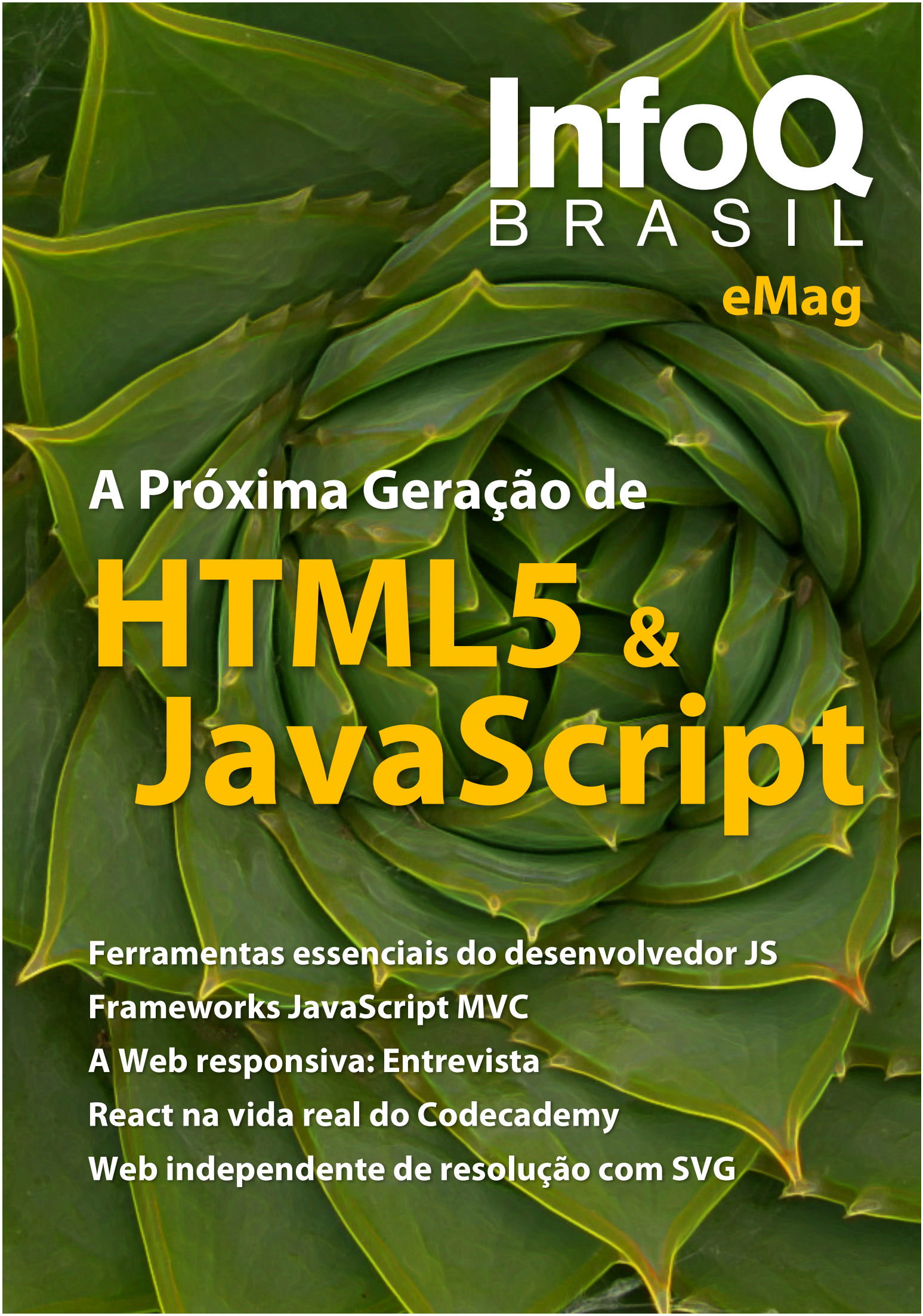
InfoQ
BRASIL
eMag
A Próxima Geração de
HTML5 & JavaScript
Ferramentas essenciais do desenvolvedor JS
Frameworks JavaScript MVC
A Web responsiva: Entrevista
React na vida real do Codecademy
Web independente de resolução com SVG

Conteúdo

Apresentação

O JavaScript está em constante evolução, e quase todas as semanas surgem novos frameworks e bibliotecas. Recentemente a maior mudança na linguagem veio com o ECMAScript 6, que foi finalizado em 2015.

O desenvolvedor web moderno precisa conhecer muito mais do que antes – e ficar parado não é opção. Esta eMag procura ser um guia nessa direção. Na seleção de artigos do InfoQ, temos a colaboração dos desenvolvedores que constroem as aplicações web mais avançadas para aplicar o que eles aprenderam em nosso próprio trabalho.

David Haney do StackOverflow começa com uma descrição do que é a caixa de ferramentas do desenvolvedor JavaScript em 2015. As IDEs são apenas o início. Foi-se o tempo de colocar o jQuery no projeto como primeira e única biblioteca JavaScript. Se agora é o AngularJS, ReactJS ou outra tecnologia que surgiu nos últimos dez minutos, David Haney nos lembra que a escolha da biblioteca JavaScript é um passo importante. Adicione gulp, linters, empacotadores e minificação a caixa de ferramentas, e o desenvolvedor moderno pode entregar alta qualidade, JavaScript e CSS rápidos – perfeito para a web moderna.

Dio Synodinos, editor do InfoQ, organizou um painel virtual com líderes da comunidade JavaScript, com o objetivo de conversar sobre os frameworks MVC. Foram abordados os tipos de problemas que os frameworks resolvem; mobile, desempenho e fluxo de trabalho. O artigo fornece uma visão rápida de trás dos teclados dos desenvolvedores JavaScript mundiais.

Muito desta eMag é sobre JavaScript, mas o JavaScript hoje não seria nada sem o HTML5. Esperamos construir aplicações web e sites que funcionam não apenas em navegadores desktop, mas também em múltiplos dispositivos móveis. Entrevistamos Matt Carver, autor do livro The Responsive Web, para conhecer os desafios envolvidos na construção de sites web responsivos.

Continuando, Bonnie Eisenman relata a história de como a Codecademy começou a usar React.js em produção. Criado pelo Facebook, o React.js possui um dos mais rápidos sistemas de renderização, e isso começou a chamar a atenção dos desenvolvedores de todas as partes. Por outro lado, há muitos exemplos e demonstrações "de brinquedo", que não demonstram o real significado do que um aplicativo precisa em produção. Com base numa experiência real, com demandas, Eisenman apresenta o que funciona e o que não funciona juntamente com uma descrição de como introduzir o React em seu aplicativo web.

As telas "retina", com altíssima densidade de pixels, agora são realidade. Embora isso seja excelente para os usuários, os designers e desenvolvedores precisam lidar com imagens em uma gama muito mais ampla de resoluções. Angelos Chaidas nos guia pelo uso do SVG para combater o trabalho excessivo e aborda o uso de animação e interação, além de fornecer detalhes de como esses processos se encaixam no fluxo de trabalho.

O HTML e o JavaScript estão entre as partes mais antigas da web, mas ainda estão mudando a cada dia e continuam ganhando em importância. Esperamos que essa eMag possa ajudar os leitores a chegar mais rapidamente ao estado da arte do desenvolvimento web.

A moderna caixa de ferramentas do desenvolvedor JavaScript

por David Haney, traduzido por Rafael Sakurai

A caixa de ferramentas de um moderno desenvolvedor JavaScript vem mudando muito nos últimos 20 anos. De IDEs a ferramentas de automatização, há diversas opções para os desenvolvedores.

O JavaScript é uma linguagem de script desenhada inicialmente para tratar páginas web, mas agora é usada de quase todas as formas imagináveis. Avanços foram feitos para permitir que o JavaScript execute no lado servidor bem como compilado em código de aplicações nativas para telefones. Hoje o desenvolvedor JavaScript faz parte de um rico ecossistema formado por centenas de IDEs, ferramentas e frameworks. Dada a quantidade de opções e recursos, alguns desenvolvedores podem encontrar dificuldade de saber como começar. Nesse artigo será discutido o perfil do desenvolvedor JavaScript moderno e apresentados os mais populares frameworks, ferramentas e IDEs.

Uma breve excursão pela história

Vamos voltar para 1995, quando o Netscape Navigator e o Internet Explorer 1.0 eram as opções disponíveis de navegadores. Os websites tinham apenas textos e muitos GIFs. Um site completo de conteúdo rico demorava dois minutos para carregar em uma conexão discada (dial-up). Ao longo do caminho, uma linguagem web nasceu permitindo que esses websites antigos pudessem executar código no lado cliente. Foi o ano em que nasceu o JavaScript.

Os websites de 20 anos atrás não usavam muito JavaScript e certamente não usavam seu potencial total. Ocasionalmente havia um alerta popup para informar algo, a rolagem de um texto em uma caixa com notícias ou um cookie que armazenava seu nome e apresentava novamente quando retornasse alguns meses depois. Certamente não havia empregos nos quais o JavaScript era a linguagem primária, salvo os poucos sortudos que tinham o trabalho de criar o próprio JavaScript. Em resumo, a linguagem era um brinquedo para os websites enfeitarem os seus DOM (Document Object Model).

Hoje, o JavaScript pode ser encontrado virtualmente em todos os locais. Do AJAX ao Bootstrap, ReactJS, Angular, o ubíquo jQuery e mesmo o Node.js no lado servidor, o JavaScript se tornou uma das mais importantes e populares linguagens de desenvolvimento.

Frameworks

Um dos principais aspectos no qual o JavaScript mudou desde sua origem foi na sua forma de uso. Já se foram os dias que precisávamos chamar **document.GetElementById** e tratar objetos do tipo **XmlHttpRequest**. Em vez disso, diversas bibliotecas abstraíram as funcionalidades principais tornando o JavaScript mais acessível aos desenvolvedores. Isso é um dos grandes motivos porque encontramos o JavaScript em todos os lugares.

jQuery

O jQuery, criado em 2006 por John Resig, fornece um rico conjunto de ferramentas para abstrair e simplificar os comandos, métodos ocultos e rígidos do JavaScript. A forma mais fácil de demonstrar isso é através de um exemplo. Esta é uma requisição AJAX feita com JavaScript puro:

```
function loadXMLDoc() {
   var xmlhttp;

   if (window.XMLHttpRequest) {
       // código para IE7+, Firefox, Chrome, Opera, Safari
       xmlhttp = new XMLHttpRequest();
   } else {
       // código para IE6, IE5
       xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
   }

   xmlhttp.onreadystatechange = function() {
       if (xmlhttp.readyState == 4 ) {
          if(xmlhttp.status == 200){
              alert("sucesso");
          }
          else if(xmlhttp.status == 400) {
             alert("erro 400")
          }
          else {
              alert("algo está quebrado!")
          }
       }
   }

   xmlhttp.open("GET", "test.html", true);
   xmlhttp.send();
}
```

Fonte: Stack Overflow

Veja uma requisição AJAX feita com jQuery:

```
$.ajax({
   url: "test.html",
   statusCode: {
        200: function() {
               alert("sucesso");
        },
        400: function() {
               alert("erro 400");
        }
   },
   error: function() {
        alert("algo quebrado!");
   }
});
```

O jQuery fez com que partes complexas das funções JavaScript se tornassem fáceis para usar e manipular trechos do DOM. Como resultado, é uma das bibliotecas mais amplamente usadas para JavaScript. A ideia de abstração que veio com o jQuery foi a base na qual outras bibliotecas e frameworks foram modelados.

AngularJS

O AngularJS, ou simplesmente Angular como é chamado, surgiu em 2009. Foi criado pelo Google para facilitar a construção de aplicações de página única (Single Page Applications – SPA). Como o jQuery, o Angular também abstrai o trabalho difícil em métodos reutilizáveis, além de fornecer uma arquitetura de model-view-controller (MVC) para o JavaScript.

ReactJS

O ReactJS, ou "React" como vem sendo chamado, é o novato. Foi criado pelo Facebook e liberado pela primeira vez em 2013. O Facebook considera o React como uma nova forma de tratar problemas de SPA que o Angular ainda trabalha para resolver. É correto comparar Angular e React como frameworks concorrentes. No entanto, o que os realmente diferencia é que o React é mais eficiente e tem uma biblioteca mais veloz. O gráfico a seguir apresenta o tempo de execução do React, Angular, Knockout (uma biblioteca de terceiros não discutida nesse artigo) e JavaScript puro para renderizar uma lista com mil itens no DOM (veja a imagem). Se o desempenho é importante para sua aplicação, então o React é o caminho a seguir.

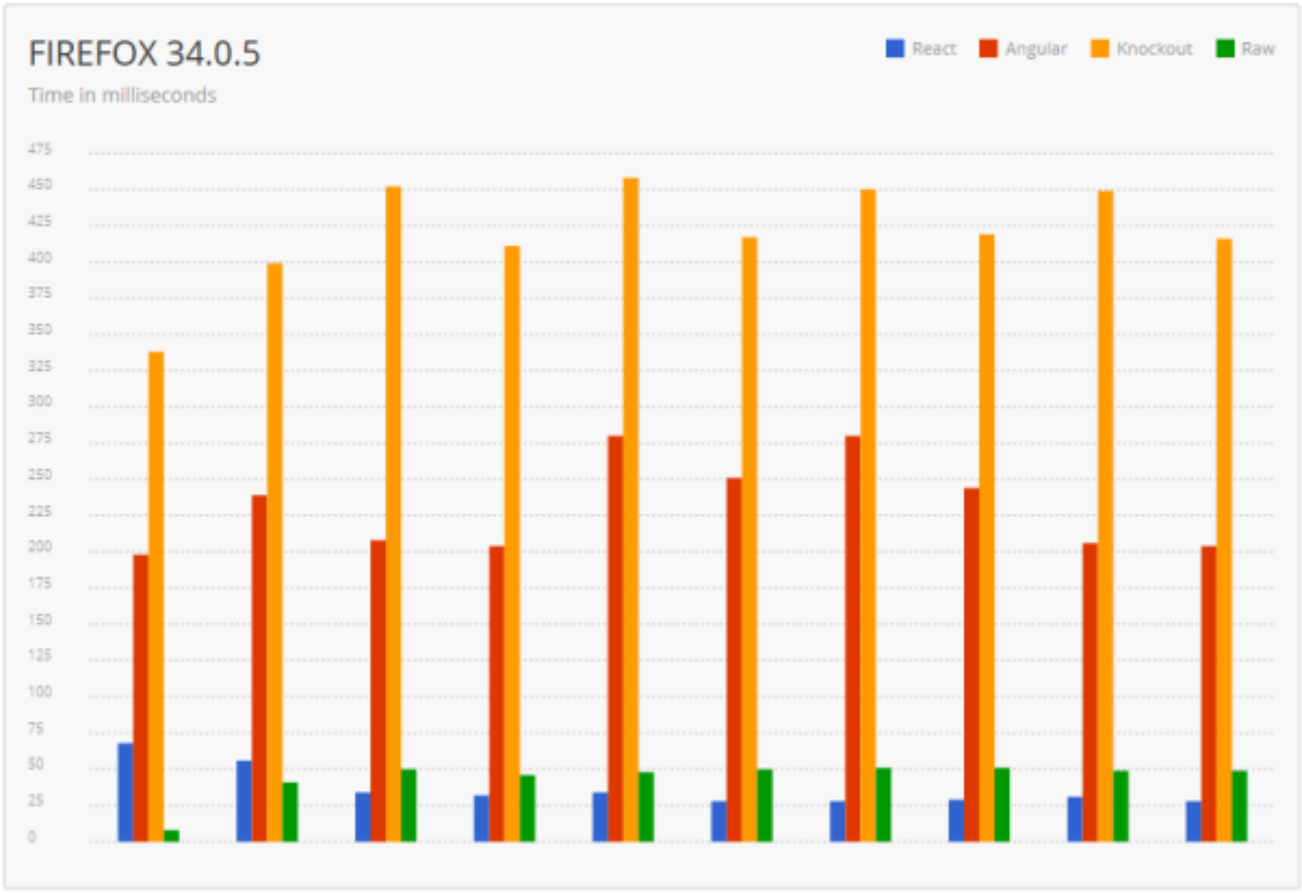

Figura 1: *Comparação de velocidade dos frameworks React, Angular, Knockout e puro JavaScript. (Fonte: The Dapper Developer)*

Ambientes de desenvolvimento JavaScript

Uma importante parte do desenvolvimento eficiente é o uso de uma IDE. Uma IDE, ou Ambiente de Desenvolvimento Integrado, é uma aplicação que oferece um conjunto de ferramentas para o desenvolvedor. A parte mais importante dessas ferramentas normalmente é o editor de texto, que oferece realce de sintaxe, autocompletar e teclas de atalho, que aceleram os processos manuais.

Sublime Text

O Sublime Text não é na verdade uma IDE. Ele é um editor de texto leve para programação superrápida, que oferece o realce de sintaxe e teclas de atalhos intuitivas. Sendo multiplataforma, é excelente para os desenvolvedores que querem usar no Mac ou PC. Virtualmente tudo no Sublime Text pode ser customizado. Também oferece múltiplos plug-ins que permitem recursos como de

uma IDE, por exemplo, a integração com Git e análise de erros no código (linting). É uma ótima escolha entre os entusiastas e novos desenvolvedores JavaScript. O Sublime Text pode ser baixado e avaliado gratuitamente, e sua licença custa $70.

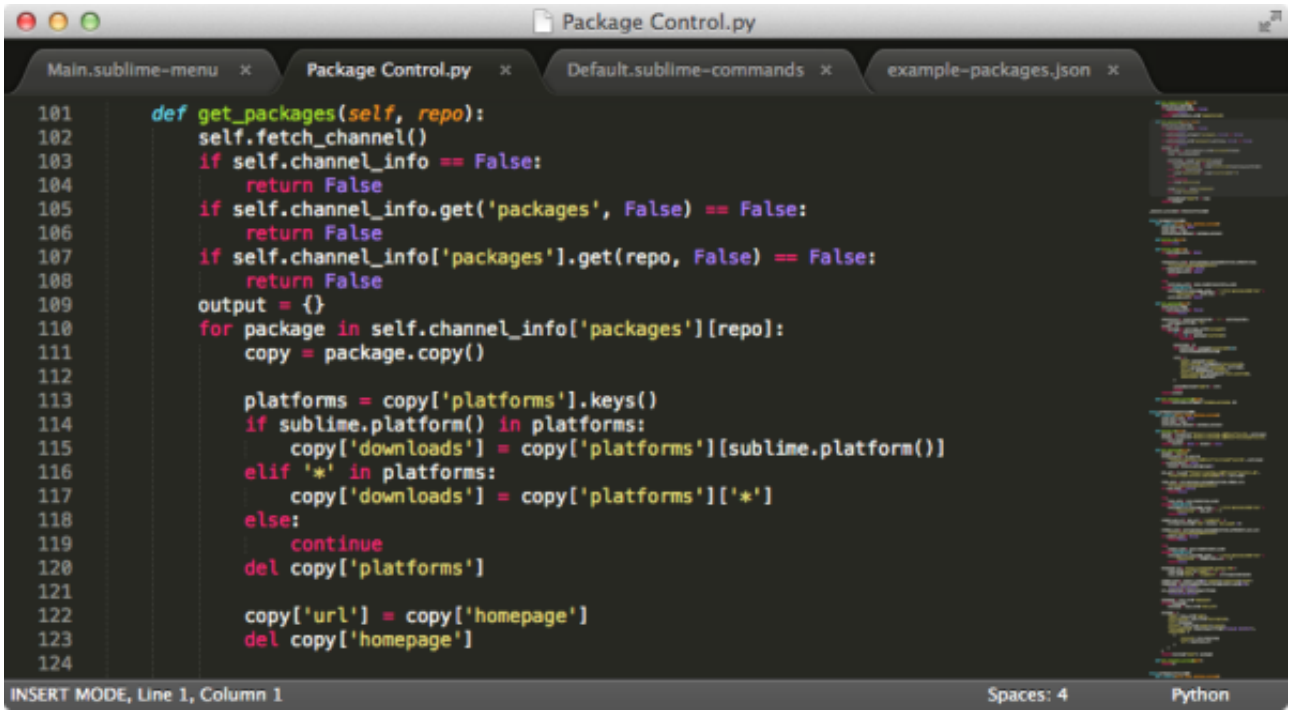

Figura 2: *Editor de texto Sublime Text. (**Fonte:** Sublime Text)*

WebStorm

O WebStorm foi criado pela equipe da JetBrains como uma pequena IDE focada em HTML, CSS e JavaScript. O custo da licença é $49 e é amplamente considerado como padrão para profissionais JavaScript, e por uma boa razão, o construtor de complementação de código e inspeção de ferramentas são inigualáveis. O WebStorm também oferece um depurador de JavaScript rico e testes de unidade integrado com os frameworks populares, como: execução de testes Karma e JSDriver e até o Mocha do Node.js.

Uma das melhores funcionalidades do WebStorm é a funcionalidade de Edição ao Vivo. Instalando um plug-in em ambos Chrome e WebStorm, o desenvolvedor pode fazer mudanças no código que são instantaneamente refletidos no navegador. Os desenvolvedores também podem configurar a Edição em Tempo Real para destacar as mudanças que são feitas na janela do navegador, permitindo depuração e alta produtividade durante a codificação.

No geral, o WebStorm é a IDE para escolher se usar JavaScript em tempo integral.

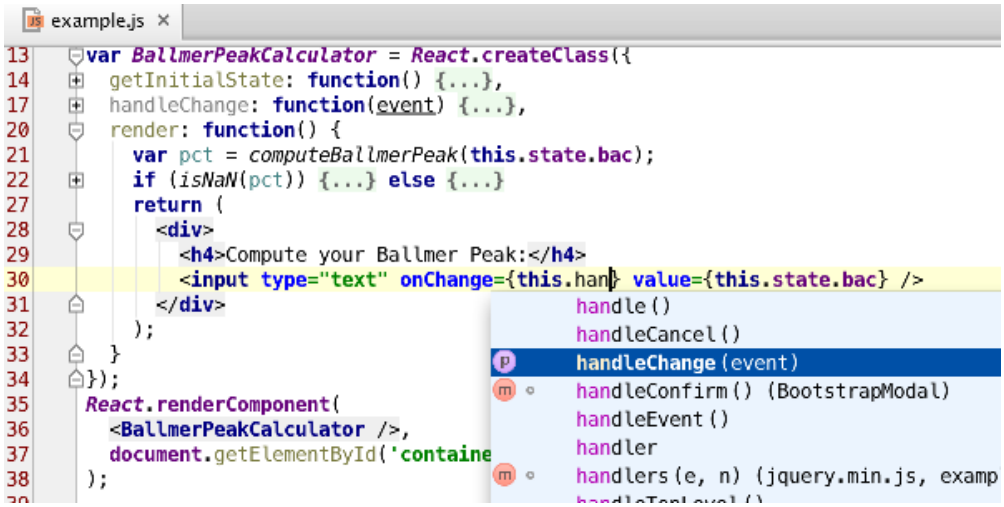

Figura 3: *IDE WebStorm.* ***Fonte:*** ***JetBrains***

Brackets

Brackets é uma IDE gratuita e de código aberto construído com foco na ferramenta visual. O Brackets também oferece a funcionalidade de edição ao vivo similar ao WebStorm que deixa visualizar instantaneamente os resultados das mudanças nos códigos na janela do navegador. Também suporta edição lado a lado, que permite trabalhar no código e ver os resultados simultaneamente sem precisar trocar entre aplicações e janelas popup. Uma das funcionalidades mais interessantes do Brackets chama Extract, que analisa arquivos Photoshop (PSD) para obter as fontes, cores e informações de medições. Essa funcionalidade faz do Brackets uma excelente escolha para os desenvolvedores JavaScript que também trabalham com design.

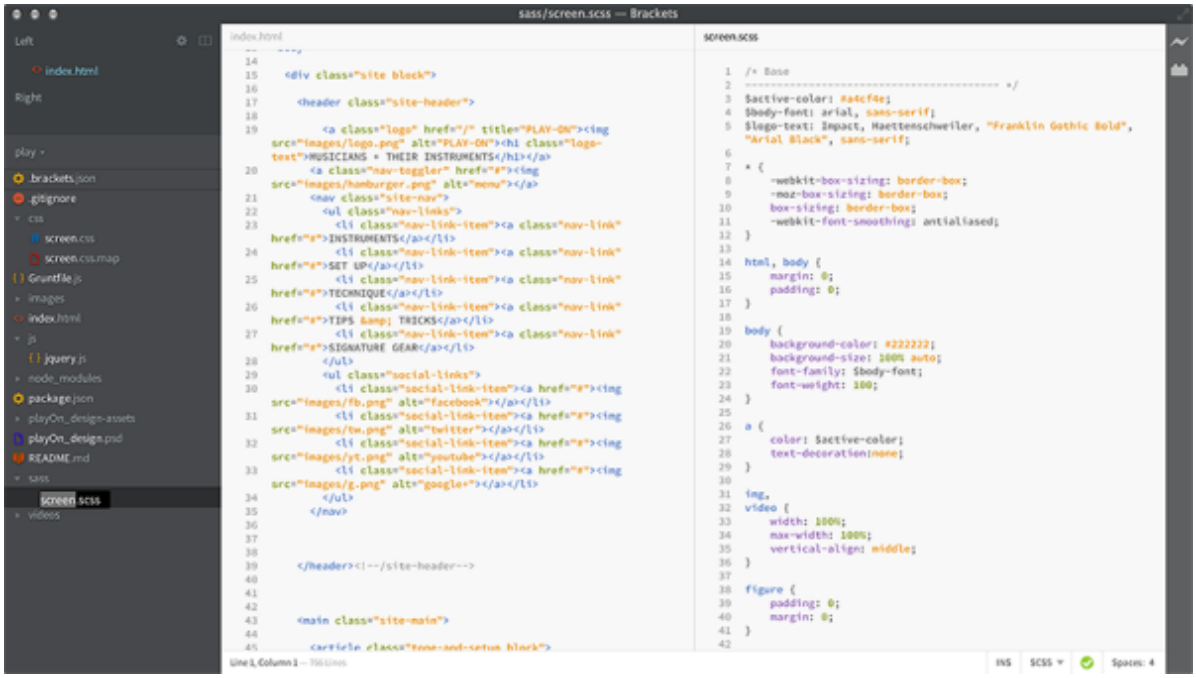

Figura 4: *IDE Brackets. (****Fonte:*** ***Brackets)***

Atom

Atom é um editor de texto rico de código aberto criado pelo GitHub. Muito acessível e fácil de usar, é possível instalar e executar sem precisar mudar os arquivos de configuração, ele "simplesmente funciona". A parte mais interessante do Atom é a habilidade para customizar todos os aspectos (ele foi definido como "hackeavel" pelo GitHub). O Atom é construído em cima de um núcleo web, permite escrever seus próprios padrões de HTML, CSS e JavaScript para customização da aparência. Quer cores de background e fontes diferentes no Atom? Apenas altere os valores do CSS. Alternativamente, podemos baixar e aplicar um dos muitos temas criados para o Atom. Isso permite flexibilizar o Atom da forma como quiser. O Atom é uma ferramenta excelente para os novos desenvolvedores JavaScript e também para hackers entusiastas.

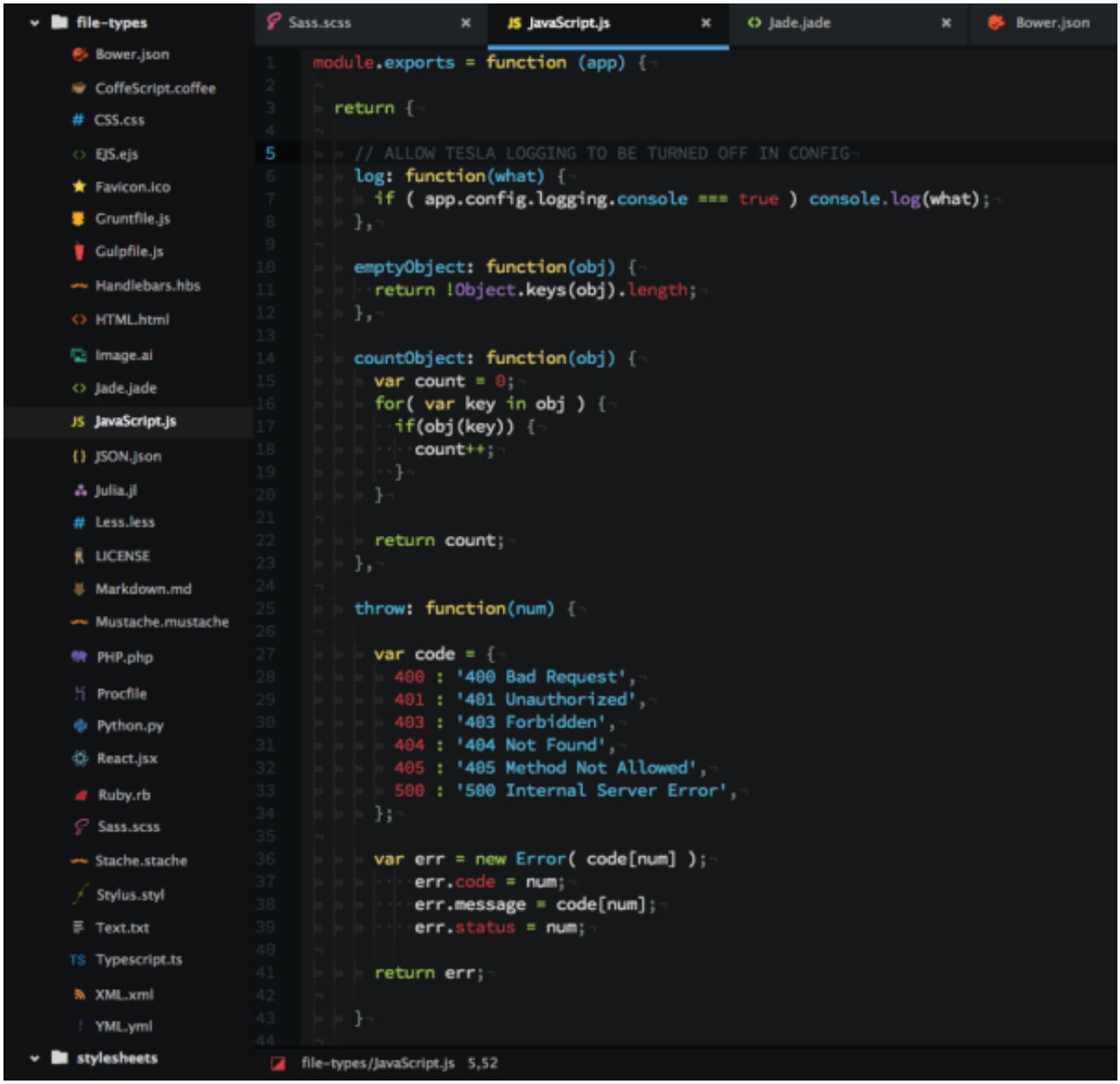

Figura 5: *Editor de texto Atom.* **Fonte:** *Atom*

Ferramentas de construção e automatização

Os modernos projetos que utilizam JavaScript tendem a ser um pouco complexos, com muitas peças móveis. Isso não é devido a ineficiência da linguagem ou das ferramentas; isso é resultado direto das aplicações web ricas, vibrantes e complexas que são construídas nos dias atuais.

Quando trabalhamos em grandes projetos, haverá muitos processos repetitivos que devemos fazer sempre que precisamos verificar um código ou liberar a versão de produção. Esses processos podem ser como: o empacotamento de vários arquivos em um único arquivo (bundling), a redução dos arquivos (minification), compilação dos arquivos CSS LESS ou SASS e até execução dos testes. Fazer todo esse trabalho manualmente é frustrante e ineficiente. É muito melhor automatizar o trabalho através de uma ferramenta que suporta essas tarefas.

Bundling e Minification

Muitos dos códigos JavaScript e CSS escritos precisam ser compartilhados entre diversas páginas web. Como resultado, o conteúdo é guardado em arquivos .js e .css, e estes são então referenciados nas páginas web. Com isso, o navegador do visitante faz uma requisição HTTP para obter esses arquivos (ou pelo menos verificar se houve mudança) para renderizar completamente sua aplicação web.

Realizar diversas requisições HTTP é custoso. Então, no topo temos os gastos com carga de dados, mas também pagamos pelo custo de latência da rede, cabeçalhos e cookies. As ferramentas de bundling e minification são projetadas para reduzir ou eliminar completamente esses custos.

Bundling

Uma das coisas mais simples que o desenvolvedor pode fazer é aprimorar o desempenho dos códigos web empacotando eles em um único arquivo. O bundling é o processo de juntar múltiplos arquivos JavaScript e CSS em arquivos únicos. Isso é como juntar partes individuais de uma foto panorâmica para criar uma única e foto contínua. Juntando os arquivos JavaScript e CSS, eliminamos muitos gastos com requisições HTTP.

Minification

Outra maneira na qual o desenvolvedor JavaScript pode aprimorar o desempenho é através da minificação (minification) do código recém empacotado. A minification reduz o JavaScript e CSS a menor forma possível, mantendo ao mesmo tempo a funcionalidade idêntica. Para o JavaScript isso significa renomear as variáveis para tokens de caracteres únicos e sem sentido, remover todos os espaços em branco e formatações. Para o CSS, que se baseia nos nomes das variáveis, isso tipicamente significa somente remover a formatação e espaços em branco. A minificação fornece uma drástica melhoria de desempenho da rede porque reduz o número de bytes enviados na resposta HTTP.

A seguir temos o trecho de código JavaScript AJAX mostrado anteriormente, não minificado:

```
$.ajax({
   url: "test.html",
   statusCode: {
        200: function() {
               alert("sucesso");
        },
        400: function() {
               alert("erro 400");
        }
   },
```

```
   error: function() {
       alert("algo quebrado!");
   }
});
```

Aqui está o mesmo código minificado:

```
$.ajax({url:"test.html",statusCode:{200:function() {alert("sucesso");},
400:function(){alert("erro 400");}},error:function(){alert("algo quebrado!");}});
```

Note que o arquivo minificado de saída foi quebrado em duas linhas para apresentar nesse artigo. A saída atual da minificação é normalmente uma única linha.

Quando empacotar e minimizar

Geralmente, os passos de empacotar e minimizar são feitos apenas em produção. Dessa forma podemos depurar o código localmente ou em ambiente de desenvolvimento, com o código completamente formatado e com número nas linhas. A depuração do código minimizado é difícil, pois todo o código estaria na linha 1. A minificação deixa o código completamente ilegível, que o torna ainda mais inútil e frustrante para quem tentar depurar.

Mapeamento de arquivos fonte

Algumas vezes um bug ocorre no seu código que somente pode ser reproduzido em produção. Isso coloca um problema quando precisamos depurar algum item, mas todo o código está minificado. Felizmente, no JavaScript é possível realizar o mapeamento de fontes que "mapeia" entre o código minificado e o código original.

O arquivo de mapeamento é gerado durante a minificação através da ferramenta de empacotamento. Seus depuradores favoritos de JavaScript usam o arquivo de mapeamento para fornecer um código legível para depuração. Podemos publicar o código de produção com os arquivos de mapeamento sempre que possível, então podemos depurar o código se algo der errado.

Linting

Uma ferramenta de linting analisa o código procurando erros comuns e equívocos nas regras de formatação. Os tipos de erros reportados são coisas como uso de tabs ao invés de espaços; ou falta de ponto e vírgula no final das linhas ou usar chaves fora da declaração de **if**, **for** ou **while**. As IDEs mais comuns vêm com ferramentas de linting; outras permitem instalar plugins.

Os linters mais populares de JavaScript são JSHint e JSLint. O JSHint é desenvolvido pela comunidade e é um fork do JSLint, o framework original de linting escrito por Douglas Crockford. Variam um pouco nos padrões de formatação de código que eles impõem. Aconselho o uso de ambos para escolher qual o melhor para o estilo de codificação do projeto que estiver trabalhando.

Automação com Grunt

Apesar do nome, o Grunt (grunhido em inglês) está longe de ser primitivo. É uma ferramenta robusta de construção em linha de comando que executa tarefas definidas pelo usuário. Através da especificação de um arquivo de configuração simples, podemos configurar o Grunt para compilar LESS ou SASS, construir e minimizar todos os arquivos JavaScript e CSS em pastas específicas, ou mesmo executar uma ferramenta de linting ou framework de testes. Também é possível configurar o Grunt para ser executado como parte de um script personalizado do Git (Git hook) – minimizando e construindo seu código somente se houver algo no repositório de controle do código.

O Grunt suporta targets nomeados, então é possível especificar diferentes comandos para diferentes ambientes; pode definir alvos como "dev" e "prod", por exemplo. Isso é útil em diversos cenários como na construção e minimização do código em produção, mas deixando normal o ambiente de desenvolvimento para facilitar a depuração.

Uma funcionalidade útil do Grunt é o "grunt watch" que monitora se há mudanças no diretório ou conjunto de arquivos. Pode ser integrado diretamente com IDEs como: WebStorm e SublimeText. Usando o "watch" é possível acionar eventos com base na alteração dos arquivos. Uma aplicação prática dessa abordagem é a compilação de LESS e SASS. Ao configurar o monitoramento dos arquivos LESS e SASS é possível compila-los imediatamente após a mudança do arquivo, fazendo com que a saída da compilação fique imediatamente disponível no seu ambiente de desenvolvimento. Outro exemplo é durante o desenvolvimento, pois a página realiza de maneira automática o "atualizar" a cada alteração nos arquivos do projeto. Também é possível usar essa funcionalidade para automaticamente executar uma ferramenta de linting em qualquer arquivo modificado. A execução em tempo real das tarefas via "grunt watch" é uma das formas de aumentar sua produtividade.

Automação com Gulp

O Grunt e o Gulp são competidores diretos, que disputam para resolver os mesmos problemas de automatização da construção (builds). A maior diferença entre eles é que o Grunt foca em configuração, enquanto o Gulp foca no código. Configuramos as tarefas em arquivos JSON no Grunt e escrevemos funções JavaScript nos arquivos Gulp para realizar as mesmas tarefas.

A seguir temos um arquivo de configuração do Grunt para compilar arquivos SASS para CSS, sempre que o arquivo for alterado (fonte: Grunt vs Gulp – Beyond the Numbers):

```
grunt.initConfig({
 sass: {
   dist: {
     files: [{
       cwd: "app/styles",
       src: "**/*.scss",
       dest: "../.tmp/styles",
       expand: true,
       ext: ".css"
     }]
   }
 },
 autoprefixer: {
   options: ["last 1 version"],
   dist: {
     files: [{
       expand: true,
       cwd: ".tmp/styles",
       src: "{,*/}*.css",
       dest: "dist/styles"
     }]
   }
 },
 watch: {
   styles: {
     files: ["app/styles/{,*/}*.scss"],
     tasks: ["sass:dist", "autoprefixer:dist"]
   }
 }
});
grunt.registerTask("default", ["styles", "watch"]);
```

O arquivo Gulp a seguir configura a compilação de arquivos SASS para CSS sempre que o arquivo for alterado:

```
gulp.task("sass", function () {
 gulp.src("app/styles/**/*.scss")
   .pipe(sass())
   .pipe(autoprefixer("last 1 version"))
   .pipe(gulp.dest("dist/styles"));
});
gulp.task("default", function() {
 gulp.run("sass");
 gulp.watch("app/styles/**/*.scss", function() {
   gulp.run("sass");
 });
});
```

Recomendo usar um ou outro de acordo com sua preferência. Ambas as ferramentas são instaladas através do npm, o Gerenciador de Pacotes do Node.js.

Conclusões

O JavaScript está evoluindo significativamente desde seu nascimento nos primeiros dias da internet. E hoje é uma funcionalidade fundamental para a interação das aplicações web. O desenvolvedor JS também evoluiu muito desde 1995. Desenvolvedores JavaScript modernos usam frameworks ricos e robustos, assim como IDEs para trabalhar de modo eficiente e produtivo.

Construir sua primeira aplicação JavaScript moderna é mais fácil do que se imagina! Apenas escolha uma IDE (recomendo o Atom para os iniciantes) e então instale o npm e grunt. Se houver problemas ao longo do caminho, o Stack Overflow é um recurso excelente. Com pouco tempo será possível aprender tanto o básico como a maneira de publicar sua aplicação JavaScript moderna.

Sobre o autor

David Haney trabalha na equipe de engenharia na Stack Exchange, criador do StackOverflow e ServerFault. Passa o dia ajudando os desenvolvedores a resolver seus problemas e melhorar seus processos. Trabalhou antes como líder de desenvolvimento no e-commerce Fanatics. David é criador do Dache, um framework de caching distribuído de código aberto.

Painel Virtual: Frameworks JavaScript MVC

por Dio Synodinos, traduzido por Roberto Pepato

A plataforma web evoluiu muito desde que o HTML5 se tornou popular. Muitas APIs surgiram e há muito conteúdo sobre como o navegador pode se aproveitar delas.

Conforme mais e mais lógica é executada no navegador, os códigos de front-end em JavaScript tornam-se maiores e mais difíceis de manter. Como forma de resolver este problema, os desenvolvedores passaram a utilizar frameworks MVC, que prometem entregar melhorias na produtividade e código de manutenção mais simples.

Em 2013, o InfoQ pediu à comunidade para classificar os frameworks MVC JavaScript, de acordo com os recursos que eles apresentavam e com sua maturidade:

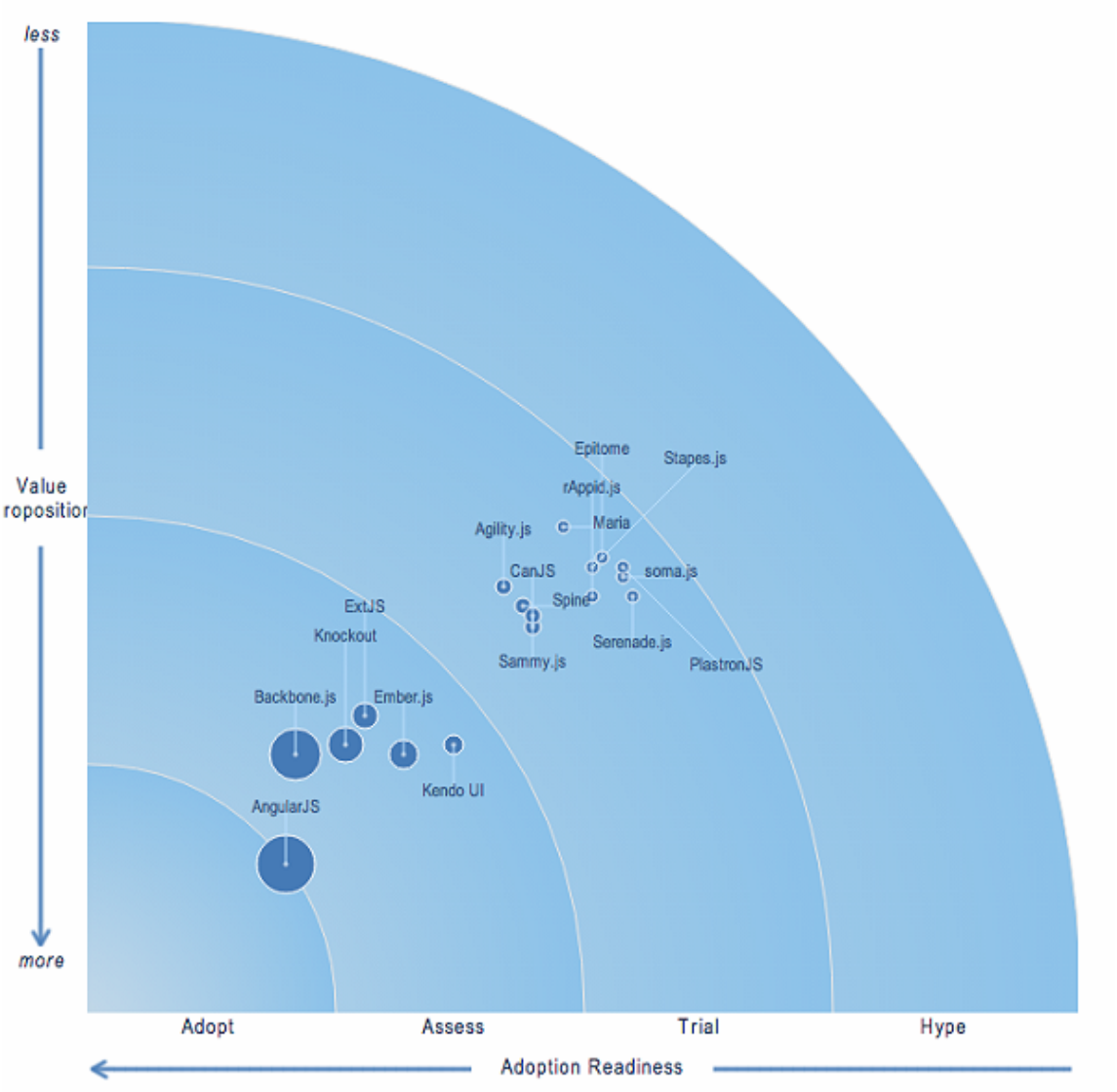

Figura 1: *Classificação dos frameworks JavaScript pela comunidade do InfoQ em 2013.*

O InfoQ consultou a opinião de especialistas sobre como utilizam estes frameworks e quais as melhores práticas que seguem para desenvolver aplicações JavaScript.

InfoQ: Qual é seu framework JavaScript MVC predileto, há quanto tempo o utiliza e qual é o problema fundamental que ajuda a resolver?

Matteo Pagliazzi: Angular é o framework do momento, e isso não é somente uma questão de estrelas (stars) no GitHub: ele tem o maior número de bibliotecas, módulos e está sendo falado (e utilizado) por muitos desenvolvedores. É óbvio que os desenvolvedores gostam dele. Penso que isso acontece pela facilidade de começar a utilizá-lo: são necessários apenas 5 minutos para ter algo não trivial funcionando e ao mesmo tempo é muito poderoso. Assim, a curva de aprendizado começa a se tornar íngreme conforme se move das coisas mais básicas para o núcleo do framework, quando se descobre que ele é muito complexo.

Julien Knebel: Venho desenvolvendo com Ember.js há mais de 2 anos e em 5 aplicações em produção. Diria que ele oferece flexibilidade, rapidez e confiança, então posso focar mais em definir toda a experiência do usuário, ao invés de investir grande quantidade de tempo em funcionalidades complexas como, por exemplo: um roteador assíncrono robusto.

Brad Dunbar: Meu favorito é o Backbone. Porque provê simplicidade e flexibilidade que os outros não oferecem. Além disso, e talvez mais importante, seu código é geralmente simples e legível.

John Munsch: Prefiro o AngularJS, que utilizo há um ano, e a lua-de-mel com ele ainda não acabou. O problema fundamental cuja resolução busco em qualquer framework de front-end é a capacidade de ajudar a construir uma interface de usuário complexa no front-end, não somente comparável, mas melhor do que aquelas que pudemos construir no passado com frameworks back-end como JSP, PHP, Rails, etc.

Julio Cesar Ody: Na maioria das vezes uso Backbone e venho utilizando cada vez mais o React.js como substituto das views do Backbone. Todas as aplicações web que construo executam completamente em navegadores e se comunicam com um servidor utilizando uma API ou WebSockets. Isso significa que muita lógica acaba sendo escrita em JavaScript e o Backbone ajuda a manter as coisas ordenadas e manuteníveis.

Thomas Davis: Minha primeira experiência com frameworks JavaScript MVC foi no final de 2010 e naquele tempo estava decidindo entre utilizar o Backbone.js ou o Sproutcore. Fiz um post comparativo que chegou a página principal do Hacker News e recebeu mais de 100.000 visualizações. Baseado no feedback que recebi naquela época, decidi utilizar o Backbone.js, pois era compacto e não impedia de implementar funcionalidades complexas sem estar preso às convenções de um framework maior. Conforme aprendia o Backbone.js, escrevi tutoriais que deram um entendimento mais concreto do framework. Estes posts receberam e ainda recebem milhões de visualizações. Os arquivos atuais estão em backbonetutorials.com.

Atualmente, ao construir um produto, a experiência do usuário e o design são cruciais para a necessidade de manter o site relevante. Aplicações de uma única página (Single Page Applications) lhe permitem implementar experiências que as páginas tradicionais não permitem devido à atualização da página. Por exemplo, imagine utilizar o Facebook e ter de atualizar a página a cada curtida ou comentário.

InfoQ: Com tantas alternativas disponíveis, como comparar um framework com os outros?

Julien Knebel: O Ember.js possui desempenho muito bom! Nunca imaginei que fosse codificar aplicações quase que inteiramente sozinho. O Angular é realmente muito bom, e gosto da forma rápida que se começa a utilizá-lo, entretanto, acredito que ele tende a se tornar mais e mais tedioso conforme a sua base de código cresça. Por outro lado, tive mais dificuldade com o Ember no início, mas se tornou mais simples conforme fui aprendendo. Outro ponto importante é a componentização. Não gosto da forma que as diretivas do Angular lidam com isto e os componentes do Ember parecem realmente estar no caminho certo na forma de tratar componentização.

Brad Dunbar: Acho muito difícil uma biblioteca JavaScript preencher todas as necessidades da minha aplicação. Isso significa que frequentemente acabo escrevendo algum código no entorno delas. A funcionalidade do Backbone é pequena e provê utilitários para composição, ao invés de utilizar grandes blocos que não são exatamente o que preciso.

John Munsch: Esta é uma pergunta difícil, minha experiência envolve apenas o Backbone.js (menos de dois anos de experiência) e o AngularJS (um ano). Mas entre estes dois, não acho que seja uma escolha difícil.

O Backbone.js estava funcionando bem, mas tivemos um grande projeto com mais de 6 pessoas trabalhando juntas e o observamos problemas em todas as grandes lacunas do Backbone.js (associação de duas vias – two way binding, validação, templates, AMD, etc) que são normalmente fornecidas com outros softwares. Se não definir todas as soluções para estas lacunas antes de iniciar o projeto, cada desenvolvedor vai procurar preenchê-las da sua maneira. Se estiver com o cronograma apertado e escrevendo muito código rapidamente, pode ser muito difícil capturar toda esta divergência e aplicar consistência.

Terminamos com um projeto funcionando sendo muito mais rápido que a solução anterior, mas era uma Quimera: cada parte parecia ser um animal diferente. Com o AngularJS, muitas destas partes são do próprio framework e há menos espaço para este tipo de problema.

Julio Cesar Ody: Diria que o Backbone é o mais compacto, mas não é um ponto particularmente útil para focarmos. De forma geral, o framework que mais lhe agradar vai ser também aquele que vai lhe tornar mais produtivo.

Então, entre Backbone, Angular e Ember não há como errar. Estou deixando o React.js de fora, pois é somente uma biblioteca de componentes de interface e não deve ser confundido com frameworks de aplicação.

Thomas Davis: Muita coisa mudou desde 2010 e, mesmo tentando constantemente quebrar meus velhos hábitos, ainda uso o Backbone.js atualmente. Ele continua compacto e não mudou nenhum de seus conceitos ou métodos centrais. Embora na minha opinião, ele esteja na eminência de ser substituído pelos seus competidores.

Os esforços realizados em bibliotecas como o Angular para permitir aos desenvolvedores um controle mais intuitivo sobre o DOM fazem o Backbone.js parecer arcaico quando se trata de renderização. Com o novo movimento de JavaScript modular também podemos chamar o Backbone.js de monolítico no aspecto de tamanho.

Não existem razões para os modelos (models), coleções (collections), roteadores (routers) e visões (views) serem empacotadas na mesma biblioteca. Alternativas como o Components.js estão escrevendo estas partes como módulos individuais. Isso não está completamente finalizado no momento e estou considerando utilizá-lo nos próximos projetos.

InfoQ: Construir rapidamente aplicações mais robustas de front-end é interessante, mas e o desempenho? Especialmente em mobile, qual é sua experiência?

Igor Minar: Acho que o mobile ainda é uma área mal servida para o desenvolvimento web. É necessário aplicar muito mais foco para esta área e o Angular está definitivamente se reposicionando para o mobile em sua versão 2.0.

Julien Knebel: O Ember é mais pesado que os demais, isto é um fato. Mesmo se minificá-lo e comprimi-lo, então isto é algo importante a se considerar principalmente se está planejando construir aplicações web para mobile. Ter de lidar com transações de 60 FPS (frames por segundo) de forma suave é difícil, seja implementando em Ember ou em qualquer outro framework. Diria que é necessário saber como lidar com os problemas comuns de desempenho na renderização, caso contrário vai enfrentar problemas independente do framework que escolher.

Brad Dunbar: O Backbone possui alto desempenho e é compacto em relação aos demais. Não tenho encontrado problemas ao utilizá-lo para mobile, apesar de não construir aplicações para dispositivos com mais de uma geração de idade.

John Munsch: Ainda não estamos focados em mobile para o meu site atual, mas já fiz isto para o meu empregador anterior. Como eles queriam executar uma interface web adicional em leitores de códigos de barras em depósitos e hangares, criamos uma interface de usuário (IU) secundária, composta de paginas construídas exclusivamente para os leitores de código de barras. Eles usaram a mesma tecnologia usada na UI para navegadores de desktop e as páginas chamavam as mesmas APIs de back-end, mas eram menores e mais simples para as pequenas telas sensíveis ao toque dos leitores de código de barras.

Naquele ambiente, essa pareceu uma solução melhor do que tentar fazer uma versão responsiva das páginas existentes no qual teríamos que abandonar 80% da funcionalidade no mobile. Tivemos muita sorte de leitores de código de barras de boa qualidade com navegadores HTML5 estarem disponíveis antes do projeto entrar em produção. O trabalho em desenvolvimento foi realizado com leitores de código de barras que possuíam apenas o IE 5.5 disponível.

Se tivesse de voltar ao projeto hoje, a primeira coisa que provavelmente faria seria adicionar o Grunt para concatenação, minificação e revisão do JavaScript para acelerar o desempenho no mobile e no desktop e também ativar a expiração dos cabeçalhos para que o cache do navegador pudesse ser ativado para um mês ou mais.

Julio Cesar Ody: O JavaScript é rápido e atualmente os navegadores são muito eficientes e continuam se aperfeiçoando. Seria necessário fazer uma grande bagunça para tornar as coisas realmente lentas.

O desempenho é muito importante quando tratamos de boa experiência de uso. Mas o JavaScript é apenas uma parte da história. Como se utiliza o CSS (transições e animações) e o quão bom e leve é sua marcação HTML (markup) também podem exercer um grande impacto sobre o desempenho. É preciso ser cuidadoso com o todo.

Se já programou qualquer micro controlador em C talvez saiba o que significa escrever programas que executam em dispositivos limitados. Apesar dos celulares serem computadores poderosos, eles ainda são significativamente mais lentos que o laptop médio. Não é possível se dar ao luxo de cometer tantos erros e precisa levar em conta a regra que mencionei antes e ter cuidado redobrado.

Medir coisas que são problemáticas ajuda bastante. E nesse contexto, o Chrome DevTools me vem à mente.

Thomas Davis: Não existe bala de prata na escolha entre mobile e desktop para abordagem de desenvolvimento de front-end no momento. Às vezes aplicações nativas são melhores, as vezes aplicações JavaScript e outras vezes páginas geradas pelo servidor. Embora por questões de eficiência relacionada ao DOM em geral, provavelmente faria uma aposta segura no React. O React implementa um DOM virtual que difere do DOM dos navegadores e somente faz alterações quando necessário buscando alto desempenho de renderização. Pelo fato do DOM ser virtual, também pode renderizar o DOM no servidor e acessá-lo programaticamente.

InfoQ: Qual é o seu fluxo de trabalho utilizando o framework escolhido? Quais ferramentas usa para desenvolver e depurar?

Igor Minar: Webstorm, Karma e Chrome.

Julien Knebel: Uso Grunt como ferramenta de construção (build) para pré-processar, pré-compilar, pós-processar, concatenar, minificar, etc. O TextMate2 como editor (IDE) há anos. Também gasto bastante tempo depurando no Chrome DevTools com breakpoints. E claro, não viveria sem o Git.

Brad Dunbar: Costumo utilizar uma configuração minimalista incluindo uma interface de linha de comando (Command Line Interface – CLI) e vim/node/npm. Ultimamente, tenho gostado de utilizar o browserify para empacotamento. No fluxo de trabalho, uso o típico ciclo codificar, atualizar, depurar. Nunca fui um grande defensor de escrever os testes antes do código ou coisas desta natureza.

John Munsch: A maioria dos desenvolvedores na equipe de front-end usa WebStorm ou Sublime em computadores Mac, o projeto é construído e executado pelo Maven porquê o back-end é Java e, recentemente começamos a executar o Grunt de dentro das nossas construções Maven para conseguir otimizações do código de front-end. O Jenkins é utilizado para construção, execução de testes unitários e implantação para vários ambientes de testes e de produção.

Os desenvolvedores usam o Karma em suas máquinas para executar em segundo plano os testes unitários em AngularJS que temos atualmente (infelizmente, a cobertura de código atual é de apenas 30%).

Julio Cesar Ody: Faço o máximo possível de design no início. Olhar como a página vai se apresentar ajuda a fazer uma divisão mental dos componentes que preciso escrever.

Recentemente tenho usado muito o Hopla como ferramenta de construção. Ele usa o Sprockets para traduzir CoffeeScript e SASS e compila a aplicação para HTML/CSS/JS estático. Isto é útil para implantações e até para a construção de aplicações Phonegap.

Então, usualmente começo um design implementando uma página HTML estática com estilos usando SCSS, revisando cada parte dela posteriormente e substituindo com componentes JavaScript.

Thomas Davis: Honestamente, sou muito indiferente sobre como as pessoas gostam de desenvolver e depurar, e minhas metodologias mudam projeto a projeto. O único conselho que dou aos desenvolvedores é que usem Asynchronous Module Definition (AMD). O Require.js oferece uma implementação robusta de AMD e na minha experiência, torna a depuração mais fácil em todas as linguagens e ambientes. Ele não só lhe ajuda a estruturar seu código de forma inteligente, mas também reduz para níveis muito baixos a barreira de entrada para a sua base de código, pois tudo precisa ser uma dependência referenciada apontando para um caminho de arquivo.

InfoQ: Conforme uma aplicação cresce, pode ser desafiador manter uma arquitetura robusta e uma grande base de código. Como o seu framework JavaScript favorito escala? Além disso, como ele escala quando uma equipe de desenvolvimento cresce e diferentes pessoas precisam trabalhar em diferentes partes das funcionalidades?

Igor Minar: A reutilização de código, redução de código boilerplate e guias de estilos e convenções são importantes para reduzir a complexidade de grandes bases de código. Mas em nossa experiência no mundo real temos observado que conjuntos de testes de alta qualidade apresentam ainda mais impacto, pois permitem refatorar enquanto mantemos o risco baixo. E refatoração é a chave para mantermos uma grande base de código limpa.

Matteo Pagliazzi: Além da complexidade (como a presença de Serviços, Factories e Providers, que são muito confusos), com simplicidade sendo um objetivo da versão 2 o que pessoalmente não gosto é o mecanismo utilizado pelo Angular para verificar quais propriedades mudaram e atualizar a visualização (mas isto também vai acabar já que Object.observe é implementado nativamente, permitindo que as notificações sejam lançadas a cada alteração, ao invés de checar por alterações em cada uma das propriedades).

Para finalizar: O Angular é realmente muito bom, apoiado por uma comunidade incrível, um enorme número de bibliotecas externas que geralmente satisfazem cada necessidade e mesmo em casos que não encontre o que procura é possível escrever sua própria diretiva. Mas ele também é cheio de conceitos complexos que levam tempo para entender.

Julien Knebel: O Ember lida muito bem com grandes bases de códigos por causa da sua "aplicação de boas práticas" e suas convenções fortes que permitem fazer muita coisa com algumas simples linhas de códigos (também conhecido como convenção sobre configuração). Ele ajuda a cada membro da equipe a depurar o código dos demais de forma mais rápida. Vi alguns desenvolvedores irritados porque eles se sentiam limitados a codificar da forma que o Ember impõe, mas acho que a real frustração vem do fato de que primeiro precisa entender/aprender o Ember antes de tentar fazer coisas legais com ele. E acredito que esse tempo de aprendizado adicional realmente se paga no longo prazo. Além disso, costumo confiar bastante no trabalho de Yehuda Katz e Tom Dale.

Brad Dunbar: Acho que grandes bases de código JavaScript são desafiadoras independente do framework utilizado. Cabe a equipe manter um conjunto de diretrizes e aderir à elas. O uso de módulos de algum tipo (mencionei o npm e o browserify) ajuda muito.

John Munsch: Até agora nossas experiências escalando aplicações tem sido boas. Código de front-end JavaScript escala bem porque existem muitos arquivos estáticos passíveis de otimização, cache e até mesmo CDNs. Praticamente as únicas experiências ruins que tivemos foram tentando renderizar milhares de valores de dados nas páginas. Tentamos encorajar a utilização de paginação e outros ajustes deste tipo, mas não conseguimos chegar a um acordo sobre como utilizá-los e grandes quantidades de dados gerando milhares de elementos no DOM é uma receita para problemas em navegadores antigos.

Outros problemas desapareceram por completo. Por exemplo, o comportamento padrão do AngularJS de descartar as visões (views) e controllers quando alternamos entre visões mantém o uso de memória da outra visão sem impactar a visão atual. Esta é uma solução simples que ajuda os desenvolvedores a evitar diversos problemas conforme eles adicionam mais e mais funcionalidades em uma aplicação de página única.

Julio Cesar Ody: Não acho que nenhum dos frameworks populares tenham um caminho para escalabilidade definido. É o desenvolvedor que deve pensar em algo e executar esta ideia.

Sempre gostei do conceito de componentes/módulos, porque penso em escrever programas da mesma forma que se constrói um relógio. Cada parte (ou componente) tem o seu propósito, e precisa funcionar de forma mais independente possível, expondo uma pequena área de superfície pela qual os outros componentes podem interagir.

Thomas Davis: Como disse anteriormente, desde que use um framework JavaScript modular como o Require.js, escalar e manter a sua base de código é um passeio no parque e só vai depender das suas práticas de codificação. Embora o Backbone.js precise ser dividido em seus próprios módulos neste momento para que, ao invés de requerer o Backbone como dependência, é possível usar apenas Backbone.Model, por exemplo.

InfoQ: Para as equipes que estão considerando adotar um framework JavaScript, quais seriam algumas das armadilhas mais comuns a serem evitadas?

Matteo Pagliazzi: Trabalhando em um grande projeto de código aberto e utilizando o Angular como framework de client-side, descobri que é fácil para os desenvolvedores sem muita experiência encontrarem problemas com herança ou poluir o $scope e o $rootScope com muitos objetos, que a partir de um certo ponto vão começar a deixar a aplicação mais lenta e aumentar o volume de memória RAM utilizada pelo navegador (facilmente usamos mais de 300 MB e isso pode chegar facilmente a 1GB mesmo com um pequeno vazamento de memória). O fato dele tornar a associação de dados (data binding) tão "simples" é fantástico, mas é preciso entender que nem sempre vai funcionar da forma que se espera só porquê ele é "mágico".

Julien Knebel: Todos esses frameworks MV* fazem cálculos pesados no front-end, então, se a aplicação precisar apresentar um grande volume de dados para o usuário, rapidamente atingirá o seu limite de desempenho de renderização (estou falando do limite de 60 FPS). Será necessário tratar da paginação, rolagem de página inteligente; utilizar CSS de forma moderada e inteligente, sequenciamento sutil, definir alguns poucos lugares escolhidos para exibir spinners e fazer chamadas de rede que recuperam dados, etc. Tudo isso importa muito.

Brad Dunbar: Acho que os problemas mais comuns aparecem antes mesmo de utilizar qualquer framework. A parte cliente da aplicação vai precisar de testes, módulos, gerenciador de pacotes e integração contínua, da mesma forma que o servidor. Se conseguir se manter fiel à isto, provavelmente não vai ter problemas.

John Munsch: Mencionei os problemas que tivemos com o Backbone.js, mas existem alguns problemas comuns que sempre podem surgir. Por exemplo, SEO é um fator a ser considerado para páginas geradas inteiramente via JavaScript no front-end. Tenho a sorte de que a maioria do meu trabalho é com SaaS, então SEO não tem sido algo com que nos importemos muito, mas se está construindo algo para um público amplo na web é melhor analisar soluções como: Prerender.io, BromBone, etc. O Google Analytics pode apresentar problemas similares. Nenhum destes problemas é insolúvel, mas é bom que conhece-los e saber que talvez precise escolher uma solução.

O IE8 tem sido sem dúvida um dos maiores problemas. É o último navegador com uma fatia significativa do mercado que não possui um compilador just-in-time para JavaScript. Como resultado, front-ends com muito código JavaScript podem ficar lentos às vezes, podem disparar erros do tipo "o script não está respondendo" quando apresentando muitos dados, etc. O quanto antes pudermos nos livrar do IE8, melhor será para todos.

Julio Cesar Ody: A curva de aprendizado é definitivamente o maior problema que vejo. A maioria das pessoas vêm desenvolvendo aplicações web há anos, que consistem no componente servidor recebendo requisições, processando-as e enviando HTML de volta ao navegador.

Esse é um paradigma completamente diferente que lembra muito o paradigma cliente/servidor e isso não é algo que muitos tenham feito antes. Ele possui muitos dos problemas que o paradigma cliente/servidor tinha, e ter algum conhecimento sobre ele certamente vai ajudar.

Thomas Davis: A única grande armadilha a se preocupar é não ter conteúdo gerado pelo servidor disponibilizado na URL. Isto obviamente vai fazer os motores de busca pararem de indexar o seu website e também vai encontrar problemas para compartilhar sua página. Muitas redes sociais tentam agora analisar o seu website quando os usuários o compartilham para que elas possam buscar informações adicionais. Se não tiver conteúdo gerado pelo servidor disponível no tempo adequado, estas tentativas vão falhar. Embora este problema tenha sido resolvido muitas vezes, recomendo prerender.io como uma boa solução de código aberto. Também tenho escrito bibliotecas SEO para mitigar este problema. Apesar das grandes empresas de motores de busca renderizarem sua aplicação para busca de conteúdo, algumas pessoas especulam que todo o projeto Chromium é uma tentativa da Google de ser capaz de carregar e executar todas as páginas para que eles possam amarrá-lo ao Google Bot e indexar todo o conteúdo apresentado pelo JavaScript.

InfoQ: Para quem quer começar a trabalhar com algum dos frameworks, qual seria o caminho de aprendizado mais rápido e eficiente? Gostaria de recomendar algum material?

Igor Mintar: Existem muitos livros bons sobre Angular, listas como a ng-newsletter.com e muitos podcasts ótimos no egghead.io.

Julien Knebel: Os guias oficiais do emberjs.com são ótimos, cada um deles foca em um recurso específico do framework. Escrevi uma introdução para iniciantes na Smashing Magazine (quando o Ember 1.0 foi lançado) que, na minha opinião, continua relevante.

Brad Dunbar: Talvez não seja a coisa mais legal a apontar, mas sempre recomendo a leitura do código fonte, porque é lá que a ação acontece. A leitura do código trará muita familiaridade com a sua escolha de framework MVC, então se não tiver estômago para encarar o código fonte talvez deva considerar utilizar outro framework.

John Munsch: Se a pessoa em questão já for familiar com JavaScript recomendo pegar um pequeno projeto (quatro ou menos telas (views) distintas formando uma aplicação web) e implementar o projeto inteiro usando o framework em questão. Construindo algo, mesmo algo muito simples, por todo o caminho até a conclusão e implantação. É muito diferente de "brincar" com o framework, porque será necessário resolver alguns problemas específicos e começar a entender algumas coisas que de outra forma passariam despercebidas, porque elas não são as partes divertidas de se aprender.

Julio Cesar Ody: Cometa quantos erros puder em projetos experimentais, mas seja **extraordinariamente** consciente sobre eles. O que quero dizer, é que é inútil tentar com unhas e dentes fazer tudo da melhor maneira possível logo de cara, e isso não vai tornar o aprendizado mais rápido.

Então, refatore seu código continuamente, mantendo em mente a procura de um cenário que possa utilizar diversos componentes em conjunto e formar uma aplicação completa, e mantenha o mais independentes possível.

Escrevi um livro gratuito sobre este assunto, com ênfase em Backbone.js, que é claro que vou recomendar.

Thomas Davis: Depende do seu estilo de aprendizado. Geralmente prefiro simplesmente começar a tentar algo direto. Tenho recebido feedbacks fantástico sobre o meu vídeo-tutorial para Backbone.js, que pode ser encontrado no Youtube.

InfoQ: Agora que os frameworks MVC tornaram-se mainstream, como pensam que eles devem evoluir para melhor se adequar às necessidades dos desenvolvedores? Por exemplo, quais funcionalidades ainda faltam?

Igor Minar: Mobilidade e testabilidade.

Julien Knebel: Sincronização de dados entre o back-end e múltiplos clientes em tempo real seria o próximo passo lógico. O Meteor.js parece que vai seguir esse caminho, então certamente vou experimentá-lo em breve.

Brad Dunbar: Costumo submeter código para coisas que acredito que o Backbone precise e nem todas elas são boas :)

John Munsch: A resposta mais fácil é algo que os números do ngmodules.org apresentam sobre soluções que os desenvolvedores acabando procurando constantemente. Qualquer coisa que una frameworks de front-end à frameworks HTML/CSS (como o Bootstrap), upload de arquivos, internacionalização e localização, etc. Se centenas de desenvolvedores tiveram que procurar um recurso e gastaram tempo para marcar em um website que eles o utilizaram, pode apostar que milhares de outros desenvolvedores também usaram o recurso. Esta é a resposta fácil, mas gostaria de conectar duas coisas que não acho que recebem a atenção que merecem, validação e ausência de back-end.

Validação

Validação no cliente é algo geralmente necessário e que pode se tornar muito complexo para algumas páginas. Temos páginas que várias datas precisam ser testadas entre si para garantir que elas ocorrem na sequência correta, e muitas outras variáveis tem várias regras também. Então, enviamos para o servidor via chamada de API e ele tenta fazer as mesmas validações, pois nunca podemos confiar na validação no cliente. Nosso back-end não é escrito em JavaScript, então não podemos usar uma biblioteca de código em comum para fazer a validação e temos de construir dois conjuntos do mesmo código em linguagens diferentes e nos dar o dobro de probabilidade de cometer erros em algum caso extremo.

Por quê não descrever os dados em esquema (schema) JSON e então tornar fácil para utilizá-lo para validação em ambos os lados (cliente e servidor) de uma chamada da API? Gostaria de ver isso poder ser feito de forma simples.

Ausência de Servidor

Houve alguns rumores sobre isso no último ano como o estado da arte (por exemplo, nobackend.org), mas isso está começando a morrer. Mas a ideia parece ainda estar forte com o Firebase, Hoodie, Backendless, etc. Acho que é uma boa maneira dos desenvolvedores familiarizados apenas com o trabalho em front-end construírem uma aplicação completa sem precisar da ajuda de alguém para construir o back-end. Mesmo para aqueles que constroem o front-end e o back-end de uma aplicação de forma confortável, ela provê uma forma simples de prototipar uma ideia ou produzir um MVP.

Julio Cesar Ody: É difícil dizer. Acho que muitas ideias nasceram de equívocos sobre modularidade, e o pensamento de ter cada vez mais recursos é algo que venho historicamente lutando contra.

Mas acho que o React.js tem a ideia correta sobre isso, na medida que ele lhe oferece uma abordagem direcionada à componentes para construção de aplicações, e é difícil fazer algo terrivelmente errado se simplesmente seguir os exemplos. Ele também abstrai alguns problemas complexos como a eficiência do DOM.

Este é o tipo de problema que ninguém quer gastar tempo resolvendo no decorrer do desenvolvimento de uma aplicação, então, este é um passo na direção certa.

Thomas Davis: O ferramental e os frameworks MVC client-side estão fazendo progresso rápido na direção certa. Acredito que as APIs de servidores estão ficando pra trás neste ponto. Não há nenhuma convenção real que as pessoas estão seguindo para construir APIs RESTful, o que torna difícil para as coleções e modelos client-side trazerem os dados de forma eficiente. O log de erros no client-side ainda precisa de melhorias, mas tentativas como o trackjs.com estão fazendo progresso neste ponto. A forma como os eventos são tratados do lado cliente também precisam de melhorias.

Integrantes do painel

- **Igor Minar** é engenheiro de software no Google. É líder do AngularJS, praticante de desenvolvimento guiado por testes, entusiasta de código aberto e hacker.
- **Matteo Pagliazzi** é apaixonado por desenvolvimento de software e contribui com projetos de código aberto.
- **Julien Knebel** é designer de interface autodidata e desenvolvedor front-end vivendo em Paris onde faz freelances para algumas das maiores empresas da França.
- **Brad Dunbar** é programador JavaScript e contribui frequentemente nos projetos Backbone e Underscore.
- **John Munsch** é um desenvolvedor de software profissional com mais de 27 anos de experiência. Atualmente, lidera uma equipe que está construindo front-end em AngularJS para aplicações web modernas, após passar alguns anos fazendo o mesmo tipo de trabalho com Backbone.js, Underscore.js e Handlebars.js.
- **Julio Cesar Ody** é desenvolvedor de software, designer, palestrante e aspirante a escritor que vive em Sydney, Austrália. Trabalha muito com desenvolvimento para web em dispositivos móveis, e construiu um conjunto de ferramentas das quais ele se orgulha muito.
- **Thomas Davis** é o fundador do backbonetutorials.com, co-fundador do cdnjs.com e desenvolvedor para o taskforce.js. Também contribui diariamente para vários projetos e código aberto que podem ser encontrados em github.com/thomasdavis.

Entrevista: Matthew Carver sobre a web responsiva

por Matthew Carver, traduzido por Wellington Soares

A Web Responsiva é uma importante parte da web moderna. O livro "Responsive Web" de Matt Carver fornece uma introdução e diversas sugestões de como iniciar nesse assunto

Em seu livro The Responsive Web, Matthew Carver fornece uma introdução acessível para os projetos web modernos e a importância do design responsivo. O livro é dividido em três seções. A primeira fornece uma breve introdução do que é o design responsivo, por que existe, e algumas das características dos navegadores modernos que permitem a web responsiva existir; e também no caso de "mobile-first" sendo que os sites são projetados para dispositivos móveis antes de começar o projeto desktop.

A segunda seção é sobre como projetar para uma web responsiva. Antes de mostrar qualquer código a ser escrito, Carver explica como apresentar ideias para os clientes. Usando a navegação como um ponto de debate, após isso, os padrões de projeto são apresentados e o pensamento sobre como construir de maneira responsiva. O livro então explica alguns dos fundamentos de como realmente construir um design responsivo e algumas das técnicas modernas de apresentação de informações como a tipografia web.

Na parte final, mais técnicas avançadas são apresentadas como soluções efetivas para problemas em vários navegadores. O livro termina com práticas avançadas para testar um projeto e como aperfeiçoar seu desempenho.

O InfoQ conversou com Carver sobre seu livro os desafios encontrados pelos desenvolvedores web hoje.

InfoQ: Os modernos desenvolvedores web enfrentam desafios com diversos navegadores e dispositivos. Quais conselhos podemos dar para os desenvolvedores que estão lutando para se manter?

> **Carver:** Há diversas maneiras para emular dispositivos e navegadores. Seria ótimo ter um laboratório com muitos dispositivos atuais e com todos telefones celulares, mas essa não é uma realidade para muitos desenvolvedores, por sorte há alternativas. O Chrome, em suas ferramentas de desenvolvimento, oferece um tipo de emulador que enquadra a janela ao tamanho de um dispositivo. Também emula coisas como controles de toques; assim é possível ter uma ideia do que esperar do navegador Chrome nos celulares e tablets. Ao instalar o Android SDK, automaticamente será instalado o emulador oficial do Android. E no Mac OS, instalando o Xcode, automaticamente será instalado um emulador para dispositivos iOS.

O que acha dos frameworks como Foundation ou Bootstrap? Como acha que eles se enquadram na caixa de ferramentas dos desenvolvedores?

> No meu livro discuto Foundation em detalhes e o mostro como sendo uma ótima ferramenta para prototipação, mas recentemente estive experimentando isso no ambiente de produção. O argumento mais comum contra isso é que aumentam seu código e podem restringir o design. Porém, sinto que são necessários no processo.

A construção de sites está ficando cada vez mais complexa, quando vamos usando a web para implementar mais e mais problemas do mundo real, então quando se decide construir um site, às vezes não haverá certeza onde irá parar.

Ter um framework como base no projeto oferece uma solução pré-fabricada que livra o desenvolvedor de resolver problemas de imediato, como "quais botões vão parecer nesse site?". Permite também resolver problemas muito mais complexos como "há um jeito melhor de implementar esse botão?". Acabo customizando 70-80% do framework, mas usar Foundation ou Bootstrap como ponto de partida economiza muito tempo.

Qual sua resposta aos críticos que dizem que o design responsivo causa consumo excessivo de banda e problema de memória em dispositivos móveis?

Acho uma critica válida. O que importa é que o desenvolvimento de web nunca teve uma enorme quantidade de soluções oferecidas que não tivesse um lado controverso. Assim pergunto, já houve alguma solução para o desenvolvimento web que nunca sofreu uma enxurrada de criticas?

Considere o Swift. A Apple tem mais dinheiro guardado do que alguns países e passou anos criando uma linguagem para melhorar o desenvolvimento do iOS e imediatamente teve críticas sobre a qualidade da compilação (por exemplo). Ao encarar isso como desenvolvedor, somos muito criticados. E se quiser saber como é deixar de aceitar mudanças no modo de trabalho, converse com um desenvolvedor do Flash.

Uso de rede e memória estão no orçamento; para completar suas tarefas é necessário gastar esse orçamento. Desenvolvedores podem gastar mais por muitas razões, porém, não é motivo para deixar de lado o design responsivo como um todo. Isso é errado. Há o ditado "Um carpinteiro ruim culpa suas ferramentas". Um projeto responsivo é uma ferramenta para resolver problemas de dispositivos semelhantes na web. A fragmentação de aparelhos é uma realidade também na web – e só porque o projeto responsivo não é perfeito, não significa que ele não valha nada.

Ferramentas integradas com os navegadores continuam avançando. Quais melhorias deixam você mais empolgado? Quais partes são mais úteis para os desenvolvedores de sistemas responsivos?

Acho que esse movimento para construir ferramentas continua fantástico. Sei que não são mais novidade, mas ferramentas como gulp e grunt têm tornado o processo de desenvolvimento bem mais eficiente. A habilidade do Compass em gerar sprites em tempo real é fascinante e ferramentas como Live Reload fazem o processo de front-end se desenvolver muito mais rápido.

InfoQ: Como a Web Responsiva vai mudar no futuro?

Isto é uma pergunta difícil. Em um anexo do meu livro falo sobre "Design consciente do contexto", a ideia de que a interface deve se adaptar ao ambiente e o modelo do usuário. Vi isso sendo discutido em diversos locais, tanto na teoria e na prática como "filosofia responsiva". Para mim, saber que esse mesmo conceito estava sendo explorado independentemente em várias frentes mostra que é uma área relevante.

Acho que temos todas as ferramentas para oferecer uma conexão profunda com a UI para os usuários da web, de tal maneira que se possa incorporar a preferência pessoal deles, ambiente, o momento e a personalidade para o projeto de interfaces.

O que acha das ferramentas Macaw e Brackets quando está projetando e escrevendo código?

Elas são ótimas. Não sei qual deveria usar na produção no momento, é ótimo ter ferramentas que ajudam o desenvolvedor a alterar dimensões e ver o que sobra de espaço. A habilidade de manipular e alterar espaços delimitados em um projeto ajuda o desenvolvedor articular suas ideias num ambiente mais parecido com os navegadores de verdade. Acho melhor os designers utilizarem um conjunto de ferramentas, em vez de uma única aplicação para comandar tudo, como o Photoshop.

Quais suas motivações ao escrever o livro?

Eu queria criar algo para solucionar o que vi como o maior erro no desenvolvimento da web moderna: a colaboração entre áreas/disciplinas. Eu tinha esperança de escrever algo que um designer pudesse ler e entender os desafios de um desenvolvedor, e que ao mesmo tempo permitisse a um desenvolvedor compreender os desafios do designer. Importo-me profundamente com a web e acredito fortemente que podemos usar a web para tornar a vida melhor. Mas isso só pode acontecer se primeiro desenvolvermos sistemas para solucionar problemas, e se tornarmos essas soluções utilizáveis pela maioria das pessoas.

Um bom desenvolvedor pode inventar uma nova forma fantástica de se procurar produtos, que corte os custos dramaticamente. Mas se a solução depende totalmente de uma interface de linha de comando, terá muito pouco impacto. Por outro lado, um designer pode vir com um lindo modelo de interface, mas se o software não funcionar, não adianta. Precisamos de pessoas com essas habilidades para se comunicar e criar soluções que nos levem adiante. Espero que meu livro possa colaborar com isso em algumas situações.

Você acredita que o design responsivo teve impacto no trabalho feito pelos desenvolvedores web de hoje?

Acho que faz parte de todas as funções no processo. Não podemos ignorar a preferência de aparelhos dos usuários; penso que o "design responsivo" já tem seu lugar no processo de desenvolvimento. Acredito que hoje é o processo padrão. Hoje já preciso me justificar quando não incluo layouts responsivos por padrão!

Nota: O livro The Responsive Web está com 40% de desconto usando o código **info40rwd** na *manning.com*. O código está ativo e não expira.

Sobre o autor do livro

Matthew Carver é desenvolvedor front-end, autor e web designer, com experiência em projetos de design responsivo para clientes como American Airlines e Dallas Morning News.

React.js na vida real do Codecademy

por Bonnie Eisenman, traduzido por Rafael Sakurai

O Codecademy adotou o React.js no ambiente front-end de aprendizado. Enquanto muitos exemplos são simples, a autora Bonnie Eisenman aborda como usar o React em um ambiente grande e crítico.

Em agosto de 2014, o Codecademy decidiu adotar as bibliotecas do Facebook para escrever as interfaces de usuários em JavaScript, o React.js, como parte de um inspeção do nosso ambiente de aprendizado. Inicialmente, corremos para corrigir os problemas procurando exemplos de como o React pode ser usado em aplicações complexas, sendo que a maioria dos tutoriais focam em exemplos específicos de demonstrações indo em caminho oposto as discussões mais gerais sobre problemas específicos em grandes aplicações. Esse artigo foca em ambos, uma visão geral do React.js e algumas dicas sobre considerações específicas do seu uso em grandes aplicações web.

O que é React?

Resumindo: o React é uma biblioteca para construir interfaces de usuários com JavaScript. Ao invés da abordagem tradicional de escrever interfaces de usuários (UI), o React trata cada elemento de UI como uma máquina de estados. O React não é um "framework" como no caso do AngularJS. Embora o Facebook algumas vezes descreva o React como o "V do MVC", acredito que essa descrição não é útil, já que as aplicações não precisam utilizar o modelo MVC. O React ajuda a construir rápidas interfaces de usuários que podem tratar interações complexas sem poluir o código.

Após começar a trabalhar com o React podemos ficar menos preocupados com essas funcionalidades:

O React trata o DOM

A manipulação do DOM é custosa. O React veio amplamente como uma forma de resolver esse problema. O React minimiza a quantidade de manipulação do DOM mantendo seu próprio DOM virtual e somente renderizando quando necessário, um feito que permite uma implementação diferente de alto desempenho.

Isso significa que raramente manipularemos o DOM diretamente, ao invés, deixe o React tratar a manipulação do DOM. Essa funcionalidade é a base para muito do design do React. Se sentir que está abusando da API do React ou tentando mudar a sua forma de fazer isso, há muita chance de estar interferindo na forma como o React entende o DOM.

Essa funcionalidade também permite construir a renderização no lado servidor usando o Node.js, que permite facilmente servir páginas que são amigáveis para ao Search Engine Optimization (SEO).

O React pensa declarativamente em componentes

No React, tudo precisa ser subclasse da classe Component. Os componentes tem propriedades (determinados pelos seus pais) e estado (que podem mudar eles próprios, no entanto baseados nas ações dos usuários). Os componentes podem renderizar e se comportar com base somente em seus estados e propriedades; **os componentes são máquinas de estados.** Esse modelo encoraja a construção de UIs modulares e na prática facilitam o trabalho e a razão da UI.

O React casa com as marcações do JavaScript

Embora possa ser meio estranho escrever HTML no JavaScript, no React essa é a forma natural de fazer as coisas. Usar o JSX – puro JS misturado com marcações HTML – é opcional, mas altamente recomendado. O React argumenta que como as marcações são fracamente acopladas com o JavaScript que os controla, eles podem viver bem no mesmo arquivo.

As informações fluem em uma direção

Isso é mais um padrão geral do React do que uma regra restritiva. O fluxo de informações tende a ser unidirecional no React. Revisaremos esse padrão posteriormente assim que consideremos como o fluxo de informação necessita ser tratado em grandes aplicações.

Anatomia de uma aplicação React

Para tornar esses princípios mais concretos, vamos analisar como o React trabalha no ambiente de aprendizado do Codecademy.

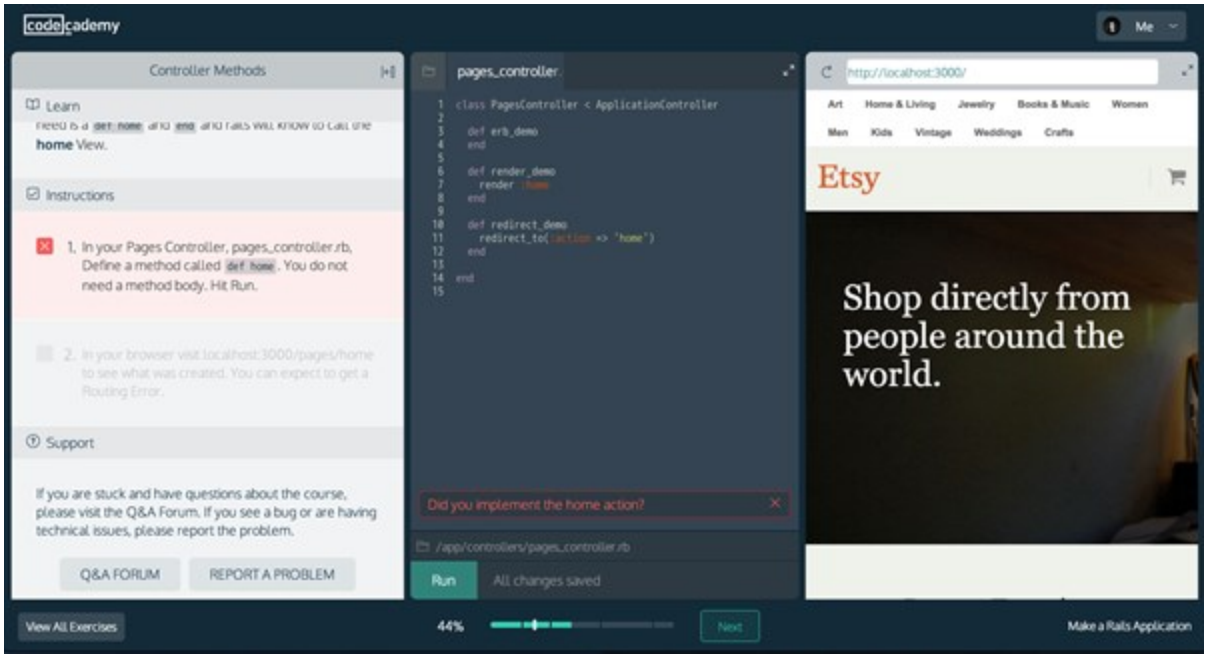

Figura 1: *O ambiente de aprendizado.*

Como podemos ver na imagem, o ambiente de aprendizado principal consiste de muitos elementos de UI diferentes. Alguns elementos como o cabeçalho, menu e navegação estão presentes todo tempo. No entanto, dependendo do exercício, alguns componentes aparecem e desaparecem – o navegador web, a linha de comando e o editor de código podem ser misturados e apresentados dependendo da lição.

A solução lógica é criar componentes React para várias partes. Na imagem a seguir são destacados os principais componentes React:

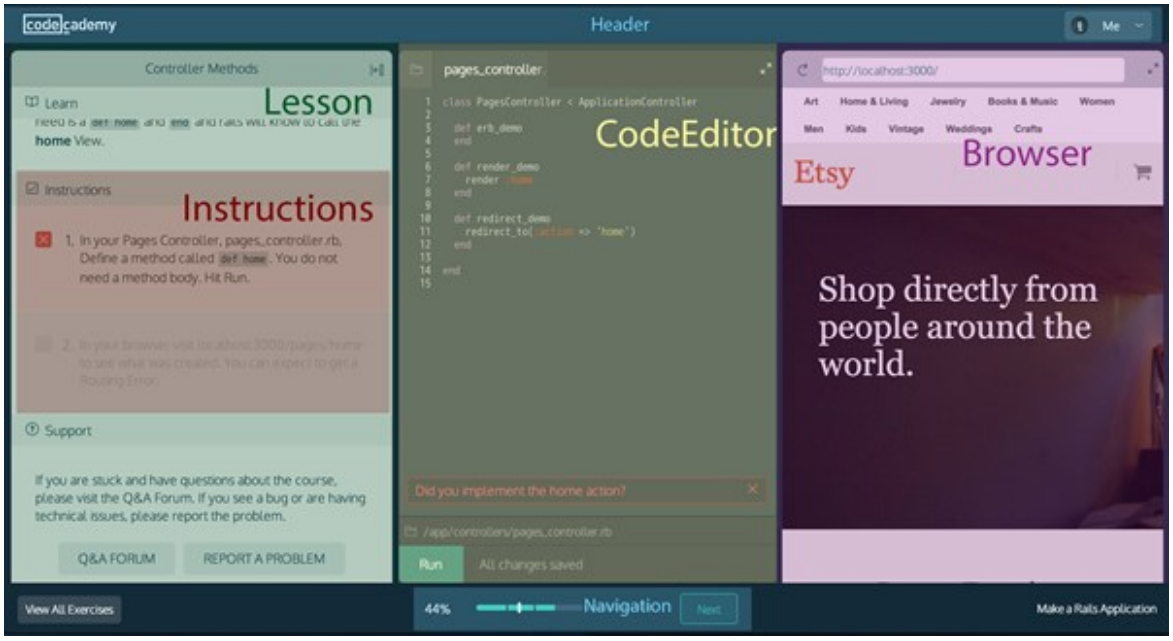

Figura 2: *O ambiente de aprendizado e componentes relacionados. Cada componente também pode conter diversos componentes filhos; por exemplo, o painel de lição da esquerda é atualmente composto por vários componentes:*

Figura 3: *Os subcomponentes que compõem o componente de Lição.*

Nesse caso, deixamos o React determinar o que apresentar no painel de lição. Por exemplo:

- Somente mostrar a botão "Report a Problem" se o usuário estiver autenticado;
- Somente renderiza a "Instructions section" se o exercício tiver testes.

Além disso, o React trata o fluxo de informações entre esse e outros componentes. Há um componente pai para todo o ambiente de aprendizado, que continua tratando o estado, tal como os exercícios do usuário. Esse componente pai dita como seus filhos podem ser renderizados atribuindo suas props (propriedades).

Agora, vamos verificar um exemplo de comunicação de componentes, usando os seguintes componentes nesse componente de árvore bem simplificado:

- LearningEnvironment (Ambiente de aprendizado);
- CodeEditor (Editor de código);
- ErrorDisplayer (Apresentar erros);
- RunButton (Botão de execução);
- Navigation (Navegação).

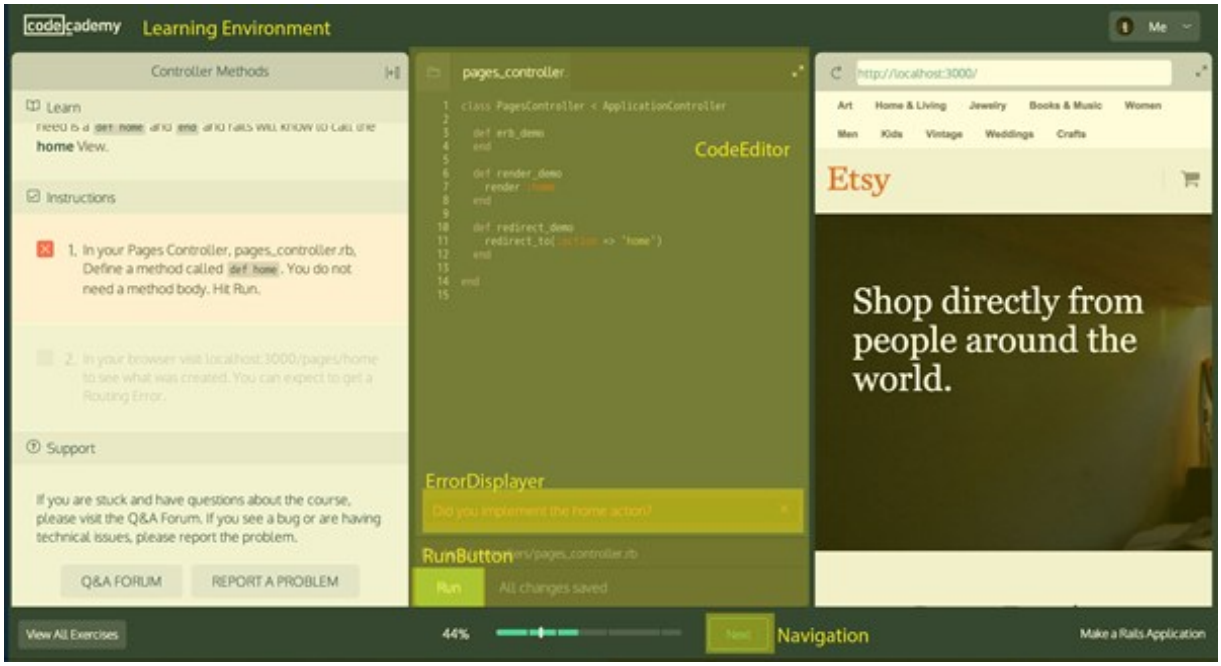

Figura 4: *Alguns componentes envolvidos no código de submissão.*

Como tratamos o fluxo de trabalho de um usuário tentando executar seus códigos? Queremos executar os testes sobre os códigos e apresentar uma mensagem de erro ou permitir sua continuação. Aqui está um possível fluxo:

- Quando o usuário clica no botão Run, ele informa seu pai, o CodeEditor, da ação via callback;
- O CodeEditor, então informa ao seu pai, o Learning Environment, lançando outro callback, passando através do código atual do usuário;
- O Learning Environment executa os testes no código do usuário.

Com base no resultado:

- O LearningEnvironment define a propriedade errorMessage no CodeEditor, que então define a propriedade **errorMessage** no seu filho, o ErrorDisplayers;

- Se o usuário passar em todos os testes do exercício, o LearningEnvironment define a propriedade de progresso no componente Navigation.

As atualizações da UI podem ser executadas com uma simples chamada de função. Se os componentes forem declarados como esse método de renderização do LearningEnvironment (novamente, uma versão simplificada):

```
render: function() {
 return(
   <div>
     <CodeEditor
       error={this.state.error}
     />
    <Navigation
      mayProceed={this.state.mayProceed}
    />
  </div>);
}
```

Lembre-se, o React mistura JavaScript com marcações HTML. Nesse caso, o método **render** está definindo um **LearningEnvironment** como algo que contém ambos componentes **CodeEditor** e **Navigation**.

Podemos atualizar o estado do LearningEnvironment, que acionará uma renderização e atualiza os componentes filhos se necessário:

```
handleTestResult: function(currentIndex, passing, error) {
 this.setState({
   error: error,
   mayProceed: passing && currentIndex === this.state.currentExercise.tests.length-1
 });
}
```

O React trata as atualizações do UI de forma simples e elegante.

Considerações para grandes aplicações – Fluxo da informação

Como notado previamente, o React não utiliza necessariamente o modelo MVC; de fato, estamos livres para tratar o fluxo de informações, mas precisamos de uma estratégia consistente para obter informações. E se não quiser passar para baixo o encadeamento das propriedades dos componentes que não precisam atualmente dele, apenas para alcançar um componente filho do filho do filho, etc? Se aquele nó folha aceitar entradas do usuário, como alertar o componente filho do filho do filho dessa mudança?

Em grandes aplicações, isso pode ser frustrante. Mesmo em um exemplo simples como o anterior, como o botão Run comunica a ação do usuário com o LearningEnvironment? Precisamos passar sempre chamadas de callback, mas é difícil escrever componentes realmente modulares e reusáveis dessa maneira.

A solução do Codecademy vem sendo gerar adaptadores para comunicação que gerenciam o fluxo de informações entre componentes específicos. Ao invés de passar as chamadas de callbacks, os componentes de alto nível, como o CodeEditor também recebem um Adapter, que fornece uma interface única para importantes tarefas de comunicação. Por exemplo, quando o CodeEditor está presente, o LearningEnvironment gerencia um Adapter, que pode emitir e processar os eventos relacionados à submissão de códigos.

O conceito principal tirado disso é que independentemente de como é tratado o fluxo das informações entre a árvore de componentes, a equipe deve entrar em acordo quanto a uma estratégia coerente.

Integração

O React é fácil de usar, mas precisa de algumas ferramentas para aproveitá-lo efetivamente no seu workflow. Por exemplo, usamos:

- Um script que observa mudanças locais em arquivos .jsx e recompilar eles se necessário;
- Um servidor separado de node.js que trata a renderização no lado do servidor;
- As ferramentas do desenvolvedor para auto geração de novos arquivos de componentes conforme a necessidade.

Nenhuma delas são muito complicadas. Gulp é uma ótima escolha para observar os .jsx, mas escrevemos a nossa própria. Para geração de novos arquivos de componentes usamos um simples script bash, que também segue nossa convenção de nomes. Se estiver usando um servidor node.js para renderizar no lado servidor, tenha cuidado; pode ser difícil forçar que o require.js pegue as mudanças no seu código React, então temos nossos observadores que reiniciam o servidor node conforme a necessidade.

Por que React?

Quando estávamos revisando o ambiente de aprendizado, tínhamos que determinar quais ferramentas ou frameworks usar; eventualmente escolhemos o React e estamos muito felizes com a escolha. (Mais detalhes sobre como escolhemos um framework JavaScript, ou não, estão nesta apresentação.

Aqui estão alguns aspectos que gostamos no React:

É testado para a batalha.

O React é usado em produção pelo Facebook e Instagram, então podemos confiar no seu desempenho e confiabilidade. Até agora, ele está servindo muito bem e não temos experimentado qualquer problema significante.

Componentes fáceis de entender

Porque o React trata somente com componentes individuais, que renderizam com base em seu estado interno, é fácil de conceituar o que deve acontecer a qualquer momento. Sua aplicação torna-se efetivamente uma grande máquina de estados. Isso significa que podemos testar partes individuais da UI de forma isolada, bem como adicionar novos componentes sem se preocupar com interferências no restante da aplicação.

SEO Facilitado

Como o React foi construído para suportar a renderização no lado servidor, podemos servir páginas praticamente completas para mecanismos de buscas, que é um grande impulso para o SEO com muito pouco esforço necessário. Sabendo que somente funciona no Node; já que a aplicação principal do Codecademy é escrita em Rails executando um servidor Node separado que apenas trata a renderização do React.

Compatibilidade com o legado e flexibilidade

Enquanto a adoção de um framework inteiro é um grande compromisso, é possível experimentar calmamente a adição do React com o código base já existente. Do mesmo modo, se for necessário podemos remover o React no futuro com bastante facilidade. No Codecademy, decidimos escrever um novo projeto inteiro no React, para testar e aprender quais as melhores formas de aproveitá-lo; isso funcionou muito bem e agora usamos para todos os novos elementos de UI. Recomendamos se aprofundar e construir alguns experimentos – e então considerar como o React pode ser integrado em seu código já existente.

Menos código repetitivo

Menos tempo escrevendo códigos repetitivos significa mais tempo resolvendo problemas interessantes. Dessa perspectiva, o React é conciso e leve. Aqui está o código mínimo necessário para criar um novo componente:

```
var dummyComponent = React.createClass({
 render: function() {
   return (<div>HTML markup in JS, what fun!</div>);
 }
});
```

Curto e direto ao ponto. O que não há para gostar?

Comunidade forte

O React conta com uma comunidade que está crescendo rapidamente. Quando encontramos problemas, há muitas pessoas para discutir. E porque muitas empresas grandes estão usando o React em produção: Facebook, Instagram, Yahoo!, Github e Netflix, são alguns nomes.

Conclusões

O React é uma biblioteca leve, poderosa e testada para construir interfaces de usuários com JavaScript. Não é um framework completo mas sim uma ferramenta que pode mudar a maneira que abordamos o desenvolvimento front-end. Vimos que pode ser uma ferramenta muito útil para o desenvolvimento front-end. Para começar a usar o React, o tutorial é o lugar mais indicado. Há também diversas postagens abordando conceitos do React (como nesses slides). Construa algo e descubra sua opinião!

Sobre a autora

Bonnie Eisenman é engenheira de software da codecademy.com. Graduada recentemente em Ciência da Computação em Princeton, também tem um pouco de conhecimento em hardware e adora trabalhar com Arduinos e programação musical em seu tempo livre. Encontre em **@brindelle** no Twitter.

Caminhando para uma web independente de resolução com SVG

por Angelos Chaidas, traduzido por Rafael Sakurai

Esse artigo examina as vantagens de SVG para arquivos gráficos em projetos web ou mobile

Veremos as vantagens de usar Scalable Vector Graphics (SVG) como formato preferido para arquivos gráficos de qualquer projeto web ou mobile. O objetivo não é impedir os designers e desenvolvedores de usar outros formatos (PNG / JPEG), mas sim demonstrar como o uso do SVG pode melhorar o fluxo de trabalho dos dois principais ciclos de vida de qualquer projeto web: as etapas de design e desenvolvimento.

Designers

Vamos começar investigando o lado do designer.

Ideal para ícones independente de resolução

Ao escrever este artigo, já era inevitável a discussão de "Achatamento do Design". A Microsoft aplicou sua abordagem de design moderno para todos os seus softwares e todas as interfaces de usuário para dispositivos móveis. Com o iOS 7, a Apple tem substituído os seus princípios de aparência e formato, em favor das boas práticas de design limpo e plano. Também não deve ser deixado para trás o lançamento do Google da sua linguagem visual "Material Design", que pode ser usado em qualquer lugar, desde aplicativos Android até sites web.

Figura 1: *Achatamento dos ícones no iOS e Windows. O "achatamento do design": aparência e formato com efeitos de água, sombras e pseudo-3D estão voltando para as formas simples.*

Com os fundos dos componentes que simulam 3D voltando às cores primitivas e os botões esculpidos voltando a ser planos (ambas as abordagens podem ser implementadas com elementos HTML e estilizados com CSS) o foco do design de interface de usuários está caminhando para tipografia, layouts e ícones.

A principal vantagem do SVG – bem divulgada em toda a web – é sua natureza escalável. Durante a montagem de um ícone, o designer não precisa se preocupar sobre como vai aparecer em diferentes dimensões ou em dispositivos móveis com diferentes densidades de pixels. Pode se concentrar apenas em implementar o recurso da melhor forma possível.

Com SVG não há necessidade de exportar dois diferentes formatos de arquivos com base em pixels para exibir em modo retina e não retina. Na verdade, não há necessidade de se preocupar com a densidade de pixels em cada dispositivo, especialmente levando em conta a sua natureza não padronizada. Pode-se manter o foco no trabalho artístico. Uma vez que o ícone está pronto, o designer exporta para um único arquivo no formato SVG, que pode ser redimensionado pelo desenvolvedor sem perda de qualidade.

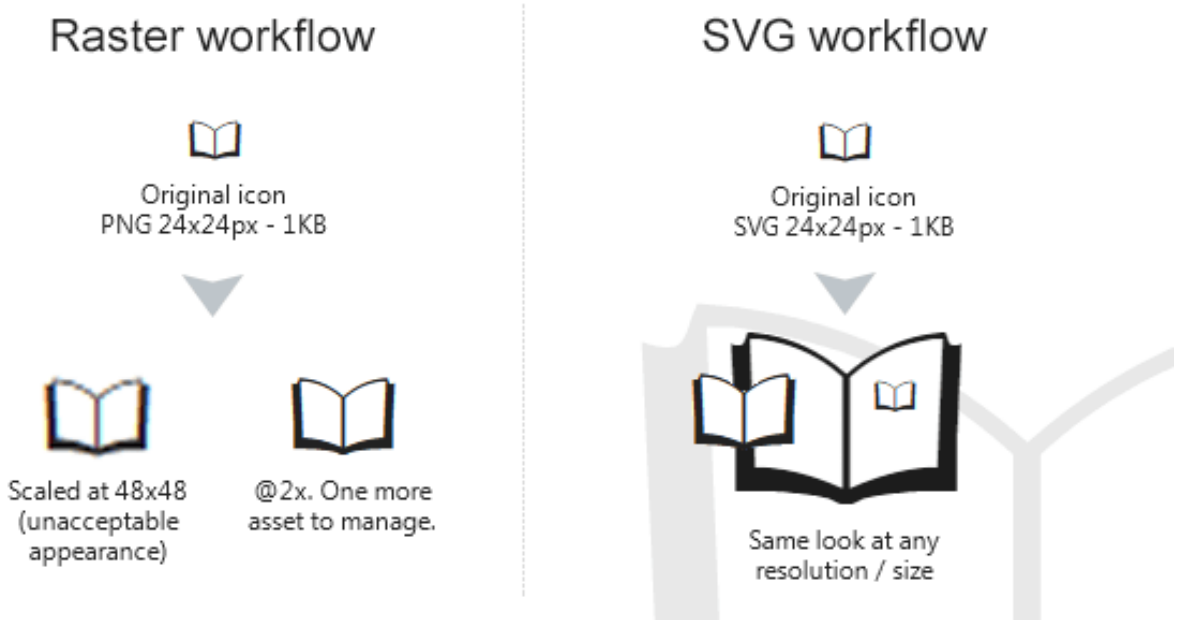

Figura 2: *Fluxo de trabalho com PNG e SVG. Temos um passo a menos no fluxo de trabalho do designer, que não precisa exportar um recurso para uma versão @2x (ou @3x no futuro). O recurso em SVG pode ser redimensionado pelo desenvolvedor sem perda de qualidade.*

Redimensionamento melhorado mesmo em pequenas dimensões

Eis um desafio comum de design para o designer cuidadoso: dentro no Photoshop um ícone parece bonito para dimensão de 44x44 pixels (amigável ao touch), mas ao diminuir para 24x24 ou 16x16, a interpolação bicúbica traz serrilhamento, que pode borrar a imagem. Mesmo com algoritmos inteligentes como a interpolação bicúbica "sharp" disponível nas últimas versões do Photoshop, muitos desenvolvedores terminam desenhando do zero as imagens dos menores arquivos (interface de ícones, favicons etc.) para obter clareza.

Com SVGs, esse cenário é melhorado pois navegadores redimensionam (e rasterizam) os arquivos SVG muito bem. Isso é especialmente verdade para telas de alta densidade, tal como as últimas gerações de dispositivos móveis.

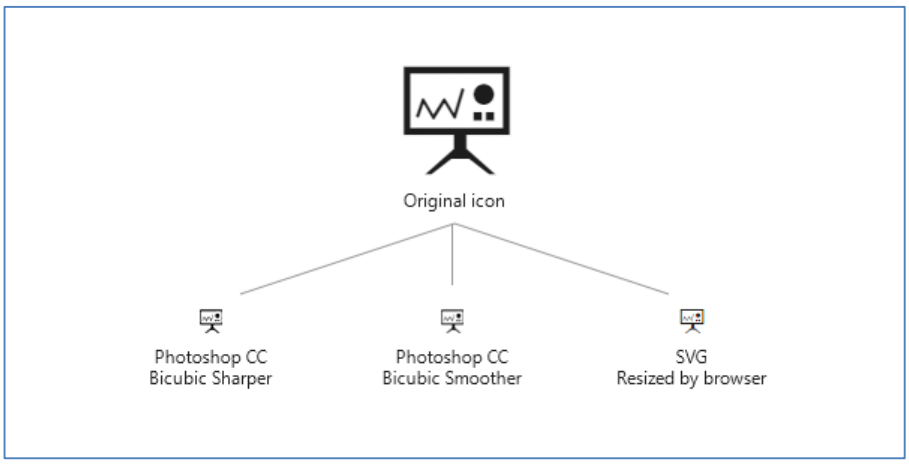

Figura 3: *Tratamento de imagens com Photoshop.*

Os navegadores estão redimensionando cada vez melhor. Em alguns casos, o navegador faz o redimensionamento com aparência melhor que os métodos avançados de interpolação do Photoshop.

Para obter resultados bons nessas dimensões, um designer pode facilmente "empacotar" todos os arquivos SVG como uma fonte de ícones (veja mais detalhes a seguir). Isso influência a capacidade de destaque do subpixel em diversas operações do sistema, resultando em ícones de tamanhos como 12, 14 ou 16 pixels claros como cristal e com contornos nítidos mesmo em antigos navegadores IE.

Elementos SVG podem ser alterados manualmente

Otimizar uma imagem JPEG é um processo sem volta, em que reduzimos a qualidade tentando não perder muita informação. Para itens PNG e GIF, o designer possui um pouco mais de controle através da habilidade para especificar uma paleta de restrições e também reduzir as informações de encode no arquivo.

Com o SVG, o designer meticuloso pode optar por usar formas primitivas, reduzindo o número de vértices em elementos de formas, bem como ter textos embutidos no SVG, ao invés de convertidos para esboços. Isso pode resultar em um arquivo SVG menos complexo e com tamanho reduzido.

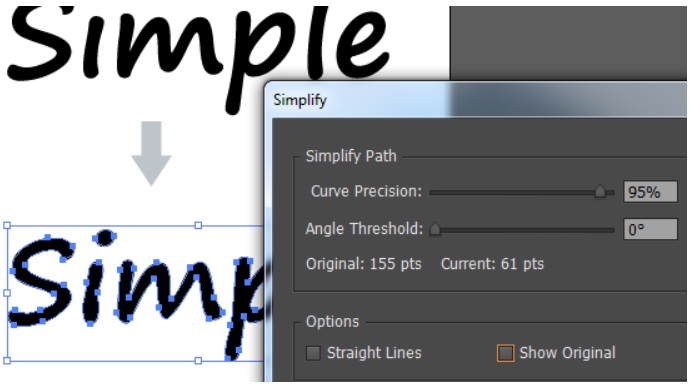

Figura 4: *O "Simplify" do Illustrator em ação. As vértices que marcam um caminho vetorial podem ser reduzidos com perdas mínimas de precisão, obtendo um arquivo SVG de menor tamanho.*

Para ainda mais otimizações, propriedades do SVG como comentários, ids ou tags (<g>) de agrupamento redundantes podem ser removidas via edição direta do código SVG.

Ferramentas de código aberto

Como alternativa ao Photoshop (e talvez Fireworks e Sketch), o mundo do código aberto oferece o editor Gimp. O Gimp é uma ferramenta valiosa com uma comunidade animada, mesmo quando comparado com as ferramentas proprietárias. Mas ainda é inferior em algumas áreas. Substituir o Photoshop pelo Gimp não é uma tarefa simples e geralmente resulta em um fluxo de trabalho mais complexo.

Para arte vetorial também temos o Inkscape, uma alternativa profissional de código aberto ao Illustrator da Adobe. Permite que os designers criem e editem vetores com múltiplos recursos e ferramentas, de forma similar aos usados nas ferramentas oferecidas pela Adobe. Muitas das ações de "dia a dia" com vetores podem ser feitos bem no Inkscape, como operações booleanas que combinam formas, simplificação de caminhos, compensação de caminhos dinâmicos, edição de subcaminhos e muito mais.

Desenvolvedores

Nas próximas seções, vamos explorar o lado do desenvolvedor.

Independência de resolução e reusabilidade

Para exposição de altas densidades como telas de retina e dispositivos móveis, a necessidade de arquivos de alta resolução "@2x" se vão. Com as metatags apropriadas e com as dimensões dos arquivos especificados em arquivos CSS, um desenvolvedor pode ter controle completo quando for redimensionar os arquivos, sem a necessidade solicitar para o designer as variações do mesmo arquivo.

Há alguns fatos emocionantes em ter um único arquivo, tal como: company-logo.svg, pois pode ser usado em vários locais da interface de usuário, com as dimensões sendo controladas pelo CSS ou por atributos de width e height, e como resultado a mesma aparência limpa como cristal independente de como será escalado (ou rotacionado).

Animação

A animação de SVGs com CSS3 mantém a nitidez e clareza ao longo de toda duração da animação. Alguns exemplos são apresentados a seguir, mas note que a baixa quantidade de frames da animações gif não justifica a aceleração suave da GPU na finalização dos efeitos.

Figura 5: *Animações com CSS3 do site Adzuna. (Para ver uma demonstração ao vivo, visite a página da* ***www.adzuna.co.uk*** *e mantenha o foco nos campos de textos ou clique em nos links "more" da lista de botões.)*

Mas isso é apenas o inicio. A sintaxe XML estruturada do SVG permite que um desenvolvedor possa programaticamente selecionar e animar elementos individuais de um arquivo SVG para criar scripts de animações, banners, propagandas, etc. O CSS-Tricks tem um excelente tutorial passo a

passo para criar um banner animado no formato SVG e mostra como – quando colocado em linha – os elementos individuais de uma imagem SVG podem ser manipulados com CSS para seguir um determinado script, resultando em um banner animado similar aos que são usados na web com o Flash da Adobe que já foi um padrão da indústria para propagandas animadas.

Interatividade

O potencial do SVG de substituir o Flash da Adobe não para com as animações. O conteúdo do SVG pode ser interativo, com eventos como: click, mousedown, mouseup (e alguns outros interessantes como SVGZoom / SVGResize) disponíveis para que o desenvolvedor programaticamente interaja usando JavaScript.

Isso abre uma lista de possibilidades como infográficos interativos e soluções gráfica para empresas.

Indo além, o desenvolvedor pode usar bibliotecas auxiliares que facilitam o desenho, manipulação e interatividade, algumas sugestões são: RaphaelJS, D3.js e Snap.svg. Com bibliotecas como essas, animação complexas e apresentações iterativas que previamente eram feitas apenas em Flash agora são possíveis, com diversos de exemplos disponíveis para inspiração, como: a estrutura de árvore customizada do NY Times para analise política das campanhas do Obama e Romney, Gráficos hamiltoniano, Gráficos orientado por potência, Mapas com zoom interativos do Censo Argentino de 2001-2010 e diversos outros.

Uso em linha (inline)

Voltando aos detalhes da implementação. As duas abordagens padrões para usar os arquivos de imagens em projetos web são realizados pela tag <img> e como imagem de background em elementos (geralmente a nível de blocos). Como o SVG é um XML, uma terceira possibilidade disponível para o desenvolvedor é o uso em linha (inline).

```
▼<form action="http://www.adzuna.co.uk/search" method="GET">
   <label for="wa" id="wa_l">What?</label>
   <input type="text" name="q" id="wa" autocomplete="off">
   <span id="wa_helptext">e.g. job, company, title</span>
  ▼<svg id="wa_i" x="0px" y="0px" width="26px" height="26px" viewBox="0 0 26 26" xml:space
"preserve">
     <path d="M13,26C5.8,26,0,20.2,0,13C0,5.8,5.8,0,13,0c7.2,0,13,5.8,13,13C26,20.2,20.2,2
     M13,2.6C7.3,2.6,2.6,7.3,2.6,13c0,5.7,4.7,10.4,10.4,10.4c5.7,0,10.4-4.7,10.4-
     10.4C23.4,7.3,18.7,2.6,13,2.6z"></path>
     <path d="M13,21.1c-4.5,0-8.1-3.6-8.1-8.1c0-4.5,3.6-8.1,8.1-
     8.1c4.5,0,8.1,3.6,8.1,8.1C21.1,17.5,17.5,21.1,13,21.1z M13,7.5c-3,0-5.5,2.5-
     5.5,5.5c0,3,2.5,5.5,5.5,5.5s5.5-2.5,5.5-5.5C18.5,10,16,7.5,13,7.5z"></path>
     <path d="M16.1,13c0,1.7-1.4,3.1-3.1,3.1c-1.7,0-3.1-1.4-3.1-3.1c0-1.7,1.4-3.1,3.1-
     3.1C14.7,9.9,16.1,11.3,16.1,13"></path>
   </svg>
   <label for="w" id="w_l">Where?</label>
   <span role="status" aria-live="polite" class="ui-helper-hidden-accessible"></span>
   <input type="text" name="w" id="w" autocomplete="off" class="ui-autocomplete-input">
   <span id="w_helptext">e.g. city, county or postcode</span>
  ▶<svg x="0px" y="0px" width="20px" height="26px" viewBox="0 0 20 26" xml:space="preserve
```

Figura 6: *Código fonte do SVG. O código SVG pode ser colocado diretamente no HTML. Os navegadores antigos simplesmente ignoram ele, os navegadores recentes irão renderizar como um recurso vetorial. Os arquivos agora são "alcançáveis" pelo CSS (#wa_i) e no Javascript (document.getElementById('wa_i')).*

Com o HTML5, o elemento <svg> pode ser colocado diretamente no código HTML das páginas. O beneficio aqui é que o elemento <svg> bem como qualquer elemento filho pode ser controlado pelo CSS.

Independentemente do tamanho e posição, cores de preenchimento e traço também podem ser manipulados, e mesmo animados. Além disso, certos atributos somente do SVG (tal como: stroke-dasharray e stroke-dashoffset) podem ser manipulados a partir de um arquivo CSS resultando em diversas possibilidades interessantes para animações como os efeitos de animações de linhas.

Mais formas de sprites

Com diversos arquivos, uma abordagem de otimização clássica é a redução de requisições HTTP combinando diversas imagens em um simples "sprite" (geralmente um PNG) e usa as propriedades de background do CSS para estilizar diversos elementos HTML com o mesmo sprite.

Com os arquivos SVG, duas possibilidades extras que são interessantes agora estão disponíveis aos desenvolvedores:

- Abordagem de sprite clássica: Um recurso SVG com dimensões fixadas pode conter múltiplos sub elementos, colocados em coordenadas especificas. O desenvolvedor usa os arquivos como imagens de background e reposiciona para apresentá-los conforme a necessidade.
- Abordagem de agrupamento em linha: Com a abordagem em linha, a habilidade para controlar e apresentar os sub elementos como CSS permite que o desenvolvedor possa criar um SVG como um "conjunto" de arquivos, no qual cada grupo tem um elemento tal como um ícone. Os IDs podem ser atribuídos para grupos individualmente e pode mudar as propriedades de apresentação de cada grupo, os desenvolvedores escolhem o que ocultar e o que apresentar. Essa técnica funciona muito bem com elementos de interface que tem as mesmas dimensões, tal como os ícones da interface de usuário.
- Empacotamento como fonte de ícone: Os desenvolvedores também tem a opção de "empacotar" diversos arquivos SVGs juntos em uma fonte de ícone. O suporte dos navegadores é excelente (mesmo no IE6) e os mecanismos de subpixels dos navegadores modernos fazem com que mesmo os menores tamanhos fiquem constantes e limpos como cristal. Mesmo nos melhores, há múltiplos ícones online de geradores de fontes disponíveis que simplificam o processo de empacotamento.

Comparando a versão dos mesmos arquivos

O formato SVG, por sua virtude de ser essencialmente um arquivo texto, presenteia o desenvolvedor com a interessante possibilidade de não apenas comparar os arquivos visualmente, mas também verifica as diferenças dos arquivos, assim conhecendo quais partes do SVG foram modificados.

No caso de arquivos SVG grandes e complexos, tal como: infográficos, a comparação de teste é ótima para entender quais partes do recurso foram modificadas na nova versão.

Fontes para aprendizado

Para os desenvolvedores, recomendo o Compedium of SVG Information de Chris Coyers. É uma grande lista de links para arquivos SVG – divididos em seções lógicas – o resumo é ótimo um ponto inicial para qualquer pessoa interessada em SVG.

Para designers, a apresentação "Leaving Pixels Behind" de Todd Parkers é a melhor introdução possível, cheio de GIFs animados apresentando o fluxo de trabalho com SVG no Illustrator.

Suporte a antigos navegadores

No momento da escrita desse artigo, o Internet Explorer 8 ainda precisa ser tratado pelos desenvolvedores. Quando essa necessidade é combinada com o fato dos navegadores Android pre-v3 que não suportam SVG, uma solução alternativa precisa ser implementada.

Para os desenvolvedores, o Modernizr é a ferramenta escolhida. Incluir as bibliotecas do Modernizr como <script> externo no <head> de qualquer projeto web adicionará as classes apropriadas nos elementos <html> para dizer para o documento carregar a página. Em seguida, é necessário adicionar algumas definições extras aos estilos CSS que atualizam as imagens SVG em background com formatos antigos, ou mesmo no caso dos SVGs em linha ou colocados em tags <img>, assim usando tags "auxiliares" que contém soluções alternativas e que são ocultadas por padrão.

O desafio aqui é não ter que pedir para o designer ter que exportar outros formatos de recurso uma vez que esses navegadores invalidam essa opção de exportar apenas um recurso SVG.

Felizmente, as ferramentas de automatização como o Grunt e especificamente os Grunticon do Filament Group estão aqui para ajudar. Em resumo, o Grunticon recebe como entrada as pastas de arquivos SVG e como saída tem uma lista de arquivos PNG junto com arquivos CSS que referenciam essas imagens PNG. Para os gurus da linha de comando, o Inkscape pode ser usado através do shell script para também converter os arquivos SVG para qualquer formato.

Conclusões

As vantagens de usar vetores na web são muito maiores que as desvantagens. Com suporte excelente dos navegadores e soluções automatizadas para funcionar nos navegadores antigos. Acredito que o futuro da UI com vetores gráficos independentes de resolução é a melhor alternativa.

Sobre o autor

Angelos Chaidas trabalha como desenvolvedor front-end sênior na Adzuna, uma engine de pesquisa de vagas de emprego internacional. Começou sua carreira em 2000 como designer e desenvolvedor PHP, mas focou em desenvolvimento front-end e JavaScript nos últimos oito anos. Adora design de UI e UX de aplicativos móveis; é apaixonado por otimização web e palestra em eventos locais sobre JavaScript. Pode ser encontrado no Twitter como @chaidas.