№ 1.
COMO CHEGAMOS AQUI?

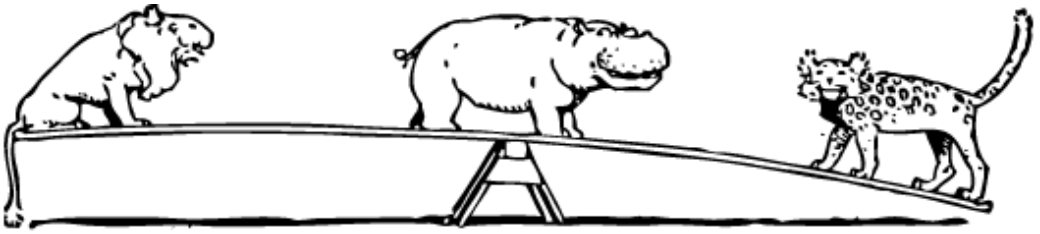
exibir índice analítico

MERGULHANDO

Recentemente, me deparei com uma citação de um desenvolvedor da Mozilla sobre a tensão que existe ao criar padrões:

> Implementações e especificações devem dançar delicadamente juntos. Você não quer que uma implementação ocorra antes que a especificação esteja completa, pois as pessoas dependem de detalhes da implementação e isso faz parte da especificação. Entretanto, você também não quer que a especificação esteja completa antes da implementação e dos testes feitos pelo autor com esta implementação, pois você precisa de um feedback. É inevitável que haja uma tensão aqui, mas isso é só uma pequena confusão.

Mantenha esta citação na sua mente, e me deixe explicar como a HTML5 surgiu.



MIME TYPES

Este livro é sobre HTML5, não é sobre as versões anteriores da HTML e não é sobre qualquer versão de XHTML. Mas para entender a história da HTML5 e as motivações por trás dela, você precisa entender primeiro de alguns detalhes técnicos. Especificamente, MIME types.

Toda vez que o seu navegador chama uma página, o servidor web envia "cabeçalhos(headers)" antes de enviar a marcação de página em si. Esses cabeçalhos normalmente são invisíveis, mas existem ferramentas de desenvolvimento que os tornam visíveis caso necessário. Estes cabeçalhos são importantes, pois eles informam ao navegador como interpretar a marcação da página. A parte mais importante do cabeçalho é chamada de `Content-Type`, e se parece com isso:

```
Content-Type: text/html
```

"`text/html`" é chamado de "content type" (tipo de conteúdo) ou "MIME type" da página. Este cabeçalho é a **única** coisa que determina o que um recurso realmente é, e portanto como ele deve ser renderizado. Imagens tem seu próprio MIME types (`image/jpeg` para imagens JPEG, `image/png` para imagens PNG, e por ai vai). Arquivos JavaScript tem seu próprio MIME type. Folhas de estilo CSS tem seu próprio MIME type. Tudo tem seu próprio MIME type. A web funciona por causa dos MIME types.

Claro que a realidade é um pouco mais complicada que apenas isso. A primeira geração de servidores web (e eu estou falando de servidores web de 1993) não enviavam o cabeçalho `Content-Type`, pois ele ainda não existia. (Ele não foi inventado antes de 1994.) Por causa da compatibilidade até 1993, alguns dos navegadores populares da época ignoravam o cabeçalho

`Content-Type` em alguns momentos. (Isto é chamado de "content sniffing.") Mas como regra geral, tudo que é procurado na web — páginas HTML, imagens, scripts, videos, PDFs, tudo que tem uma URL — é exibido com seu MIME type específico no cabeçalho `Content-Type`.

Guarde isto na sua cartola. Nós voltaremos a esse assunto.

❧

UMA LONGA DIGRESSÃO ENTRE OS PADRÕES QUE FORAM FEITOS

Porque existe um elemento `<img>`? Esta não é uma pergunta que você vê todo dia. Obviamente *alguém* criou ela. Estas coisas não aparecem do nada. Todo elemento, todo atributo, toda funcionalidade da HTML que você já usou alguma vez foi criada por alguém, que decidiu como ela deveria funcionar, e escreveu todo o código. Estas pessoas não são deuses ou invencíveis. São apenas pessoas. Pessoas inteligentes, com certeza. Mas apenas pessoas.

Uma das grandes coisas sobre padrões "abertos" é que você pode voltar no tempo e poder responder esse tipo de perguntas. As discussões ocorrem em listas de emails, que geralmente são arquivadas e podem ser procuradas depois. Então eu decidi fazer um pouco de "arqueologia de email" para tentar responder a seguinte pergunta, "Porque nós temos um elemento `<img>`?" E eu tive que voltar para antes que uma organização chamada World Wide Web Consortium (W3C) existisse. Eu voltei aos primeiros dias da web, quando você ainda podia contar o número de servidores web com as duas mãos e talvez alguns dedos do pé.

(Tem alguns erros tipográficos nas citações abaixo. Eu decidi deixar eles intactos para manter a precisão histórica)

Em 25 de Fevereiro de 1993 Marc Andreessen escreveu:

> Eu gostaria de propor uma nova tag HTML:
>
> IMG
>
> O argumento obrigatório é: `SRC="url"`.
>
> Ela serve para apontar um arquivo bitmap ou pixmap para o navegador interpretar como uma imagem no meio do texto no ponto de ocorrência da tag.
>
> Como exemplo temos:
>
> ```
> <IMG SRC="file://foobar.com/foo/bar/blargh.xbm">
> ```
>
> (Não há uma tag de fechamento; é uma tag standalone.)
>
> Esta tag pode ser usada como link como qualquer outra coisa; quando isto acontece, ela vira um ícone sensitivo a ativação exatamente como um texto de âncora comum.
>
> Navegadores devem ter flexibilidade para os formatos de imagem que irão suportar. Xbm e Xpm são bons para suportar, por exemplo. Se um navegador não conseguir renderizar o formato dado, ele pode fazer o que quiser (No X Mosaic irá aparecer um bitmap padrão como substituto da imagem).
>
> Isso é uma função necessária no X Mosaic; nós temos isso funcionando, e nós iremos usar internamente pelo menos. Eu estou bastante aberto a sugestões de como isso pode ser feito com HTML; se você tem uma ideia melhor que a minha, por favor me informe. Eu sei que é difícil lidar com formatos de imagem, mas eu não vejo uma alternativa além da que eu acabei de falar "deixe o navegador fazer o que ele consegue" e esperar que a solução perfeita apareça (MIME, um dia, quem sabe)

Xbm e Xpm eram formatos gráficos populares no Unix.

"Mosaic" foi um dos primeiros navegadores web. ("X Mosaic" era a versão que rodava no Unix.) Quando ele mandou essa mensagem no início de 1993, Marc Andreessen ainda não havia fundado a companhia que fez ele famoso, Mosaic Communications Corporation, nem tinha começado a trabalhar no principal produto da empresa: "Mosaic Netscape." (Você deve conhecer melhor pelos nomes posteriores, "Netscape Corporation" e "Netscape Navigator.")

“MIME, um dia, quem sabe” é uma referência a negociação de conteúdo, que é uma funcionalidade do HTTP onde um cliente (como um navegador web) diz ao servidor (como um servidor web) qual tipo de recursos ele suporta (como `image/jpeg`) para que então o servidor retorne para o cliente no seu formato preferido O protocolo HTTP original foi definido em 1991 (a única versão que havia implementada em Fevereiro de 1993) não possuía nenhuma maneira dos clientes dizerem aos servidores que tipo de imagens eram suportadas, o que levou ao dilema de design que Marc encontrou.

Algumas horas depois, Tony Johnson respondeu:

> Eu tenho algo bastante similar no Midas 2.0 (em uso aqui no SLAC, e haverá um release público a qualquer semana), exceto que todos os nomes são diferentes, e tem um argumento extra `NAME="name"`. Tem quase exatamente a mesma funcionalidade que você propôs na tag `IMG`, como por exemplo:
>
> ```
> <ICON name="NoEntry" href="http://note/foo/bar/NoEntry.xbm">
> ```
>
> A ideia do parâmetro name é de permitir ao navegador o uso de imagens internas. Se o nome for o mesmo que uma imagem interna, ele usará ela ao invés de buscar a imagem. O nome pode atuar também como uma dica para quando o navegador rodar no terminal como um tipo de símbolo para substituir a imagem.
>
> Eu não ligo muito sobre os nomes dos parâmetros ou das tags, mas eles devem ser sensíveis se forem utilizados para mais de uma coisa. Eu não ligo muito para abreviações, ou seja porque não `IMAGE=` e `SOURCE=`. Eu prefiro `ICON` uma vez que é menos que `IMAGE`, mas talvez `ICON` seja uma palavra sobrecarregada, não?

Midas foi um outro dos primeiros navegadores, contemporâneo ao X Moisac. Ele era multiplataforma; funcionava tanto no Unix quando no VMS. "SLAC" refere-se ao Stanford Linear Accelerator Center, agora SLAC National Accelerator Laboratory, foi o lugar que hospedou o primeiro servidor web dos Estados Unidos (na verdade o primeiro servidor web fora da Europa). Quando Tony escreveu essa mensagem, SLAC era um ancestral da WWW, e hospedava cinco páginas por gritantes 441 dias.

Tony continuou:

> Enquanto ainda estamos no assunto de novas tags, eu tenho outra, de algum modo parecida, tag que eu gostaria de suportar no Midas 2.0. A princípio ela é:
>
> ```
> <INCLUDE HREF="...">
> ```
>
> A intenção aqui é de poder colocar um segundo documento dentro do primeiro documento no ponto que a tag aparece. A princípio o documento referenciado pode ser qualquer coisa, mas o objetivo principal é de permitir imagens (nesse caso de tamanho arbitrário) para ser incluída dentro de documentos. Novamente a intenção é de que quando o HTTP2 surgir, o formato do documento incluído possa ser negociado separadamente.

"HTTP2" é uma referência ao HTTP básico definido em 1992. Neste ponto, no início de 1993, ele ainda não estava largamente implementado. O rascunho conhecido HTTP2 foi eventualmente padronizado e implementado como "HTTP 1.0" (embora não por mais três anos). HTTP 1.0 não incluía a requisição de cabeçalhos para negociação de conteúdo, a.k.a. "MIME, um dia, quem sabe."

Tony continuou:

> Uma alternativa que considerei foi:
>
> ```
> <A HREF="..." INCLUDE>See photo</A>
> ```
>
> Eu não gosto muito de colocar mais funcionalidade na tag `<A>`, mas a idéia aqui é de manter a compatibilidade entre os navegadores que não podem honrar com o parâmetro `INCLUDE`. A intenção é que os navegadores que entendam `INCLUDE`, alterem o texto do

> link (nesse caso "See photo") com o documento incluído (foto), enquanto os navegadores antigos ou burros ignorem completamente a tag `INCLUDE`.

Esta proposta nunca foi implementada, no entanto a ideia de colocar um texto quando a imagem não é encontrada, é uma importante técnica de acessibilidade esquecida pela proposta inicial da tag `<IMG` de Marc. Anos depois esse atributo foi incluído como a tag `<img alt>`, que o Netscape mostrava erroneamente o nome quando se colocava o mouse em cima da imagem.

Algumas horas depois que Tony enviou aquela mensagem, Tim Berners-Lee respondeu:

> Eu imaginei que figuras poderiam ser representadas como
>
> ```
> <a name=fig1 href="fghjkdfghj" REL="EMBED, PRESENT">Figure </a>
> ```
>
> onde a relação entre os valores significaria
>
> ```
> EMBED    Coloque isso quando apresentar
> PRESENT  Mostre isso sempre que o documento fonte for apresentado
> ```
>
> Veja que você pode ter diversas combinações disso, e se um navegador não suporta nenhuma, ele não quebra
>
> [Eu] vejo que usando esse método para ícones selecionáveis significam links. Hmmm. Mas eu não quero nenhuma tag especial

Esta proposta nunca foi implementada, mas o atributo `rel` continua por aí.

Jim Davis adicionou:

> Seria legal se houvesse um content type específico, por exemplo:
>
> ```
> <IMG HREF="http://nsa.gov/pub/sounds/gorby.au" CONTENT-TYPE=audio/basic>
> ```
>
> Mas eu estou completamente disposto a viver com a necessidade de especificar o content type pela extensão do arquivo.

Esta proposta nunca foi implementada, mas o Netscape posteriormente incluiu o suporte de inserir áudio e vídeo com o elemento `<embed>`.

Jay C. Weber perguntou:

> Enquanto imagens estão no topo da minha lista de tipos de mídias num navegador WWW, eu não acho que nós devemos incluir especificações idiossincráticas para cada tipo de mídia. O que aconteceu com o entusiasmo de usar um mecanismo de MIME type?

Marc Andreessen respondeu:

> Isto não é um substituto para o futuro uso do MIME como um padrão do mecanismo; Isso apenas dá a implementação necessária e simples da funcionalidade independente do MIME.

Jay C. Weber respondeu:

> Vamos temporariamente esquecer do MIME, se é o que está atrapalhando. Minha objeção é a discussão sobre "como nós iremos incluir o suporte a imagens" e não sobre "como nós vamos suportar os diversos problemas nas diversas mídias"
>
> De outra maneira, semana que vem alguém vai sugerir ‘vamos colocar uma nova tag `<AUD SRC="file://foobar.com/foo/bar/blargh.snd">`‘ para áudio.
>
> Isto é muito custo no lugar de usar algo que generalize.

Com o benefício de termos uma retrospectiva, parece que as preocupações de Jay estavam bem fundamentadas. Levou mais de uma semana, mas o HTML5 finalmente inclui os novos elementos `<video>` e `<audio>`.

Respondendo a mensagem original de Jay, Dave Raggett disse:

> Verdade! Eu gostaria de considerar toda a gama de possibilidade de tipo de arte em imagens, além da possibilidade de negociação do formato. A mensagem de Tim sobre

> áreas clicáveis nas imagens também são importantes.

Mais tarde em 1993, Dave Raggett propôs HTML+ como uma evolução do padrão HTML. Esta proposta nunca foi implementada, e foi substituída pela HTML 2.0. HTML 2.0 foi uma "retrospectiva", o que significa que formalizou as funcionalidades já em uso "Essa especificação reúne, esclarece e formaliza o conjunto de funcionalidades que grosseiramente corresponde as capacidades do HTML que estavam em uso em Junho de 1994."

Dave mais tarde escreveu a HTML 3.0, baseado no rascunho feito por eles do HTML+. Fora isso existia a referência da W3C do Arena, HTML 3.0 nunca foi implementado, e foi substituído pelo HTML 3.2, outra "retrospectiva":"HTML 3.2 incluiu largamente outras funcionalidades como tabelas, applets e textos ao redor de imagens, enquanto mantinha a retro-compatibilidade com o padrão existente: HTML 2.0."

Dave mais tarde foi co-autor do desenvolvimento da HTML 4.0, desenvolveu a HTML Tidy, e ajudou com as especificações do XHTML, XForms, MathML, e outras especificações modernas da W3C.

Voltando para 1993, Marc respondeu a Dave:

> Na verdade, talvez nós devêssemos pensar numa linguagem procedural genérica para gráficos que com ela nós possamos incluir hyperlinks aleatórios anexados a ícones, imagens, ou texto, ou qualquer coisa. Alguém vê as capacidades Intermedia disso?

Intermedia foi um projeto de uso de hipertexto da Brown University. Ela foi desenvolvida de 1985 até 1991 e rodava no A/UX, um sistema operacional Unix-like utilizado nos primeiros computadores Macintosh.

A ideia de "uma linguagem procedural genérica para gráficos" foi eventualmente implementada. Navegadores modernos suportam tanto SVG (marcação declarativa com scripts embutidos) e `<canvas>` (uma API procedural e direta para gráficos), mesmo que começasse como uma extensão proprietária antes de começar a ser "revista" pela WHATWG.

Bill Janssen respondeu:

> Outros sistemas para olharmos que tem alguma noção disso (bastante válida) são Andrew e Slate. Andrew foi feito com _insets_, e cada um deles tem um tipo interessante, como texto, bitmap, desenhos, animações, mensagens, planilhas, etc. A noção de inclusão arbitrária e recursiva está presente, então um inset de qualquer tipo pode ser incluído em qualquer lugar que suporte incorporação. Por exemplo, um inset pode ser incluído em qualquer ponto do texto de um widget de texto, ou em qualquer área retangular de um widget de desenho ou em qualquer célula de uma planilha.

"Andrew" é uma referência a Andrew User Interface System (nessa época era conhecida apenas como Andrew Project).

Ao mesmo tempo, Thomas Fine teve uma ideia diferente:

> Esta é a minha opinião. A melhor maneira de usar imagens na WWW é utilizando MIME. Eu tenho certeza que postscript é um tipo suportado no MIME, e que lida tranquilamente com a mistura de texto e imagens.
>
> Mas isto ainda não é clicável, você diz? Sim você está certo. Eu suspeito que já tenha uma resposta para isso ser exibido utilizando display postscript. Mesmo que incluir isto ao padrão postscript seja trivial. Definir um comando de âncora que especifica a URL e o uso do caminho atual como uma região próxima ao botão. Desde que o postscript lide bem com os caminhos, isto faz com que formatos de botões aleatórios sejam triviais.

Display Postscript era uma tecnologia de renderização na tela co-desenvolvida pela Adobe e NeXT.

Esta proposta nunca foi implementada, mas a idéia de um jeito melhor de consertar HTML e substituí-lo por algo completamente diferente aparece de tempos em tempos.

Tim Berners-Lee, 2 de Março de 1993:

> HTTP2 permite que um documento contenha qualquer tipo que o usuário diga que pode lidar, não apenas os MIME types registrados. Então pode ser feito um experimento. Sim eu acho que tem algum caso que funciona postscript com hipertexto, eu não entendo de display postscript o suficiente. Eu sei que a Adobe está tentando estabelecer seu próprio

> tipo de postscript "PDF" que terá links e poderá ser lido pelos leitores proprietários deles.
>
> Eu penso que uma linguagem genérica para links entre camadas (baseados em Hytime?) pode permitir padrões hipertexto e imagens/video que envolva-os separadamente, que pode ajudar ambos.
>
> Deixem que a tag `IMG` possa ser `INCLUDE` (incluída) e deixe ela se referir a um tipo arbitrário de documento. Ou `EMBED` se `INCLUDE` parece com um include do cpp e que as pessoas possam esperar por um código fonte SGML para ser interpretado — o que não é a intenção.

HyTime foi um dos primeiros sistemas de documentos hipertexto baseado em SGML. Ele teve uma grande importância nas primeiras discussões sobre HTML e posteriormente sobre XML.

A proposta de Tim para uma tag `<INCLUDE` nunca foi implementada, no entanto pode-se ver ecos dela nos elementos `<object>`, `<embed>`, e `<iframe>`.

Finalmente em 12 de Março de 1993, Marc Andreessen revisitou a thread:

> Voltando a thread sobre inclusão de imagens — eu estou chegando perto de lançar o Mosaic v0.10, que irá suportar a inclusão de GIF e imagens/bitmaps XBM, como eu disse anteriormente. ...
>
> Nós não estamos preparados para suportar as tags `INCLUDE`/`EMBED` nesse momento. ... Então nós provavelmente iremos usar as tags `<IMG SRC="url">` (não a tag `ICON`, dado que nem todas as imagens podem ser propriamente chamadas de ícones). Por enquanto as imagens incluídas não deverão possuir um content-type específico; futuramente, nós planejamos dar suporte a isto (junto com a adaptação do MIME). Na verdade, a rotina de leitura de imagens que nós estamos utilizando lida com o formato da imagem no momento de renderizar, então a extensão do arquivo não é tão relevante.

UMA LINHA CONTÍNUA

Eu tenho um fascínio incrível por todos os detalhes dessa conversa de quase 17 anos de idade que levou a criação de um elemento HTML que é utilizado em praticamente todas as páginas da internet. Considere que:

- HTTP continua existindo. HTTP evolui com sucesso de 0.9 para 1.0 e posteriormente para 1.1. E continua evoluindo.
- HTML continua existindo. O formato de dados rudimentar — ele sequer suportava imagens em linha! — evoluiu com sucesso para 2.0, 3.2, 4.0. HTML é uma linha contínua. Uma linha torcida, cheia de nós, embolada, com certeza. Existem diversos "branches mortos" na árvore evolutiva, lugares onde pensamentos estavam a frente das próprias pessoas (e na frente dos autores e desenvolvedores). Mas continua. Aqui estamos, em 2010, e as páginas da web de 1990 continuam sendo exibidas corretamente nos navegadores modernos. Eu acabei de carregar uma no navegador em estado-da-arte do meu celular Android e eu nem recebi uma mensagem dizendo "por favor aguarde enquanto estamos importando formatos legados..."
- HTML sempre será uma conversa entre marcações de navegadores, autores, padrões, e outras pessoas que simplesmente apareceram e gostaram de conversar sobre símbolos de maior e menor. A maioria das versões bem sucedidas da HTML foram "retrospectivas," pegando tudo que existia e tentando empurrar isso para direção certa. Qualquer um que diga a você que a HTML deveria continuar "pura" (presumivelmente ignorando os marcadores dos navegadores, ou ignorando os autores ou ambos) está simplesmente mal informado. A HTML nunca foi pura, e todas as tentativas de purificá-lo foram falhas incríveis, vistas apenas pelas tentativas de substituí-lo.
- Nenhum dos navegadores de 1993 continua existindo de uma forma reconhecível. O Netscape Navigator foi abandonado em 1998 e re-escrito a partir do rascunho para criar a Suite Mozilla, que foi subdivida para criar o Firefox. O Internet Explorer teve diversos "começos" no "Microsoft Plus! for Windows 95," que foi empacotado junto com alguns temas para área de

trabalho e um jogo de pinball. (Mas é claro que também podem ser encontradas referências anteriores a este navegador).

- Alguns dos sistemas operacionais de 1993 continuam existindo, mas nenhum deles é relevante para internet moderna. A maioria das pessoas que "usa" a internet faz em um PC rodando Windows 2000 ou superior, um Mac rodando Mac OS X, um PC rodando algum sabor de Linux, ou um smartphone como um iPhone. Em 1993, o Windows estava na versão 3.1 (e competindo com OS/2), Macs rodavam System 7 e o Linux era distribuído pela Usenet. (Quer se divertir um pouco? Encontre um dinossauro da internet e sussurre "Trumpet Winsock" ou "MacPPP.")
- Algumas das mesmas *pessoas* continuam por ai e continuam envolvidas no que nós simplesmente chamamos de "web standards (padrões da internet)." Isso após quase 20 anos. Alguns destes estavam envolvidos com os predecessores da HTML, na década de 80 e anteriores.
- Falando dos antecessores... Com a popularidade adquirida da HTML e da internet, é fácil esquecer os formatos e sistemas modernos envolvidos na criação deles. Andrew? Intermedia? HyTime? E o HyTime não foi somente um projeto de pesquisa acadêmico; ele foi um padrão ISO. Ele foi aprovado para uso militar. Ele era um Grande Negócio. E você pode ler sobre ele por você mesmo... na página HTML dele, no seu navegador.

Mas nenhuma dessas coisas responde a pergunta original: por que temos um elemento `<img>`? Por que não um elemento `<icon>`? Ou ainda um elemento `<include>`? Por que não temos um link com um atributo `include`, ou alguma combinação de valores em `rel`? Por que um elemento `<img>`? Simples, porque Marc Andreessen enviou um código com ele e código enviado ganha.

Isso não quer dizer que *todos* códigos enviados ganham; afinal, Andrew e Intermedia e HyTime enviaram seus códigos também. Código é necessário, mas não o suficiente para o sucesso. E eu *com certeza* não quis dizer que enviar código antes de um padrão irá produzir a melhor solução. O elemento `<img>` de Marc não funcionava com diversos formatos comuns de figuras; ele não definia como o texto ficaria ao redor da imagem; ele não suportava alternativas de texto ou de substituição de conteúdo em caso de falhas em navegadores antigos. E 17 anos depois, continuamos lidando com o sniffing de conteúdo, e continuam existindo diversas vulnerabilidades loucas. E você tem como voltar atrás 17 anos e ver a Grande Guerra dos navegadores e voltar para 25 de Fevereiro de 1993 quando Marc Andreenssen simplesmente comentou, "MIME, um dia, quem sabe," e enviou seu código de todo modo.

Ganha aquele que é enviado.

UMA LINHA DO TEMPO DO DESENVOLVIMENTO DA HTML DE 1997 À 2004

Em Dezembro de 1997, a World Wide Web Consortium (W3C) publicou a HTML 4.0 e provavelmente fechou o grupo de trabalho da HTML. Menos de dois meses depois, um grupo de trabalho separado da W3C publicou o XML 1.0. Três meses depois apenas, as pessoas que faziam parte da W3C deram um workshop chamado: "Moldando o futuro da HTML" para responder a questão: "A W3C desistiu da HTML?" Esta foi a resposta:

> Nas conversas foi concordado que seria difícil ocorrer uma futura extensão da HTML, como a conversão da 4.0 para uma aplicação XML. Foi proposto quebrar com estas restrições e fazer um novo começo para a nova geração da HTML baseada em um conjuto de tags XML.

A W3C recriou o grupo de trabalho da HTML para criação do seu "conjunto de tags XML." O primeiro passo deles, em dezembro de 1998, foi um rascunho de uma especificação interna que simplesmente reformulou a HTML no XML sem adicionar nenhum novo elemento ou atributo. Esta especificação posteriormente ficou conhecida como "XHTML 1.0." Ela definiu um novo MIME type para documentos XHTML: `application/html+xml`. Entretando, para realizar a migração das atuais 4 páginas da HTML existentes, segundo o Apêndice C, resume nas "principais guias de design para os autores que desejam processar documentos XHTML nos agentes HTML existentes." O apêndice C diz também que é permitido ao autor chamar documentos "XHTML", mas utilizar ainda o MIME type `text/html`.

O próximo objetivo deles foram os formulários web. Em agosto de 1999, o mesmo grupo de trabalho da HTML publicou o primeiro rascunho da XHTML Extended Forms. Eles colocaram as

expectativas no primeiro parágrafo:

> Depois de uma cuidadosa consideração, o grupo de trabalho da HTML decidiu que as metas para a nova geração dos formulários são incompatíveis com a retro-compatibilidade com os navegadores desenvolvidos para suportar as versões antigas da HTML. É nosso objetivo criar um novo modelo de formulários do zero ("XHTML Extended Forms") baseado em um conjunto bem definido de requisitos. Os requisitos descritos nesse documento são baseados na experiência adquirida com um grande expectro de aplicações com formulários.

Alguns meses depois, "XHTML Extended Forms" foi renomeado para "XForms" e movida para seu próprio grupo de trabalho. Este grupo trabalhou paralelamente com o grupo de trabalho da HTML e finalmente publicou em outubro de 2003 a primeira edição do XForms 1.0.

Enquanto isso com a transição para o XML completa, o grupo de trabalho da HTML colocaram seus esforços na "nova geração da HTML." Em maio de 2001 eles publicaram a primeira edição da XHTML 1.1, que adicionou apenas poucos recursos além dos que haviam na XHTML 1.0, mas também eliminou a brecha existente no "Apêndice C." A partir da versão 1.1, todos os documentos XHTML passam a funcionar a com o MIME type `application/html+xml`.

TUDO QUE VOCÊ SABE SOBRE XHTML ESTÁ ERRADO

Por que os MIME types são importantes? Por que eu fico voltando a eles? Três palavras: draconian error handling (tratamento de erros draconianos). Navegadores sempre foram "tolerantes" com a HTML. Se você criar uma página HTML mas esquecer a tag `</head>`, os navegadores irão mostrar a página de todo modo. (Algumas tags implicitamente realizam o fechamando da tag `</head>` e o inicio da tag `<body>`.) Você supostamente aninha as tags de maneira hierárquica — fechando elas da última para primeira — mas, se você criar uma marcação como `<b><i></b></i>`, os

navegadores simplesmente vão lidar com isso (de algum jeito) sem mostrar nenhuma mensagem de erro.

Como você pode esperar, o fato dessa marcação HTML "quebrada" continuar funcionando em alguns navegadores, permite que os autores criem páginas HTML quebradas. Muitas páginas quebradas. Segundo algumas estimativas, cerca de 99% das páginas HTML na internet hoje possuem pelo menos um erro nelas. Mas como os navegadores não mostram mensagens de erro para isto, ninguém conserta.

A W3C viu que este era o problema fundamental com a web, e, então, eles decidiram corrigir isso. O XML publicado em 1997 quebrou com a tradição de perdoar os clientes e mandou que todos os programas que consumissem XML deveriam tratar os erros chamados de "boa-formatação" como erros fatais. Este conceito de falha ficou conhecido como "tratamento de erro draconiano" depois do líder grego Draco que foi quem instituiu a pena de morte para quem cometesse menores infrações das suas leis. Quando a W3C reformulou a HTML como um vocabulário XML, eles mandaram que todos os documentos servidos pelo novo MIME type `application/xhtml+xml` deveriam submeter um tratamento de erro draconiano. Caso houvesse apenas um erro de boa-formação na sua página XHTML — Como por exemplo esquecer a tag `</head>` ou uma colocação errada de tags de início e fim — os navegadores não teriam escolha a não ser parar o processamento e mostrar uma mensagem de erro para o usuário final.

Essa ideia não foi popular universalmente. Com uma taxa estimada de erro em 99% das páginas, a sempre presente possibilidade de mostrar mensagens de erro para o usuário final e a escassez de novos recursos no XHTML 1.0 e 1.1 que justificassem o custo, os autores web basicamente ignoraram a `application/xhtml+xml`. Mas isso não significa que eles também ignoraram o XHTML. Definitivamente eles não fizeram isso. O Apêndice C da especificação do XHTML 1.0 deu aos autores da web uma nova brecha: "Usar algo que pareça com a sintaxe do XHTML, mas que continua gerando o MIME type `text/html`." Isso foi exatamente o que centenas de desenvolvedores web fizeram: eles fizeram um *upgrade* para a sintaxe do XHTML mas que continuassem gerando o MIME type `text/html`

Até hoje, milhões de páginas da internet dizem ser XHTML; Elas começam com o doctype XHTML na primeira linha, usam tags com caixa baixa, usam aspas nos atributos e uma barra simples ao final de elementos vazios como `<br />` e `<hr />`; Mas uma pequena fração dessas páginas são realmente servidas com o MIME type `application/xhtml+xml` e que façam o tratamento de erro draconiano. Qualquer página que possua o MIME type `text/html` — independente do doctype, sintaxe ou estilo de código — será interpretada usando um parseador HTML tolerante, que silenciosamente ignorará qualquer erro de marcação e nunca alertará os usuários finais (ou qualquer outra pessoa) se a página está tecnicamente quebrada;

O XHTML 1.0 inclui esta brecha, mas o XHTML 1.1. a fechou e o nunca finalizado XHTML 2.0 continuou com a tradição de requerer o tratamento de erro draconiano. É por isso que as bilhões de páginas que dizem ser XHTML 1.0. e apenas algumas dizem ser XHTML 1.1 (ou XHTML 2.0). Então você realmente está usando XHTML? Verifique o seu MIME type. (Na verdade, se você não sabe qual MIME type você está usando eu posso garantir que você continua usando `text/html`.) A menos que as suas páginas estejam utilizando o `application/xhtml+xml`, as páginas que você chama de "XHTML" é XML apenas no nome.

❧

UMA VISÃO CONCORRENTE

Em junho de 2004, o W3C realizou o Workshop sobre Aplicações Web e Documentos Combinados. Estavam presentes nesse workshop representantes de 3 fabricantes de browsers, companhias de desenvolvimento web e outros membros do W3C. Um grupo de partes interessadas, incluindo a Fundação Mozilla e a Opera Software, fizeram uma apresentação sobre sua visão concorrente do futuro da web: uma evolução do existente padrão HTML 4 para incluir novos recursos para desenvolvedores de aplicativos web modernos.

> Os 7 princípios a seguir representam o que acreditamos serem os requisitos mais importantes para este trabalho.
>
> Compatibilidade com versões anteriores, com um caminho claro de migração

Tecnologias de aplicação web devem ser baseadas em tecnologias em que os desenvolvedores estão familiarizados, incluindo HTML, CSS, DOM e JavaScript.

Características básicas de aplicativos web devem ser implementáveis através de comportamentos e se utilizando scripts e folhas de estilo em IE6 hoje para que os autores tenham um caminho claro de migração. Qualquer solução que não possa ser usada com o atual agente de usuário (user agent) de maior penetração no mercado, sem a necessidade de plugins binários, é altamente improvável de ser bem sucedida.

Tratamento de erro bem definido

O tratamento de erros em aplicações web deve ser definido em um tal nível de detalhe que permita que os agentes de usuário não tenham que bolar seus próprios mecanismos de manipulação de erros ou fazer engenharia reversa em outros agentes de usuário.

Usuários não devem ser expostos a erros de autoria

Especificações devem indicar o comportamento exato de recuperação de erros para cada cenário de erro possível. A manipulação de erro deve ser definida, na maior parte, em termos de recuperação de erro normal (como em CSS) ao invés de erros óbvios e catastróficos (como em XML).

Uso prático

Todo recurso que entra em especificações de aplicações web deve ser justificado por um caso de uso prático. O inverso não é, necessariamente, verdadeiro: cada caso de uso, não necessariamente, precisa ser o fundamento para um novo recurso.

Os casos de uso devem ser, preferencialmente, baseados em sites reais, nos quais os desenvolvedores utilizaram, anteriormente, uma solução ruim para contornar a limitação.

Scripting veio pra ficar

Mas deve ser evitado onde marcação mais convenientemente declarativa puder ser usada.

Scripting deve ser dispositiva e representativamente neutro, a não ser quando em um escopo bastante específico de dispositivos (por exemplo, a menos que incluso em XBL).

Profiling específico de dispositivo deve ser evitado

Desenvolvedores devem ser capazes de se valer das mesmas características, sejam elas implementadas em versões desktop ou móveis do mesmo agente de usuário.

Processo aberto

> A web foi beneficiada por ter sido desenvolvida num ambiente aberto. Aplicações web serão o *core* da web e seu desenvolvimento também deve se valer desse ambiente aberto. Listas de discussão, arquivos e rascunhos das especificações devem estar visíveis ao público.

Os participantes do workshop participaram de uma enquete com a pergunta "Deve o W3C desenvolver uma extensão declarativa para HTML e CSS e extensões imperativas para o DOM para tratar do nível médio de requisitos de aplicativos web, ao contrário das APIs maduras a nível de sistemas operacionais? (proposto por Ian Hickson, da Opera Software)". A votação foi de 11 contra 8. No sumário do workshop, o W3C escreveu: "Atualmente, o W3C não pretende colocar quaisquer recursos no terceiro tópico da enquete: extensões para HTML e CSS para aplicativos web, exceto tecnologias que estão sendo desenvolvidos sob a licença dos atuais Grupos de Trabalho do W3C."

Ante essa decisão, as pessoas que tinham propostas envolvendo HTML e formulários HTML tinham somente duas opções: desistir ou continuar seu trabalho fora do W3C. Eles escolheram a última e registraram o domínio `whatwg.org` e, em junho de 2004, o WHAT Working Group nasceu.

❧

WHAT WORKING GROUP?

Mas que diabos, afinal, é o WHAT Working Group? Vamos deixá-los explicar:

> O Web Hypertext Applications Technology Working Group é um grupo amplo, não oficial e aberto de fabricantes de navegadores web e partes interessadas. O grupo visa desenvolver especificações baseadas em tecnologias HTML e afins para facilitar o desenvolvimento de aplicativos web interoperáveis, com a intenção de submeter os resultados a uma organização de padrões. Esta submissão, então, constitui a base de trabalho em formalmente estender o HTML no caminho dos padrões web.
>
> A criação deste fórum decorre de vários meses de trabalho por e-mail, particularmente sobre as especificações de tais tecnologias. O foco principal, até este ponto, tem sido

> estender o HTML4 Forms para suportar as características solicitadas pelos autores sem quebrar a compatibilidade com o conteúdo existente. Este grupo foi criado para garantir que o desenvolvimento futuro dessas especificações será completamente aberto, através de uma lista de discussão aberta publicamente arquivada.

A frase-chave aqui é “sem quebrar compatibilidade com versões anteriores”. XHTML (menos o "furo" do apêndice C) não é retrocompatível com HTML. Ele precisa de um tipo de MIME inteiramente novo e trabalha com tratamento de erros draconianos para todo o conteúdo servido com aquele tipo de MIME. XForms não é retrocompatível com formulários HTML porque pode ser usado apenas em documentos disponibilizados com o novo tipo de MIME XHTML, o que significa que XForms também funciona com tratamento de erros de forma draconiana. Todos os caminhos levam ao MIME.

Ao invés de acabar com mais de 1 década de investimentos em HTML e fazer com que 99% das páginas da web existentes fiquem inutilizáveis, o WHAT Working Group optou por uma abordagem diferente: documentar os algoritmos de tratamento de erros "perdoáveis" que os navegadores, efetivamente, utilizam. Web browsers sempre foram indulgentes com erros de HTML, mas ninguém jamais se preocupou em escrever sobre como, exatamente, eles fizeram isso. NCSA Mosaic teve seus próprios algoritmos para lidar com páginas quebradas e a Netscape tentou o mesmo. Então, o Internet Explorer tentou se igualar ao Netscape. Em seguida, o Opera e o Firefox tentaram fazer a mesma coisa que o Internet Explorer. Daí, o Safari tentou se igualar ao Firefox. E assim por diante, até os dias de hoje. Ao longo do caminho, os desenvolvedores queimaram milhares e milhares de horas tentando tornar seus produtos compatíveis com os de seus concorrentes.

Se isso soa como uma quantidade insana de trabalho, é porque, realmente, é. Ou melhor, era. Demorou 5 anos, mas (exceto por alguns casos obscuros extremos) o WHAT Working Group tem documentado com sucesso como analisar HTML de uma maneira que seja compatível com o conteúdo web já existente. Em nenhum ponto do algoritmo final há um passo que exija que o consumidor de HTML deva parar o processamento a fim de exibir uma mensagem de erro para o usuário final.

Enquanto toda a engenharia reversa estava acontecendo, o WHAT Working Group foi, calmamente, trabalhando em algumas outras coisas, também. Uma delas era uma especificação, inicialmente apelidada Web Forms 2.0, que adicionou novos tipos de controles aos formulários HTML. (Você aprenderá mais sobre formulários web em Uma Forma De Loucura.) Outra foi um rascunho de uma especificação chamada "Web Applications 1.0", que incluía grandes novidades, tais como uma tela para desenhar canvas diretamente e suporte nativo para áudio e vídeo, sem plugins.

DE VOLTA AO W3C

Por dois anos e meio, o W3C e o WHAT Working Group se ignoraram mutuamente. Enquanto o WHAT Working Group focou em formulários web e novos recursos de HTML, o W3C HTML Working Group estava ocupado com a versão 2.0 do XHTML. Mas, em outubro de 2006, ficou claro que o WHAT Working Group teve seu momento crítico, enquanto o XHTML 2 ainda estava definhando em forma de rascunho, não implementado por nenhum navegador principal. Em outubro de 2006, Tim Berners-Lee, o fundador da W3C, em pessoa, anunciou que o W3C poderia trabalhar em conjunto com o WHAT Working Group a fim de evoluir o HTML.

> Algumas coisas ficaram mais claras com o passar dos anos. É necessário evoluir o HTML de forma incremental. A tentativa de fazer o mundo mudar para XML, incluindo aspas em torno de valores de atributos e barras em tags vazias e namespaces, de uma só vez, não funcionou. O grande público gerador de HTML não se moveu, em grande parte porque os navegadores não reclamaram. Algumas grandes comunidades fizeram mudanças e estão colhendo os frutos de sistemas bem-formados, mas não todos. Isso é

> importante para manter a HTML de forma incremental, bem como continuar a transição para um mundo bem-formado e desenvolver mais poder nesse mundo.
>
> O plano é constituir um grupo completamente novo de HTML. Ao contrário do anterior, este será fretado para fazer melhorias incrementais na HTML, bem como em XHTML, paralelamente. Ele terá uma cadeira e pessoal diferentes. Ele irá funcionar em HTML e XHTML em conjunto. Temos um forte apoio para esse grupo, a partir de muitas pessoas que falaram com, incluindo os fabricantes de browsers.
>
> Haverá trabalho, também, em formulários. Esta é uma área complexa, já que existem formulários tanto em HTML, quanto em XForms. Formulários HTML são ubiquamente implementados e existem muitas implementações e usuários de XForms. Enquanto isso, a apresentação de Webforms sugeriu extensões sensatas para formulários HTML. O plano é, informada pela Webforms, de estender formulários HTML.

Uma das primeiras coisas que o recém re-formado W3C HTML Working Group decidiu mudar é o nome de "Web Applications 1.0" para "HTML5". E aqui estamos nós, mergulhando em HTML5.

POSTSCRIPT

Em outubro de 2009, o W3C descontinuou o Grupo de Trabalho de XHTML 2 e emitiu esse comunicado para explicar sua decisão:

> Quando o W3C anunciou os Grupos de Trabalho de HTML e XHTML 2 em março de 2007, indicamos que iríamos continuar a acompanhar o mercado de XHTML 2. O W3C reconhece a importância de um sinal claro à comunidade sobre o futuro do HTML.
>
> Embora reconheçamos o valor do Grupo de Trabalho do XHTML 2 ao longo dos anos, após discussão com os participantes, a gestão W3C optou por permitir a expiração do Grupo de Trabalho no fim de 2009 e não renová-lo.

Os que ganham são aqueles a bordo.

LEITURA COMPLEMENTAR

- A História da Web, um projeto antigo de Ian Hickson
- HTML/História, de Michael Smith, Henri Sivonen e outros
- Uma Breve História do HTML, de Scott Reynen

Este foi o "Como chegamos aqui?". Consulte o Sumário, caso queira continuar com a leitura.

VOCÊ SABIA?

Em associação a Google Press, O'Reilly está distribuindo este livro em variados formatos, incluindo papel, ePub, Mobi, DRM-free e PDF. A edição paga é chamada "HTML5: Up & Running" e está disponível agora. Este capítulo está incluído na versão paga.

Se você gostou deste capítulo e quer mostrar sua apreciação, basta comprar o livro "HTML5: Up & Running" com esse link afiliado ou comprar a edição eletrônica diretamente da O'Reilly. Você vai ganhar um livro, e eu vou ganhar um trocado. Atualmente, não aceito doações diretas.