Designing the Future Web
jeffrey veen

Thank you. It's good to see to you all. We are going to shift gears a little bit now, and talk about design and technology. And specifically some of the trends that I see and where we are heading in the future.

So if you can, close your eyes again and take that small baby and switch it over to the other arm, and let's talk a little bit about designing the future web. I'm going to look at three things. Architecture, patterns and what I call self-aware content. The logo at the top, Adapt-A-Path, let me just quickly mention that that's a company I've started recently with a group of other user-experience designers and information architects having left Wired and subsequently Lycos after 6 1/2 years at the same job in the dot com industry, which I think is something of a record.

I am a technologist and an information architect who has worked with some of, what I consider the best visual designers in the world, particularly when I was working at Wired magazine. That puts me sort of in the middle. I have this little diagram that I like to use to talk about it. In the middle I'm talking about design and integrating design with architecture in web sites. Structural design of sites as well as how web sites interact with their users. The behavioural aspect of it. And that sort of gray box in the middle, that's our web design project. That's the product that we work on. And that's also where I typically sit. In between the structure of a site, the presentation of a site and the behaviour of a site. What I want to talk about today is how that's changing.

"And he asked me, he says…and I'm trying to get my gifs to dither into the colour save palate and I'm wondering do you know how to make PhotoShop do that? And I'm like, oh, give me the oxygen."

I have a story to illustrate that. I too am from San Francisco. San Francisco is a city with a dramatic geography. These incredible hills, but with industrial revolution city design. We have a grid on top of the geography that doesn't pay any attention to the geography which leads to streets that go straight up and straight down. And I'm an avid cyclist. I enjoy, what I call masochistic cyclist, riding my bike up and down these hills. About 5 years ago, I was cycling to my office at Wired, coming down one of these hills really fast, like 40 kilometres an hour, flying down this hill when a car turned left in front of me. I hit the car, flew across the intersection and landed on my shoulder in a crumpled heap. An ambulance came, picked me up. Put me in the back. Put my bike in the back. Of course my bike comes with me. Off we go to the hospital. The guy in the back, the EMT is asking me, so how do you feel? Are you all right? I'm like no, I'm in a lot of pain. He says like well do you want some oxygen or anything? No, no, that's all right.

So he tries to get my mind off of what's going on. He asks, so what do you do for a living? And I said well I'm a web designer. I build web sites. It's like ah, that's very interesting because I have a web site. And he asked me, he says…and I'm trying to get my gifs to dither into the colour save palate and I'm wondering do you know how to make PhotoShop do that? And I'm like, oh, give me the oxygen.

But the reason I like to tell that story is because it illustrates where the web was 5 years ago. We are all the producers of content; we also consumers of content. We were sort of designing for ourselves back then. The reason that much of what I do today is user research, in the context of architecture and design is because that's not the case any more. Our audience is the world now. It's mainstream North America for many of us. That's changed a lot of what we have to do. Also changing is the technology. Our sites are growing incredibly rapidly and therefore growing sort of out of our control as designers. To be able to hold on to the visual design that's happening across thousands, sometimes millions of pages. That control is something I want to talk about throughout the rest of the day here. Because I see a lot of designers grasping for control of things they may not necessarily be able to control on the web. But also losing control of things like how their designs are generated out of code and things like that. So that's what we're going to be looking at here. And I'm going to do so in these three sections. We're going to start talking about structure of web sites. Of very large-scale web sites. Some of the biggest sites on the web. The portal sites and how they have structured themselves in ways that integrate lots and lots of services to users. Then we're going to look over at that presentation and talk some about design patterns and template design and dynamic database web sites. And finally to behaviour and talking about self-aware content.

So let's start with structure. I'm going to talk about integration architecture. How to take a tremendous amount of content and services and futures and integrate it together in a consistent way that makes sense to people as they visit the site. I want to start with a fairly common site. You may be familiar with Amazon.com, and how they've managed to look forward to the future and scale and have a good overall architectural strategy over time. Because they started with a very good clean interesting information architecture. They used Tabs, a rendering of that architecture as a way of illustrating the facets of a single task that you can do. A great way to do an architecture.

So they have one thing you can do here and that is shop. And they have essentially the aisles of the store rendered as tabs. So you can shop for books, music, DVD's etc. That started to grow over time. And in fact started to get a little messy. From things that you can shop for, you can see that their business model eventually expanded into different ways of shopping. So from buying books, buying music, buying auctions. Well no, actually that's a kind of shopping. To Z shops. Well what the heck is that? That's now starting to add ambiguity into the architecture. Z shops, as it turns out, is one of the world's largest flea markets where you can set up your own store using the backend of Amazon's technology. But you'd never know that unless you sort of drilled in and looked. So a clean architecture expanding out a little bit farther than it could

conceptually, but also physically. After awhile they ran out of screen real estate and they had to start stacking the tabs. And they brought another row out and now what's the interaction model. Not making sense. In fact some people started to wonder what's going to happen in 5 or 6 years. Oh my god! You have this brain tumour growing up on top of Amazon here. Luckily this didn't happen. They actually switched to a hybrid of sort of a directory model like Yahoo has with a few featured tabs.

But here's another example of a very large web site that was built on top of a search engine. I worked for a couple of years on the HotBot search engine and I knew, as an architect and designer, that I would have to deal both with the needs of the user and the needs of a business, that the best possible search experience for a user is two pages. It's the front page and the results page. And if it's not on the results page then we've done something wrong and they have to keep looking and searching and it's not the best user experience. That doesn't map at all to a business goal based on page views and advertising where we want as many pages as possible. So we've seen sites like this, Yahoo, which started as a search engine and grew and grew and grew. Try to give the best possible results, but also provide as many other goods and services as possible to the users when they come to the site. Because a user really only sees this. That sort of subject-based area in the middle. And the only reason I highlighted it is because I know from watching user behaviour that when a user comes to search they will use some of the other interface elements on the page to help them formulate their query. But they won't necessarily click on things. So you have these big areas of the page that don't get a lot of traffic, but do actually help the user experience. But really all they're doing here is searching when in fact Yahoo is trying to get them to do all this other stuff. Yahoo want: shopping and auctions and yellow pages, companion, whatever that is. A little Yahoo that follows you around or something. News, sports, weather. Everything right. What they would like to do is say you've come here to look for something on the internet, I want to provide you a lot more too so you don't have to go away. So, again from a user experience, you can see it in two ways. The sort of evil profit driven corporate way. We got them to come here; we don't want to let go of them. We want to keep them inside of our universe as long as possible. Or, the more user experience point of view which is the more successful I can make users on my site with the tasks that they have, the more successful we'll be overall.

So they've done this and a number of other large portals have done this architecture through something I call the matrix. Which is, I'm afraid, nothing to do with Keanu Reeves. Which is in fact a literal matrix of subjects in which they have content and features, sort of sliced by what those features are. So we have you know, the typical auto business careers computers. Now this is sort of an exhaustive list of things that are available, but it is the most popular list. And this list would probably go on outside the auditorium and down into the hall when you look at everything that they have available. Intersected by things like a directory of web site reviews. A community of message boards. Instant messaging, chat, that sort of stuff. Commerce, news, etc., all the way down below the floor, making this sort of big space of information that they have but they want to communicate as quickly as possible to users so that they can build a mental model

of what's happening on the site as quickly as possible. I've seen sort of three different ways that these multiple hierarchies, all these different kinds of subjects and content and features, arranged hierarchically as it often is, is integrated across a bunch of different portal sites. So we have our directory and commerce and news. Yahoo does it with something I call impulse buy integration. This is very similar to going to the supermarket and you look at romaine lettuce and there just happens to be croutons and Caesar dressing in front of you. Those two things don't go in the produce section, but it is a good piece of integration. They're showing you, oh yes I need these things I'd better buy these. Yahoo is doing the same thing on their web site. So when you go to Yahoo's site…I probably should have used hockey examples, but I'm going to stick with baseball because it is getting close to the World Series now.

When you go here in the sports page, this is a page for the San Francisco Giants and on the Giants web page you have the stuff you would expect this site to have, as well as a variety of features that have been integrated as sort of impulse buys. Not only can you come here and read about what the Giants did in their last game and where they are in the standings, but you can interact with other Giant's fans. There's community stuff. Over here in another part of Yahoo's web site they've pulled links in for community, other web sites that you can navigate to and also some really interesting feature-based integration. So things like the Yahoo calendar. A place where you can go and keep your schedule on line. You can click a button and the Giant's schedule will be placed to your calendar. Great integration. Wow I came here to find out about the Giants, I'm going to set up a calendar as well because then I can track their schedule. So they're building their audience base. They're satisfying user needs that way. That's very, very interesting. It's also human right. The way Yahoo is able to do this is because, at one time at least, it had thousands of people in Santa Clara, California making these links. Making the connections. Searching through all the stuff Yahoo has and pointing this way and that. They did it by hand with human effort. That's why you can see things like this. In the Giant's community, there's an NHL chat, which is hockey. Oops, that's a mistake. Humans make mistakes. That's what happened there. So it's not necessarily the best possible way to do it, but it is an interesting way of showing things.

The Uber Tree is another one. It's taking all of those hierarchies and mashing them together into one big overall structure, which is a little different than pointing off in all these different directions the way Yahoo has done. In fact it's a strategy that Snap used. Snap eventually turned into NBCI. But Snap as a portal was doing things like bringing all these things together into one big structure and really, really branding where you were and orienting the user by putting a breadcrumb trail across the page. By bringing all this stuff together so that the web site reviews and the other categories and the chat and all the things that they have available are all at this node in the hierarchy. All brought together and then you surf through that node.

Let me show you an example of how this has worked at NBCI. Here's the front page. I'm going to go find that page on the San Francisco Giants again. So I'm going to click on sports and I'm going to drill into baseball. There it is. And you can see now across the top it says Home, Sports, Major League Baseball. That's the path I've taken so far. Into Baseball and then I'll click

on Teams, drilling in, Home Sports…oh wait…Scores and News Major League Baseball Teams. I think I still know where I am. And I scroll down and sure enough there's the San Francisco Giants. This is a great way of showing a massive amount of information. How it's all structured together and letting users drill down. The problem is most architecture works best when it's derived from the patterns inside of the data as opposed to developing an overall structure and cramming the data into it. Sort of going the other way. And that's what's happening here with a site this big. I've already seen an example of this. I thought I was going from Sports to Major League Baseball to the Giants, and in fact they've added something here called Scores and News. And that sort of threw me off a little bit. And so now I'm going to go back up to that sports page where I was before and oh wait. Now I'm in search and find Sports. Well that's okay because I see Baseball right here, so I'll go back into Baseball and oh I'm getting a little lost now. Where was Baseball again? Where are the Giants again? So, this architecture hasn't been derived from an anticipated user experience. It's been derived from an overall structure they want to impost on the data. It's a good way of providing a navigation system, but probably on a smaller scale. As you'll see with a lot of these they apply well to a smaller scale, but often get a little stuck in the massive.

So finally instead of a taxonomy I'm going to show you a stackotomy which is a little term I derived for picking all of those hierarchies and literally stacking them one on top of another. Whereas before we had crammed everything into one big tree, I'm going to keep all the trees of content separate and let users switch between different tasks that they have. The Go.com portal did a really good job of that. They had a very consistent interface across the whole site and a hierarchy for allowing users to drill down into information. But they provided a navigation system that said there's a lot of other hierarchies and you can drill through those hierarchies just by clicking this tab system. A little confusing.

But overall if I went down to that San Francisco Giants page, I would have tabs across the top so that I could switch over to Search and a directory of web sites. Communicate, shop, news

"Being able to essentially look at a cube and look at the different sides of that cube one you got to it. This is the stuff I call integration architecture where good classification systems can spawn both prediction and serendipity right?"

headlines. Things like that. I could click across the tabs and essentially show different facets of the node in the hierarchy that I was at. It's a pretty successful way of designing information.

I'm going to show you Lycos music, a site I worked on while I was at Lycos. We were pretty successful with a system like this. Being able to drill down into artists and then switch through to different kinds of content for each artist whether it was a biography or discography or a way to buy the CD's or download the music. Being able to essentially look at a cube and look at the

different sides of that cube once you got to it. This is the stuff I call integration architecture where good classification systems can spawn both prediction and serendipity. That happens in the real world. When you go to the library, once you figure out-maybe I'm dating myself a little bit-how the card catalogue works. Bear with me on this. They used to have little pieces of paper that showed you where to go in the library. Once you figured out how that system worked, everytime you came back to the library, and in fact at every other library you went to, they pretty much worked the same way. You could predict how the system would work. It also fostered serendipity. I would go look for a sea kayaking book and I would walk over to the shelf and ah, there's my book…and wow…look at all these other books! I'd start to find resources that I didn't know existed.

"Once you figured out how that system worked, every time you came back to the library and in fact every other library you went to, they pretty much worked the same way. You could predict how the system would work. It also fostered serendipity."

Well the same thing happens on web sites. Once you have an architecture that will illustrate to users how the system works, they can predict how to use it in the future. And they can also discover resources that they didn't know you had. Now, this seems to work fairly well in a portal, but how many of us are working in portals these days? There are plenty of opportunities to express these kinds of architectures in different web sites of different sizes, scales and scopes. Size for example. Does it matter? roadrunnersports.com is a web site that sells running shoes and apparel. Has an amazing directory of running shoes. Probably every running shoe that exists, in a variety of different categories. So you can drill down and you can find exactly the shoe you want. Then there's a link that says Message Boards. And if my wife is any indication, runners, she's a marathoner, are rabid about the shoes that they have. I think it's because it's one of the only pieces of gear you actually need for running. But they will talk and talk and talk and talk about their running shoes. And they do. So this message board is 18 pages of people talking about this shoe. Why you should buy it. Why you shouldn't buy it. Why it's wider than last year's model. Things like that. But they're not connected. When I click the message board link from the shoe page where I was about to buy it, I switched to the top of the message boards and I had to drill back down and find that shoe. Why weren't those integrated together? Well, they're on different servers. The person with the message boards hasn't had the right meeting with the person with the directory. Whatever. I don't know what it is, but that's where user needs should be deriving the architecture. And again, not the politics of the company or limitations of the technology.

So another strategy for doing integration on a smaller scale is to look externally as well as internally. Teevo is a little box that you plug into your television. And we all have so many channels now we need computers to watch tv for us. That's what it does. It watches tv for you and picks shows that it thinks you will like and lets you know so you can watch them. You can skip the commercials. Again, a rabid following of users. And what Teevo found was that all of these users were on this bulletin board system. And they were talking. There were thousands and thousands of messages there. Now when we think big corporation, we typically think well then the lawyers come in and so no you have to do this at our web site and we're going to shut you down. And we'll have a message board up in the third quarter of next year.

What Teevo did is work with the people who had the message board and gave them

collateral, identity and helped them design the site so that they could integrate it into their web site without having to have those capabilities. Something called ASP is out on the Internet. They're called application server providers. These are companies that build applications that they host and you can use. Some are free, some are advertising-based. But they make things like search, content integration, message boards and a variety of other services allowing you to integrate more content for your users without having to have those technological capabilities in-house.

Now, the future of this, from what I've seen, is a technology called XML, the extensive markup language. And a lot of people are talking about XML as the future of the web for a variety of different reasons. But at its core, it allows you to develop your own tag sets and allows groups of interested people to share those tag sets so we're no longer limited to just HTML. What that means is I can describe my content in a way that other people can understand and even other machines can understand. And we can start to share that content back and forth and start to integrate it much like the big portals do, but with other people relatively easily. So I will

"And we can start to share that content back and forth and start to integrate it much like the big portals do, but with other people relatively easily."

show you a little more of that a little later. But here's an example of this self describing content. This is an SML scheme that talks about how content for a news site had updates fairly regularly. So we can see that there are a variety of items on the page. Each item is made up of a title, an image, a description, a link. Much more descriptive and useable from an integration point of view than things like thought tag and table and even paragraphs. So by sharing vocabularies like this, it makes the integration between web sites or even internally inside web sites much, much easier. I'll show you an example of that a little bit later. But what I'd like to do now is move over from the structured point of view to the presentation.

On the web we talk a lot about keeping structure and presentation separate and how they come together in interesting ways. So we have our XML, our structured content that we're now going to share with other people or integrate internally. Now we have to start presenting this. Especially in the context, of a lot of change looking towards the future and designers needing more and more control over bigger and bigger web sites.

So how are we going to do that? I'm going to talk a little bit about design patterns. But first sort of a history lesson because historically the page design that we were doing on the web, and historically when I say in the olden days, that was gosh 5 or 6 years ago, designers would design HTML pages and put them on a server. And that's how we started with Wired. Actually we typically started in PhotoShop. We did that and we got to four or five thousand pages and all of a sudden, how do we redesign all of this stuff? It's a problem that organizations come across

all of the time. We have all this content on our web site and the only way to redesign it is to go back through and redesign every single page. And we don't have the resources to do that. So, we started turning our sights into dynamic web sites. And separating the structure from the presentation. I'll put my structured content in a database and I'll keep all of the presentation, the design of that, in templates and they'll come together only when the user asks for it. That way I can twiddle with the visual design however much I want and all the pages will update. So that sort of solved that problem to a certain extent. Designers found that they needed the flexibility of dynamic design to build these big huge web sites, but that rigidity of templates was often too much. I worked on Wired news. As a news web site we now have 15 to 20,000 stories that go back years and years and years. All of the story pages come off of one template. And that template means that every single page looks absolutely the same. So anything I change, changes it on all of them. There's no flexibility There's no way to do a story differently. When a big, big story happens, like something that happened last month (referring to Sept. 11th), something big happens, how do I reflect that on my web site when every single page looks identical? There was no way to do that. So, the benefits of course, the things that I've been mentioning, you can do more with fewer resources. You can separate the operation of a web site from the development. So for example this is how we used to publish content on our web site. Pretty standard process, author to editor to copy editor, production design, QA, post it on the web site. Afterwards we started doing things like this. Operations and development were separate so that production and design could work on a template system. And they could spend all their time making sites better and doing multiple sites at once. Whereas the operations of the site, the author, editor and copy editor had the ability to add content whenever they wanted and they could work separately and only intersect whenever features needed to be added or design needed to change.

To get past that dilemma of having a fairly flexible publishing system that allowed for no design flexibility, we started looking at an idea that was formed in the 70s by Christopher Alexander and some of his colleagues when he wrote the book A Pattern in Language. How many of you are familiar with this book? Anybody here? A few hands are up. This is a fascinating book. It's not the kind of book you sit down and read. It's 1,600 pages. It's a reference book. Hundreds and hundreds of what he calls patterns. And what they did was try to attack architecture, which was typically a very top-down design process, sort of designing for the portfolio or designing for the photograph. They decided instead to go from the bottom up. So, for example, they became experts in doorknobs. How do people open doors? What are the requirements for opening a door? What are the human factors for turning your wrist or moving things like this or sliding doors? What are the aesthetics? Is a doorknob different in a banquet hall versus a janitor's closet? Are there security implications? You can imagine all the things you'd have to learn to become an expert in doorknobs. No kidding. Wow that sounds like fun! But they did that for doorknobs, for window latches, for park benches, for roadways and intersections and on and on and on. And started to group these together in what they call patterns.

"Now patterns are really abstract solutions to very common problems in context."

Now patterns are really abstract solutions to very common problems in context. He called it "a" pattern language not "the pattern" language because he found the easiest solutions were applicable to the things that he was working on. And that's why it's so interesting to look at this kind of bottom-up pattern-based design because you can think of it more as a process than an end result. The process of looking at these little things, becoming experts in those things and grouping them together to solve design problems. It's applicable not only to architecture, but to computer science. Object-oriented programming is very similar to this. Little bits of code that do one thing very well and communicate with other bits of code to group together into bigger solutions. Corporate organization…how teams collaborate with one another to solve bigger problems within an organization. And of course web design. So when I was working at Lycos we were tasked with a design challenge of redesigning a web site that had 50 different products, many of which were from acquisitions, so they were completely different in any sense visually. Fifty different products that had inside them maybe two million pages. A big design challenge. And we knew there was no way we could do this top-down make the style guides, send it out to everybody and say, "Ok, go do it." That would have been of be unwieldy and unmanageable. In fact what we wanted to do was to make the right thing to do for designers at Lycos the easy thing. Make it easier than what they were doing today. And so trying to get designers to stop solving the same problems over and over again and to step up a level and think about user experience and architecture and how pieces of the web site go together to create that. The Lycos music web site was one of the first that we did to derive some of these patterns. We looked at dozens and hundreds of content types and tried to find similarities between them. Tried to become experts in the little things. The doorknobs on the web. So radio buttons, text fields how they go together to make search engines was one of the first things we did. Iterating and iterating. Testing with lots of users. Pulling best practices off the web to see which one of these solutions would be the best in our context to derive a pattern from it. Then we started eliminating them. Well you have to be able to switch between artist or song because you can't do both. So the checkboxes won't work. The pull-down menu will work, but from userability testing we see that users don't explore pull-down menus. They'll just type in an artist or a song regardless of what's there. The option box seems to work. It exposes bulk and let's you choose between one or the other, but doesn't fit aesthetically. It's too tall for the interface area that we wanted to fit this into. So the radio buttons seem to work the best. So we wrote it down as a pattern. The second one here, selecting between distinct search options, radio buttons are the most appropriate. Other ones were derived directly from user needs. Text input boxes should be based on the expected query length. Kind of common sense, but think how it applies in multiple contexts. So, in a stock site where you have four letter searches for ticker symbols, a search box only needs to be four characters long. But other sites like the music site it needs to be up to 25 or 26 characters long. We were trying to get the designers to think about these patterns. And then trying to encapsulate some of these patterns into code.

Now when Christopher Alexander was putting together his patterns in the world of architecture, he saw that there was a hierarchy of these patterns. If you're an expert in doorknobs, well you can become an expert in doors, where the knobs go in the doors. Where the doors go on the walls. How do those walls become rooms? Where do the rooms go in the buildings? And how are they organized? How are buildings then built up into communities all the way up to regions? He was talking about how far towns should be away from state capitals and stuff like that. Really, really sort of micro and macro thinking. Especially in the micro, there's almost an infinite number of decisions. There's so much that you have to get through, but at least you can start locally and work up.

So we had a hierarchy of patterns, from our radio buttons into forms. Those forms work as an instance of a search interface which goes into a page layout, into a navigation system for site architecture that's integrated with other sites. These patterns then grouped up into components. We tried to find the spot where it made the most sense for us to put our level of abstraction, as it were. Taking the smallest patterns, grouping them into instances like stories that change frequently, a need for user input. Searching. Another good one. And group the smallest patterns into these components which made sort of a Lego system for our web sites. We had all these components. Think of Lego. You can build almost anything out of Legos, but when you do they almost always look similar because Legos are very constrained. It's hard to do curves in Lego, but it's relatively easy to block things together. So we found that we could mix and match these components to create almost all of our sites. We had probably 16 or 18 total components with some standard pages elements that our designers could use to put together to make a variety of different web sites. Almost all of our web sites. So content that changes, item detailed, tabular data, user input, lists, that could stop trying to solve the same problems over and over again which all the designers across the whole network at Lycos had been doing. And they could start to focus on a higher level than architecture. I'm going to use this item detail to show my stock price. Well it turns out that item detail works well for stock prices. Also for things like the forecast in the weather section or for showing details about a new car in the automobile section. So these were reusable. And we could use them through the entire web site to get an overall look and feel that was very consistent. And the user benefit to consistency was just fantastic because they could leverage their knowledge from section to section. Once they figured out how a particular component worked, they would see that component throughout the site and feel pretty confident about, how they would interact with all the different web sites. And, it gave Lycos a consistent look and feel over the whole site.

Now, admittedly a fairly primitive look and feel, but at least it had more control than if we had just made standard page templates. Now we have these components that we can mix and match into almost anything. And it maps to a web technology thankfully. We were able to map this pretty well to something to cascading style sheets. CSS is a technology that allows for the true abstraction of content and its presentation by taking all the presentation out of HTML, leaving just the basics or basically that XML that we were talking about earlier, and taking all the

presentation, the visual design, the layout and abstracting it into separate files that we could keep pretty separate. This allowed our structured content to be shared with the XML vocabularies to do things like create components that were derived from standard XML vocabularies. Meaning now since it was a big portal that had lots and lots of partnerships, all we had to do was say give us this XML and it'll show up on the pages in the right components because we've already designed those components. Those components have been tested. The designers know how to use them and mix and match on the page. And all a designer really had to do was decide on the components they want based on the experience they were trying to provide for users through the research that they had done and wire it up to the content that was flowing in these XML vocabularies. So that's sort of the perfect world. Of course it doesn't always work like that but at least that's the direction we were heading. So much different than everybody with different content, different design and all of that work that had to happen every single time we wanted to add a new feature to the web site. So our self-describing content is now structured and has presentation. I'd like to take it one step farther and move over to behaviour now. How the sites interact and work. And how we can make the self-describing content a little more self aware and a little smarter. Add some intelligence to this content so that we can really start to step back as designers and really think about how our pages are going to interact when they go to our users' browsers. I'm going to talk about the self-aware content now in the context of behaviour.

We really are evolving to rule-based design. And this is sort of a tough thing for me to say to a room full of designers here. But that is really true. You can't tell what your pages are going to look like on the web. You just can't do it. You can attempt and you can approximate and you can get close, but you'll never ever fully get there. And that's something we really have to start to embrace. One of the hardest things that we've gone through with the designers that I've worked with at Wired for example, who came from the print magazine, trying to explain and embrace this notion that you can't possibly control the visual design of our pages in every single instance. But what we can do is set up constraints and make suggestions for how our content should change and adapt to the variety of environments it finds itself in. So this is really getting to the end of those pages. Remember you used to come to web sites and there'd be this disclaimer

"You can attempt and you can approximate and you can get close, but you'll never ever fully get there. And that's something we really have to start to embrace."

before you went in that said this page is best viewed with this browser at this resolution with these fonts installed. It would be like if I opened a store and said you can't come in unless you're over 6 feet tall or something like that.

And once again it's a system architecture. Rather than pushing the pixels around in PhotoShop…there's a time and place for prototyping and creating visual design in PhotoShop,

but at some point we move to the native language of the web and we start to let go of some of the control.

What I'm talking about here is called liquid pages. These are designs that react intelligently to the environments in which they're displayed. There are an exponential number of variables out there. There are fonts installed in machines that you have no control over. There are screen resolutions. There are different browsers and platforms. There are a variety of people with a number of different accessibility issues that again you have no control over. So people are going to be approaching your pages in ways you never even conceived. Especially as we move to things like mobile devices, interacting with the web in places like your kitchen, or in your car. No idea where are pages are going to start to show up.

My first example is guessing at typography. I would imagine some of you have done some print design. Print designers here? Yes, good. I speak to developers all the time, so often I get one or two hands. I'm going to assume most of you have done print design. So a typical situation would be you're doing an annual report or some project like that and you get the file the way you want it. Everything looks right on the screen and you send it off to be printed. And you say, "You know, whatever typeface you have go ahead and use that. I don't care. Fine with me."

"…embrace this notion that you can't possibly control the visual design of our pages in every single instance. But what we can do is set up constraints and make suggestions for how our content should change and adapt to the variety of environments it finds itself in."

Well that's what we do every day on the web. Look that's a font tag like that's one of the oldest web technologies for visual design way back in '95 I think the font tag came out. And there's no way of embedding fonts. In fact it's kind of a bad idea. They tried to in the 4.0 version of the browsers, to find a way to embed a typeface and send a page with the typeface and it slowed the pages down and there were encryption and intellectual property issues with typography that they weren't able to solve very well. So we're left with this. We're saying, "Hey you know what? This would be great if you could show this in Verdana. If you don't have Verdana I guess Arial would be alright. And if you're on a Mac you know, like Helvetica's not so bad either. And gosh just use a Sandsurf typefaces if that's all you've got." And that's web design. Like that's what we're doing. So this is the first step right in our recovery process to become web designers is to say, "Ok, I can't control typography. I can make guesses at it. I can approximate it, but I don't have the exact control." Now think about that example I painted a little bit earlier. So we went our annual report off and said "Yes, you pick the type faces at the press. I don't care, whatever. Oh, and whatever paper you have loaded up, just use that. That's fine. I don't care what size it is or anything." But we do that on the web all the time too. The table equals 100% of the width right. There's a classic example of liquid layout of a page that's been designed to expand or contract based on the size of the user's browser.

So let's look at a quick example of that. This is a friend of mine's web site. A wonderful web site called "statingtheobvious.com". It's actually "theobvious.com". And my friend Michael designed it with a very sort of sparse, clean, minimalist interface, an interface that feels like it's sort of been designed, no matter what size I had in my browser. So I can sort of expand it and contract it. I mean this is really simple stuff for web designers. Like, oh sure, some of the table cells have fixed widths. Other ones of relative widths. And in fact if I show you the table borders, that's the overall structure of the page. That's how the page has been laid out from a code perspective. These are the cells that are relative. But think about what he's doing in a broader sense. In a visual design sense, he's allowing the layout to adapt to whatever it's being displayed in. And I know there are plenty of issues about legibility and the line lengths. You want an alphabet and a half and this allows the user to decide what the line length is going to be. But at the same time there is an anticipated white space on the page. He's done a design that allows his design to sort of morph to what the user is expecting and anticipating based on again, their browser size.

So the final example I'd like to give of this is what I call magic headlines. This experiment that I was working on with a couple of designers over at Wired (before I left that company) in which we were really blurring the lines between visual design and programming and overall structure of a web site. Before I had got to Wired, I was working at a newspaper and one of the tasks I had at the newspaper was doing some of the layout of some of the pages including writing the headlines that fit on top of the columns of text. And one of the heuristics of newspaper design is that you show which columns on the page go with one another by placing a headline over them. You use some rules and things like that. But the headlines then have to sort of fit on top of those columns of text. And we would manipulate the size of the font as well as the language we would use. That's why you get crazy words in headlines sometimes because you're trying desperately to find a 4 letter equivalent for condominium. But we would manipulate the size based on the content that was below it. And I thought well can't we do that on the web sort of. Can I take some of that process that I used as a designer at a newspaper, and encapsulate it in code and send that code to the browser and let the browser try to approximate that design.

So let me show you a quick example. And again, this is fairly rough stuff, but it does illustrate the point. And I'm going to show you some code that happened. Because this really starts to talk about the change that's happening in web design. We're really moving away from that prototype in PhotoShop that I send over the wall to the techie people so they can build it. This is a relatively straightforward layout. I've got some advertising on the side. I've got a column again of text that changes size as I open and close the browser. Relatively straightforward. I wanted to see if I could get the headline to fit on top of the text that I had. So what I did was added just a little bit of code to make the text feel a little bit more like it goes with the headlines. So now when I expand or close the page, the headline changes right? And to do that was really straightforward. Here's the code. I've got it fairly big. I added a little function into my page, writing a little bit of script here that does two things. The first thing it does is ask the browser

how big you are. It says how wide are you on the screen. So as the user opens and closes it, it asks over and over again, how big are you? How big are you? And the browser tells the script that I'm "X" pixels wide. I take that number, divide it by a constant. In this case 40, to derive a size that fit's pretty well on my web page. So I've got 800 pixels wide. I ask the browser. It says I'm 800 pixels. I divide by 40 I get 20. The second line says the font size style of the headline should be my size which is based on that number. So I end up with 20. I set the font size to be 20 on my headline. All three corners of that triangle. I have structured content.

I have a headline. I'm asking for the headline. And if we were to scroll done into the code, you would see that my markup said that this piece of text here is a headline. I'm taking that structured piece of code and I'm setting the visual design, the presentation of that, based on the behaviour. I'm running some code that sends the font size to a structured element. So all three corners of the triangle are being manipulated here.

But there are problems with this right. There are some things that I want to have control over that I currently don't. So for example, I can make this so small as to be absurd right? Like there's 1 point type. Well that's crazy. I can't have 1 point type. I want to set some constraints. And when I think back to the style guide I had at the newspaper, our headlines never got smaller than say, 14 point and never bigger than 50 point. Unless we went to war or something like that. And admittedly, this I realized headline size is a direct correlation to the importance of a story in a newspaper. I'm using this to illustrate a different point. There's editorial value in visual design clearly, but this makes for a good example. So, this new version now, if I show you the source of this, adds a little bit more code. And it adds code that was actually a step in the process back when I was at the newspaper. This is that checking the style guide to see what my constraints are for font size in my headlines. So now I say I'd like you to again ask the browser how big it is. Get that value back, divide by 40 to get down to a size that I can use. But then check is that result smaller than 14 points. If my size is less than 14, well then my size should be 14. Like don't go any smaller than 14 points. Don't go any bigger than 50 points. And then set the font size.

So what that does is I can't get any bigger than this, so I can't show you constraining big, but it will stop at 14 now so my headline at least stays legible no matter what size I am. So I've added some constraint. But if you think about it, I've got more variables out there that I'm not accounting for.

For example, not only do I not know the width of this browser window, that's why I'm asking for it. I don't know what that headline is going be. This is design in a template in a publishing system. So this design is happening 30, 40, 50 times a day as they publish new content every time the user sees it with a new story. So I don't even know what headline is going to be there. So that divide by 40 would only sort of work for headlines that are about this long… that have this many characters. So my next version does again another step of the design process encoded into the script. I may have short ones that need to be bigger. I may have long ones that need to be smaller. So what I do is again, try to make my headline a little bit smarter. So when you're at the user's browser, first again look how big it is. Then what 'd like you to do, actually

the first line of this function says, what I'd like you to do is go look at the headline and find the content inside that headline. Then the second line says I would like you to count how many characters are in that headline and report that back to me. Then, find out how big the browser is. Then do some math that takes the size of the browser with the number of characters that are in the headline, divide those together, give me a good font size, tweak it a little bit for the visual design on my page. That's the times 1.05 at the end. Then check the constraints. Make sure it's not too big and not too small. Then display it. All of those are steps along the path that I used to do as a visual designer.

So what that means is that this next step in web design is all about manipulating the style of structured elements. This is about web design that's being generated from code. It's getting more control back in a place where we thought we were losing a lot of control as designers. But it's a step, sort of technologically. Now I have designers that ask me all the time, "Like what technologies do I have to learn. Oh my gosh there's so many of them." And the answer is well pretty much have a working knowledge of everything that's going on out there so that you can sort of incorporate that into your visual design. Now that means you don't have to be somebody who's out there writing sequel queries against a big data base and things like that. But you do need to know the capabilities and limitations of the medium in which you're working. I worked at Wired magazine for years with the creative directors Barbara Coor and John Plunket who are amazing visual designers. And one of the things I learned from them, is that part of their design talent came from their deep technical knowledge of how design worked. And they could on it forever and they did. About things like dot gain, especially dot gain with lurid fluorescent pinks. they knew all about those things. But how the ink would relate to the paper and type faces. And they had their own type faces designed. And on and on. Very, very technical designers in the world of print. And all I'm suggesting is that we become the same when we're working in this

"…have a working knowledge of everything that's going on out there so that you can sort of incorporate that into your visual design."

medium. This is again rule based design. This is setting constraints, adding behaviour and sort of letting our content go be free in the world and doing the right thing. It's like being parents and letting our kids go off. And you know you gave them the tools so that they would do the right things in the right situations. Well that's what we're doing here with our little pieces of content. And I gave you a really simple example of headline that gets bigger or smaller based on how big it is or how big the page is. But think of all the elements that were on that page. The advertising, the text length. The font that was being used for reading and the navigation system. All of those things could be inter-related with one another and all have levels of intelligence as well. And really, what I'm doing is using technology to encode the design process. I'm looking back at how

we used to do things, from my own experience in the print world, and trying to figure out how I can incorporate that into a design that's going to work in the virtual world.

"I'm looking back at how we used to do things, from my own experience in the print world, and trying to figure out how I can incorporate that into a design that's going to work in the virtual world."

So that sort of takes us all the way around here. One other thing I wanted to point out with this structure presentation and behaviour, is that these technologies map pretty well to the product development process with people. With engineers, architects and designers. So this is not only a technological collaboration, but it's a social collaboration as well. And I found, from my own experience working with designers and architects and engineers, that you have to own the corner of your triangle. You have to master those skills, but you also have to be as broad as possible to look towards the other corners of the triangle and understand those capabilities and limitations. Again the best designers I've worked with on the web are the ones that could write Java script, could get their hands dirty in the template language of a web page. Likewise the best engineers I've ever worked with are the ones that an understanding of the sensibilities of visual design and architecture. So it sort of works in all directions. And it's of course very limited to the scope of prototyping and developing. Marketing fits in here. User research fits into this. Business goals and sales, selling advertising on the web and all the drivers for success in the commercial world fit into this diagram as well. But in the instance in which I'm talking about with these design principles, engineering design and architecture are sort of, I see, as a nice neat social collaboration.

So overall, this planning for the future, this structured integration that I showed you, self describing content leading to automated integration I think is a big place. We see lots of people doing contents indication and sharing web services with one another and building business models out of that. Design patterns, generic solutions to abstract problems as the next step in template design and dynamic data based driven web sites with more control for designers on top of the flexibility of large scale web site design.

Self aware content in coding that design process and script, really, really relying on the collaboration. Really understanding the capabilities and limitations of our medium. And finally, collaboration both technologically and socially. Really the keys to success there. If you'd like to have a look at some of that code, it's all on my web site. You can find it here at this URL "veen.com/presentations", and let me hear from you any feedback. Or, if you like, look at my code and make it better and send it back to me. That'd be great. That's my email address. Thank you so much.