsive web design.

The flexible grid that incorporates media queries to create a responsive, adaptive layout.

Every artistic movement is a counterweight to the movement that came before it. This translates to the web.

The grid started in print design, and some brought the grid to the web. This is good, except that they're all fixed-width. Can we have both?

Happy Cog likes using ems for font-sizing. It allows for flexibility while maintaining precision. To get the em size that you want instead of pixels, you divide the target size by the context font size. This will give you the new em value to use. Don't worry about decimals, you can drop those right in.

If you can do this for fonts, you can do this for structural elements as well. When you do this, you can then change the width of the holding element (in CSS or by changing the width of the browser window).

But how do you incorporate images? `img { max-width: 100%; }` Easy.
And movies? `img, video, object { max-width: 100%; }` Bam!

This will let the images or whatever to never be larger than the size of its container.

IE6 doesn't support max-width. To fix this, you can give not so great browsers something like `img, video, object { width: 100%; }`. But there could be a problem if the scale goes higher than the original size.

You may have noticed that scaling in some older browsers looks horrible. If you use the AlphaImageLoader proprietary IE filters, the rendering engine works more like most modern browsers. Amazing!

Flexible grids are not bulletproof or future-proof. For example, if we scale down too far, they're almost pointless. If we scale too high, images get out of control.

Generally, different devices will need a different experience, and we can do that by parsing through the browser's user agent string. You can do this in JavaScript and allow the client end to handle it, or you can parse it with a back-end technology like PHP and redirect people to different versions of your site/app.

People have been playing with resizing the browser and adjusting the design appropriately for years, mainly with JavaScript. But now we have media queries! We can check a number of factors and apply certain styles to certain situations.

When you have multiple media queries, they can inherit from one another. For example: when you have a max-width rule for 320px and 640px, the smaller one will render the 640 stuff too because both are under 640px.

As far as support, the big ones are pretty much there, and IE9 has committed to adding it in. There is a project on Google code that will help you implement this in older browsers.

Fixed width is not bad, but it's not the only way. The future of the web is designing for flexibility.

How the Web Works

by Jeff Veen

Jeff has a great history on the web, and he's going to talk about web history.

A small topic: the history of the web.

Beer

Beer can be a hobby. A delicious, delicious hobby. There are beers that are cask-conditioned. It is brewed in the traditional style of how it was made many years ago. We take for granted that a lot of the drinks we consume are cold. Just 200 years ago you couldn't get a cold drink in Boston around this time unless you were rich.

Ice

The rich had ponds where they'd grab the ice in the winter and bury it until they needed it. That is, until Frederic Tudor came along and made ice more available to people. What was innovative was how he shipped ice all over the world. He sold people frozen water.

However, if there's an existing market, people will try to incorporate processes and robots to make things more automated.

Dr. John Gorrie invented the ice-making machine. You didn't have to harvest it anymore, but you could make it. These machines were the size of buildings, so each town got one. And then eventually it became more optimized to the point that it fit into your refrigerator.

GE capitalized on this and other technologies by bringing them down to the consumer level.

The interesting thing: with each optimization level, not one company made the leap to the next level.

James Marshall found gold in California and tried not to tell anyone, but soon enough everyone knew, thus began the gold rush. At the time, venture capitalists would invest in prospectors sending them to the gold areas and taking a cut of whatever they brought back.

It was important to get the gold from the west coast to the east coast, but going around South America was inefficient, so there was the pony express. This was short-lived when the telegraph came about as people were then ordered credit when they found out who dug up what. Once the first telegraph line was wired from San Francisco to New York, the pony express closed down 2 days later.

Wells Fargo was responsible for this, and they're still around today. So, some companies don't make it, but some do.

In the mid-1990s, computers came with CD-ROM drives. This meant you could rip your music. The MP3 came around and then boom, iPods.

We've had 600 years of typography, and now we're rebuilding that on the web. In 1993, there were no fonts. Now we have about 18, only 4 of which you'd want to use. And now we have @font-face.

The font foundries weren't ready for that, so there was hesitation.

Ice = Health
Gold = Communication
Media = Attention

How to Win on the Web

The qualities that contribute to the success of the web are the qualities that will make us successful, too. We need to be native to the web.

IETF is like the W3C of IMAP and such. The creation of the web standards are people locked in a room, each arguing and trying to be right.

Netscape early on proposed JavaScript-based stylesheets. It didn't work, but eventually (over beer) Microsoft and Netscape worked together to propose CSS as we know it today.

The IMG tag was suggested by Marc Andreessen. You know, the Mosaic guy. Tony Johnson was also thinking about this, but wanted to use an ICON tag. SO how did img win? He shipped it first. And that was that.

Typekit was built around ideas created in notebooks, threw it out on the web and there seemed to be instant traction. From there, there were iterations and feedback from the

Internet, implementing the best parts as fast as possible. Twitter has been great for quick feedback and turnaround.

"If you're not embarrassed when you ship your product, you waited too long."
— Reid Hoffman

It's true. Think of Google and Facebook and Amazon.

Twitter is doing a great job of listening to users and implementing the great ideas (and quickly).

The speed of iteration beats the quality of iteration.

Metcalf's Law: every time you add a user, things are getting exponentially more valuable. The reason the Internet got popular was because of the ease at which it could grow. That, and the ability to link to other documents.

Information wants to be free. Computers process files without regard for what they are.

Excludable: once I sell it and you consume it, it cannot be consumed again.

Rivalrous: a product can prevent access to those who don't pay.

Combining these makes for very interesting results.

Excludable & Rivalrous: an apple

Non-Excludable & Rivalrous: fishing

Excludable & Non-Rivalrous: satellite television

Non-Excludable & Non-Rivalrous: everything that is digital

That's the fundamental change. There are barriers in place by default, so the attempts to make things excludable are complicated for the user and have all failed.

Consensus = Code

Value = Users²

Information = Free

We have to stay on top of our game. We need to advocate for our standards.

The web transcends what happened with ice and gold, and maybe even media itself.

We need to push back against walled gardens. We need to push back on people telling others how to code. Some companies absolutely don't care, we have to watch out for that too.

And That's That

If you've liked these notes, hated these notes, have any questions, found a typo or just want to send virtual high-fives, you can find me on Twitter at @fredleblanc or email me at fred@fredhq.com. Happy websiting. :)