APIs web do lado cliente

Samuel Amaro

9 de maio de 2021

Se você decidir usar JavaScript no lado do cliente para sites ou aplicativos, você rapidamente vai se deparar com as **APIs** - interfaces para manipular diferentes aspectos do navegador e do sistema operacional em que o site está sendo executado, ou mesmo dados de outros sites ou serviços. Neste módulo, descobriremos o que são APIs, e como usar algumas das APIs mais comuns, que serão úteis no seu trabalho de desenvolvimento.

Introdução a APIs para a web

Primeiro, vamos começar com apis de alto nível — o que elas são, como elas funcionam, quando usar no seu código, como elas são estruturadas? Nós veremos diferentes tipos de classses principais e o que elas são, e quais são as possibilidades de uso.

Introdução ás Web APIs

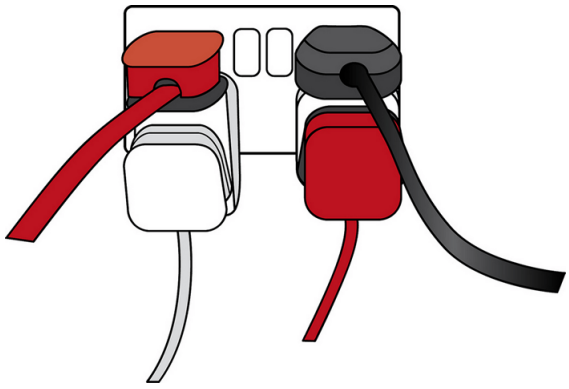
Primeiro, vamos ver as APIs a partir de um nível mais alto: o que são, como funcionam, como usá-las em seu código e como são estruturadas? Ainda, vamos entender quais são as principais classes de APIs e quais usos elas possuem.

Objetivo: Familiarizar-se com APIs, o que elas podem fazer, e como usá-las em seu código.

O que são APIs ?

As APIs (Application Programming Interfaces - Interfaces de programação de aplicativos) são construções disponíveis nas linguagens de programação que permitem a desenvolvedores criar funcionalidades complexas mais facilmente. Tais construções abstraem o código mais complexo, proporcionando o uso de sintaxes mais simples em seu lugar.

Pense no seguinte exemplo: o uso de energia elétrica em sua casa ou apartamento. Quando você deseja utilizar um eletrodoméstico, você precisa somente ligar o aparelho na tomada. Não é preciso conectar diretamente o fio do aparelho diretamente na caixa de luz. Isso seria, além de muito ineficiente, difícil e perigoso de ser feito (caso você não seja eletricista).



plug de tomada

Da mesma forma, caso você queira programar gráficos em 3D, é muito mais fácil usar uma API escrita em linguagem de alto nível como JavaScript ou Python, do que escrever em código de mais baixo nível (C ou C++) que controla diretamente a GPU ou outras funções gráficas.

APIs JavaScript client-side(lado do cliente)

A linguagem JavaScript, especialmente client-side(lado do cliente), possui diversas APIs disponíveis. Elas não fazem parte da linguagem em si, mas são escritas sobre o core da linguagem JavaScript, fornecendo superpoderes para serem utilizados em seu código. Geralmente, tais APIs fazem parte de uma das seguintes categorias:

- **APIs de navegadores:** fazem parte do seu navegador web, sendo capazes de expor dados do navegador e do ambiente ao redor do computador

circundante, além de fazer coisas úteis com esses dados. Por exemplo, a API Web Áudio fornece construções JavaScript simples para manipular áudio em seu navegador - pegar uma faixa de áudio, alterar o volume dela, aplicar efeitos, etc. Por trás dos panos, o navegador utiliza códigos complexos de baixo nível (ex: C++) para realizar o processamento de áudio de fato. Como foi dito anteriormente, essa complexidade toda é abstraída de você pela API.

- **APIs de terceiros:** geralmente, não fazem parte do navegador e você precisa recuperar seu código e suas informações de outro local da web. A API do Twitter, por exemplo, permite mostrar os seus últimos tweets no seu site. Ela fornece um conjunto de construções especiais para ser usado de maneira a consultar o serviço do Twitter e retornar informações específicas.

Relacionamento entre JavaScript, APIs, e outras ferramentas JavaScript

Na seção anterior, abordamos as APIs JavaScript client-side(lado do cliente) e como elas se relacionam com a linguagem JavaScript. Esse ponto merece uma revisão e também uma breve menção de como outras ferramentas JavaScript encaixam-se nesse contexto:

- JavaScript — linguagem de alto nível, embutida em navegadores, que permite implementar funcionalidades em páginas web/aplicativos. A linguagem também está disponível em outros ambientes de programação, tais como o Node.
- APIs de navegadores — construções presentes no navegador, as quais são baseadas em linguagem JavaScript e permitem a implementação de funcionalidades de uma maneira mais fácil.
- APIs de terceiros — construções presentes em plataformas de terceiros (ex: Twitter, Facebook), que permitem o uso de alguma funcionalidade da plataforma em suas páginas na web. Um exemplo é a possibilidade de mostrar os últimos tweets em sua página.
- Bibliotecas JavaScript — em geral, um ou mais arquivos JavaScript contendo funções personalizadas, as quais podem ser usadas em sua página web para acelerar ou permitir escrever funcionalidades comuns. Exemplos: jQuery, Mootools e React.
- Frameworks JavaScript — uma evolução das bibliotecas. Frameworks JavaScript (ex: Angular e Ember), normalmente, são pacotes de tecnologias HTML, CSS, JavaScript e outras, que você instala e usa para escrever

uma aplicação web completa do zero. A principal diferença entre uma biblioteca e um framework é a inversão de controle ("Inversion of Control"). Quando um método de uma biblioteca é chamado, a pessoa desenvolvedora está no controle. Em um framework, o controle inverte-se: é o framework que chama o código da pessoa desenvolvedora.

O que as APIs podem fazer ?

Existem muitas APIs disponíveis, nos navegadores modernos, que permitem uma liberdade de ação na hora de codar.

APIs comuns de navegadores

As categorias mais comuns de APIs de navegadores que você irá utilizar (e que veremos em detalhes neste módulo), são:

- **APIs para manipular documentos** carregados no navegador. O exemplo mais óbvio é a API `DOM (Document Object Model)`, que permite manipular `HTML e CSS` — criando, removendo a alterando o HTML, aplicando dinamicamente novos estilos a sua página, etc. Toda vez que você vê uma janela pop-up em uma página, ou um novo conteúdo é mostrado, o `DOM` está em ação.
- `APIs que buscam dados no servidor` para atualizar pequenas seções da página, por conta própria, são bastante usadas. Isso, a princípio, parece ser um detalhe pequeno, mas tem um impacto enorme na performance e no comportamento dos sites. Se você precisa atualizar a cotação de uma ação ou listar novas histórias disponíveis, a possibilidade de fazer isso instantaneamente sem precisar atualizar a página dá a impressão de um site muito mais responsivo. Entre as APIs que tornam isso possível, podemos destacar o `XMLHttpRequest` e a `API Fetch`. Você pode também encontrar o termo Ajax, que descreve essa técnica.
- **APIs para desenhar e manipular elementos gráficos** são completamente suportados nos browsers(navegadores) — os mais comuns são `Canvas` e `WebGL`, que possibilitam que você atualize os dados dos pixels em um elemento `HTML` de maneira programática. `<canvas>` elemento para criar cenas 2d e 3d. Por exemplo, você poderia dezenhar formas como retangulos e circulos, importar uma imagem para o canvas, e aplicar um filtro para sepia ou grayscale usando o Canvas API, ou criar uma complexa cena 3d com brilho e texturas usando WebGL. Essas APIs são frequentemente combinar com APIs para criar loops de animações(como

`window.requestAnimationFrame()`) e outros para constantemente lançar cenas like como cartoons e jogos.

- **Audio and Video APIs** como `HTMLMediaElement (en-US)`, a `Web Audio API`, e `WebRTC` permiten a você fazer coisas realmente interessantes com multimedia como a criação personalizada controles de UI(user interface - interface usuario) para executar audio e video, exibindo faixas de texto como legendas e legendas ocultas junto com seus vídeos, capturando vídeo de sua câmera da web para ser manipulado por meio de uma tela (veja acima) ou exibido no computador de outra pessoa em uma webconferência,ou adicionar efeitos às trilhas de áudio (como ganho, distorção, panorâmica, etc.).

- **Device APIs - APIs de dispositivo** São basicamente APIs para manipulação e recuperação de dados de hardware de dispositivo moderno de uma forma que seja útil para aplicativos da web.Já falamos sobre a Geolocation API acessando o dispositivo dados de localização para que você possa marcar sua posição em um mapa. Outros exemplos incluem informar ao usuário que uma atualização útil está disponível em um aplicativo da web por meio de notificações do sistema(Veja em `Notifications API`)ou hardware de vibração(Veja em `Vibration API`).

- **Client-side storage APIs(APIs de armazenamento do lado do cliente)** estão se tornando muito mais difundidos em navegadores da web - a capacidade de armazenar dados no lado do cliente é muito útil se você quer criar um app que vai salvar seu estado entre carregamentos de página, e talvez até funcione quando o dispositivo estiver offline. Existem várias opções disponíveis, por exemplo, armazenamento simples de nome/valor com o `Web Storage API`, e armazenamento de dados tabulares mais complexos com o `IndexedDB API`.

APIs comuns de terceiros

APIs de terceiros são bastante variadas. Dentre as mais populares, que você eventualmente irá utilizar em algum momento, podermos destacar:

- A Twitter API, que permite coisas como mostrar seu últimos tweets no seu website.

- O Google Maps API permite que você faça todo tipo de coisa com mapas nas suas páginas web (curiosamente, também alimenta o Google Maps). Este é agora um conjunto completo de APIs, que lidam com uma ampla variedade de tarefas, conforme evidenciado pelo Seletor de API do Google Maps.

- O pacote de APIs do Facebook permite que você use várias partes do ecossistema do Facebook para beneficiar seu aplicativo, por exemplo, fornecendo login do aplicativo usando o login do Facebook, aceitando pagamentos no aplicativo, lançando campanhas publicitárias direcionadas, etc
- A API do YouTube, que permite incorporar vídeos do YouTube em seu site, pesquisar no YouTube, criar listas de reprodução e muito mais.
- A API Twilio, que fornece uma estrutura para a criação de funcionalidade de chamada de voz e vídeo em seu aplicativo, envio de SMS / MMS de seus aplicativos e muito mais.

Como as APIs funcionam ?

APIs JavaScript possuem pequenas diferenças mas, em geral, possuem funcionalidades em comum e operam de maneira semelhante.

Elas são baseadas em objetos

Seu código interage com APIs usando um ou mais objetos JavaScript , que servem como contêineres para os dados que a API usa (contidos nas propriedades do objeto) e a funcionalidade que a API disponibiliza (contida nos métodos de objeto).

Vamos voltar ao exemplo da API de geolocalização - esta é uma API muito simples que consiste em alguns objetos simples:

- Geolocation, que contém três métodos para controlar a recuperação de geodados.
- Position, que representa a posição de um dispositivo em um determinado momento - contém um Coordinates objeto que contém as informações de posição real, além de um carimbo de data/hora que representa o horário determinado.
- Coordinates, que contém muitos dados úteis sobre a posição do dispositivo, incluindo latitude e longitude, altitude, velocidade e direção do movimento e muito mais.

Então, como esses objetos interagem?

```
1
2     navigator.geolocation.getCurrentPosition(function(position) {
3       var latlng = new google.maps.LatLng(position.coords.latitude,
    position.coords.longitude);
4       var myOptions = {
5         zoom: 8,
6         center: latlng,
7         mapTypeId: google.maps.MapTypeId.TERRAIN,
8         disableDefaultUI: true
9       }
10      var map = new google.maps.Map(document.querySelector("#
    map_canvas"), myOptions);
11    });
```

Listing 1: exemplo de uso da API geolocation

Primeiro, queremos usar o `Geolocation.getCurrentPosition()` método para retornar a localização atual do nosso dispositivo. O `Geolocation` objeto do navegador é acessado chamando a `Navigator.geolocation` propriedade, então começamos usando:

```
1
2     navigator.geolocation.getCurrentPosition(function(position) {
    ... });
```

Isso é equivalente a fazer algo como:

```
1
2     var myGeo = navigator.geolocation;
3     myGeo.getCurrentPosition(function(position) { ... });
```

Mas podemos usar a sintaxe de ponto para encadear nosso acesso de propriedade/método, reduzindo o número de linhas que temos que escrever.

O `Geolocation.getCurrentPosition()` método possui apenas um único parâmetro obrigatório, que é uma função anônima que será executada quando a posição atual do dispositivo for recuperada com sucesso. Essa função em si tem um parâmetro, que contém um `Position` objeto que representa os dados da posição atual.

Esse padrão de invocar uma função apenas quando uma operação foi concluída é muito comum em APIs JavaScript - certificar-se de que uma operação foi concluída antes de tentar usar os dados que a operação retorna em outra operação. Elas são chamadas de **operações assíncronas**. Como a obtenção da posição atual do dispositivo depende de um componente externo (o GPS do dispositivo ou outro hardware de geolocalização), não podemos garantir que isso será feito a tempo de usar imediatamente os dados que retorna. Portanto, algo assim não funcionaria:

```
1
2     var position = navigator.geolocation.getCurrentPosition();
3     var myLatitude = position.coords.latitude;
```

Se a primeira linha ainda não tivesse retornado seu resultado, a segunda linha geraria um erro, porque os dados de posição ainda não estariam disponíveis. Por esse motivo, APIs envolvendo operações assíncronas são projetadas para usar funções de retorno de chamada , ou o sistema mais moderno de Promises , que foram disponibilizados no ECMAScript 6 e são amplamente usados em APIs mais recentes.

Estamos combinando a API de geolocalização com uma API de terceiros - a API do Google Maps - que usamos para traçar a localização retornada por `getCurrentPosition()` em um mapa do Google. Disponibilizamos essa API em nossa página criando um link para ela - você encontrará esta linha no HTML:

```

    <script type="text/javascript" src="https://maps.google.com/
    maps/api/js?key=AIzaSyDDuGt0E5IEGkcE6ZfrKfUtE9Ko_de66pA"></
    script>
```

Para usar a API, primeiro criamos uma `LatLng` instância de objeto usando o `google.maps.LatLng()` construtor, que usa nossos valores de geolocalização `Coordinates.latitude (en-US)` e `Coordinates.longitude (en-US)` como parâmetros:

```

    var latlng = new google.maps.LatLng(position.coords.latitude,
    position.coords.longitude);
```

Este próprio objeto é definido como o valor da `center` propriedade de um objeto de opções que chamamos `myOptions`. Em seguida, criamos uma instância de objeto para representar nosso mapa chamando o `google.maps.Map()` construtor, passando-lhe dois parâmetros - uma referência ao `<div>` elemento no qual queremos renderizar o mapa (com um ID de `map_canvas`) e o objeto de opções que definimos logo acima dele.

```

    var myOptions = {
      zoom: 8,
      center: latlng,
      mapTypeId: google.maps.MapTypeId.TERRAIN,
      disableDefaultUI: true
    }

    var map = new google.maps.Map(document.querySelector("#
    map_canvas"), myOptions);
```

Feito isso, nosso mapa agora é renderizado.

Este último bloco de código destaca dois padrões comuns que você verá em muitas APIs. Em primeiro lugar, os objetos API comumente contêm construtores, que são invocados para criar instâncias daqueles objetos que você

usará para escrever seu programa. Em segundo lugar, os objetos API geralmente têm várias opções disponíveis que podem ser ajustadas para obter o ambiente exato que você deseja para o seu programa. Os construtores de API geralmente aceitam objetos de opções como parâmetros, que é onde você define essas opções.

Possuem pontos de entrada reconhecíveis

Ao usar uma API, você deve certificar-se de saber onde está o ponto de entrada para a API. Na API de geolocalização, isso é muito simples - é a `Navigator.geolocation` propriedade, que retorna o `Geolocation` objeto do navegador onde todos os métodos de geolocalização úteis estão disponíveis.

A API Document Object Model (DOM) tem um ponto de entrada ainda mais simples - seus recursos tendem a ser encontrados pendurados no `Document` objeto ou uma instância de um elemento HTML que você deseja afetar de alguma forma, por exemplo:

```
1
2     var em = document.createElement('em'); // cria um novo elemento
3     var para = document.querySelector('p'); // referencia a um
      elemento HTML <p> existente
4     em.textContent = 'Hello there!'; // d   a eles algum conte do
      de texto
5     para.appendChild(em); //inseri o elemento <em> em paragrafo
```

Outras APIs têm pontos de entrada um pouco mais complexos, geralmente envolvendo a criação de um contexto específico para o código da API a ser escrito. Por exemplo, o objeto de contexto da API Canvas é criado obtendo uma referência para o `<canvas>` elemento que você deseja desenhar e, em seguida, chamando seu `HTMLCanvasElement.getContext()` método:

```
1
2     var canvas = document.querySelector('canvas');
3     var ctx = canvas.getContext('2d');
```

Qualquer coisa que quisermos fazer na tela é então alcançada chamando propriedades e métodos do objeto de conteúdo (que é uma instância de `CanvasRenderingContext2D`), por exemplo:

```
1
2     Ball.prototype.draw = function() {
3       ctx.beginPath();
4       ctx.fillStyle = this.color;
5       ctx.arc(this.x, this.y, this.size, 0, 2 * Math.PI);
6       ctx.fill();
7     };
```

Usam eventos para lidar com mudanças de estado

Já discutimos os eventos no início do curso, em nosso artigo de introdução aos eventos - este artigo analisa em detalhes o que são os eventos da Web do lado do cliente e como eles são usados em seu código. Se você ainda não está familiarizado com o funcionamento dos eventos de API da web do lado do cliente, leia este artigo antes de continuar.

Algumas APIs da web não contêm eventos, mas algumas contêm vários eventos. As propriedades do manipulador que nos permitem executar funções quando os eventos são disparados geralmente são listadas em nosso material de referência em seções separadas "Manipuladores de eventos". Como um exemplo simples, as instâncias do `XMLHttpRequest` objeto (cada uma representa uma solicitação HTTP ao servidor para recuperar um novo recurso de algum tipo) têm vários eventos disponíveis, por exemplo, o `load` evento é disparado quando uma resposta é retornada com sucesso contendo o recurso solicitado e agora está disponível.

O código seguinte fornece um exemplo simples de como isso seria utilizado:

```
1
2     var requestURL = 'https://mdn.github.io/learning-area/
      javascript/oojs/json/superheroes.json';
3     var request = new XMLHttpRequest();
4     request.open('GET', requestURL);
5     request.responseType = 'json';
6     request.send();
7
8     request.onload = function() {
9       var superHeroes = request.response;
10      populateHeader(superHeroes);
11      showHeroes(superHeroes);
12    }
```

As primeiras cinco linhas especificam a localização do recurso que queremos buscar, crie uma nova instância de um objeto de solicitação usando o `XMLHttpRequest()` construtor, abra uma `GET` solicitação HTTP para recuperar o recurso especificado, especifique que a resposta deve ser enviada no formato `JSON` e envie o solicitação.

A `onload` função de manipulador então especifica o que fazemos com a resposta. Sabemos que a resposta será retornada com sucesso e estará disponível depois que o evento de carregamento for solicitado (a menos que ocorra um erro), então salvamos a resposta contendo o JSON retornado na `superHeroes` variável e a passamos para duas funções diferentes para processamento posterior.