MÓDULO

JAVASCRIPT/AJAX

UNIDADE

INTEGRAÇÃO DE BIBLIOTECAS EXTERNAS EM JAVASCRIPT

Instituto Tecnológico

ÍNDICE

OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Saber integrar a biblioteca do Google Maps.
- Saber integrar a biblioteca de calendário do Google.
- Saber integrar a biblioteca gráfica dinâmica do Google.

INTRODUÇÃO

O mundo da programação web é um mundo em constante desenvolvimento e mudança, sobretudo no que respeita ao JavaScript, onde várias bibliotecas foram criadas para facilitar trabalho. Nas unidades anteriores, aprendeu a usar a biblioteca jQuery, mas esta não é o único exemplo deste tipo de bibliotecas gratuitas que fornecem uma série de ferramentas muito poderosas para o desenvolvimento web.

Esta unidade incidirá sobre algumas outras bibliotecas utilizadas no desenvolvimento das páginas web.

1. INTEGRAÇÃO DO GOOGLE MAPS

Uma das bibliotecas mais utilizadas na internet é a biblioteca Google Maps, que permite integrar os mapas do Google em websites e interagir com os mesmos. Para integrar a biblioteca do Google Maps deverá fazer uma chamada para a referida biblioteca, que está nos servidores do Google. A ligação é feita da seguinte forma:

```
<script type="text/JavaScript"
src="https://maps.googleapis.com/maps/api/js?key=YOUR_KEY&callback=
myMap">

</script>
```

A biblioteca requer uma key para a API gratuita, que pode obter no link abaixo, e que deverá ser inserida no código acima no lugar de "YOUR_KEY":

- https://developers.google.com/maps/documentation/javascript/

Colocará esta linha dentro da tag<head> do código, e esta será responsável por fazer a chamada para a biblioteca, para permitir o uso das funções.

O elemento map que se cria no Google Maps requer, além da chamada à biblioteca, que no código exista uma div com um id que usará para conter o mapa. O tamanho do mapa será o tamanho da div.

```
<divid=“mapa”
style=“position:relative;width:300px;height:140px”></div>
```

Para criar o mapa utilizam-se as funções da biblioteca. O objeto chave da biblioteca é o **google.maps.Map**, que é o que permite criar um novo mapa, e recebe dois parâmetros: o primeiro é o id da div do mapa na página web e o segundo são as opções do mapa. As opções do mapa mais comuns são:

- **zoom**: indica o zoom que o mapa terá (quanto maior, mais próximo estará o mapa).
- **center**: recebe as coordenadas (latitude e longitude) que serão o ponto do mapa onde a visualização está centrada. Para passar essas coordenadas, é utilizado outro objeto da biblioteca chamado **google.maps.LatLng**, com o qual é criado um objeto de coordenadas que recebe a latitude e longitude:

```
var ponto = newgoogle.maps.LatLng(latitude,longitude)
```

- **mapTypeId**: indica o tipo de mapa que pretende exibir. Para fazer isso, é utilizado outro objeto da biblioteca chamado **google.maps.MapTypeId**, e os tipos que podem ser selecionados são:
  - ROADMAP: exibe o mapa padrão.
  - SATELLITE: mostra o mapa com imagens de satélite do Google Earth.
  - HYBRID: mostra uma mistura dos dois anteriores.
  - TERRAIN: mostra o mapa físico do relevo.

```
google.maps.MapTypeId.ROADMAP
```

Exemplo 01

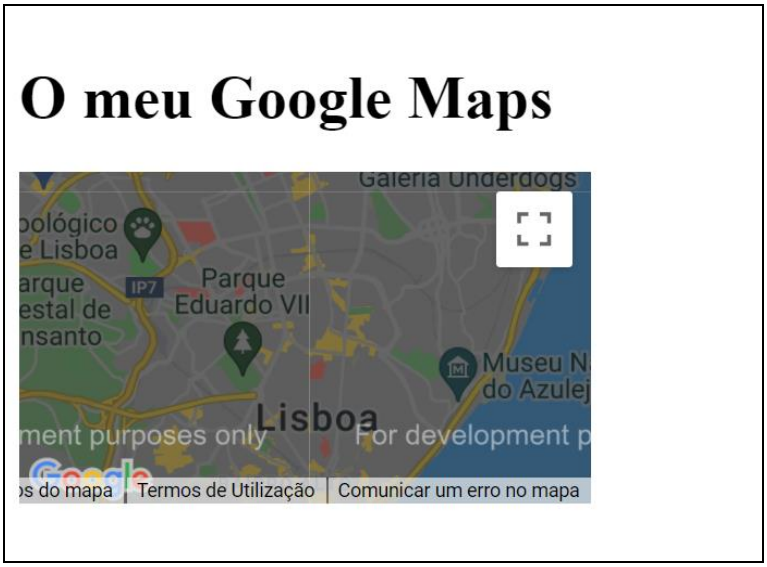
Este exemplo carrega o mapa do Google com a biblioteca numa div chamada "mapa".

Exemplo:

```
<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<script type="text/JavaScript"></script>

<script

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCJ4UYojA3Ldu
ldgOlizCQJjjpNCjHkvn4&callback=myMap"></script>

<script type="text/JavaScript">

/* criar a variavel ponto com definição de coordenadas */

function carregarmapa() {

var ponto = new google.maps.LatLng(38.733572717953415, -
9.141140002274987);

/* criar as opções do mapa */

var opcoes = {

zoom: 12,

center: ponto,

mapTypeId: google.maps.MapTypeId.ROADMAP};

/* criar o mapa */

var m = new google.maps.Map(document.getElementById("mapa"),
opcoes);

}

</script>

</head>

<body onload="carregarmapa()">
```

```
    <h1>O meu Google Maps</h1>
    <div id="mapa" style="width:300px; height:175px"></div>
</body>
</html>
```

A visualização deste exemplo será:



1.1. CRIAR MARCAS VISUAIS NUM GOOGLE MAPS

Uma das ferramentas mais úteis é a criação de pontos no mapa, ou seja, poder colocar uma marca nas coordenadas desejadas, para, por exemplo, marcar a localização de uma empresa.

Para fazer isso, utiliza-se um método da biblioteca chamado **google.maps.Marker** que permite criar esta mesma marca. Este método recebe vários parâmetros:

- **position**: o ponto de coordenada onde colocará a marca.
- **map**: o identificador do mapa onde pretende colocar a marca.

- **title**: texto que será exibido ao passar o cursor sobre a marca.
- **icon**: a imagem que pretende que seja mostrada como marca; se não colocar nada, aparecerá a marca padrão do Google.

Exemplo 02

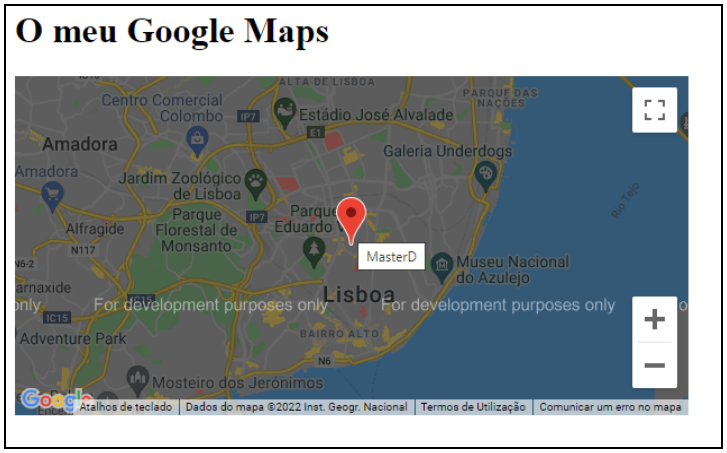
Este exemplo cria um mapa do Google e coloca uma marca nas coordenadas desejadas.

Exemplo:

```
<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<script type="text/JavaScript"></script>

<script

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCJ4UYojA3Ldu
ldgOlizCQJjjpNCjHkvn4&callback=myMap"></script>

<script type="text/JavaScript">

/* criar a variavel ponto com definição de coordenadas */

function carregarmapa() {

var ponto = new google.maps.LatLng(38.73376520878317, -
9.141150730691137);

/* criar as opções do mapa */

var opcoes = {

zoom: 12,

center: ponto,

mapTypeId: google.maps.MapTypeId.ROADMAP};

/* criar o mapa */
```

```
var m = new google.maps.Map(document.getElementById("mapa"),
opcoes);

/* criar a marca */


var marca = new google.maps.Marker({

position: ponto,

map: m,

title: "MasterD"

});

}

</script>

</head>

<body onload="carregarmapa()">

<h1>O meu Google Maps</h1>

<div id="mapa" style="width: 600px; height: 300px;"></div>

</body>

</html>
```

A visualização deste exemplo será:



Agora que aprendeu a criar uma marca dentro de um mapa, falta ver como fazer uma caixa com o texto que desejar ao clicar na marca. Para isso, é necessário inserir um evento na referida marca e utilizar o método **google.maps.event.addListener**, que recebe o identificador da marca, o evento no qual a ação é realizada (por exemplo, click), e a função a ser executada.

Além disso, é necessário criar uma caixa de conteúdo para a marca; para isso será utilizado o objeto **google.maps.InfoWindow**, que cria uma caixa dentro do mapa do Google. E com o método **open** deste objeto mostrará a referida caixa.

Exemplo 03

Neste exemplo pretende-se criar uma marca no mapa do Google com um evento que, ao clicar, mostre os dados da empresa.

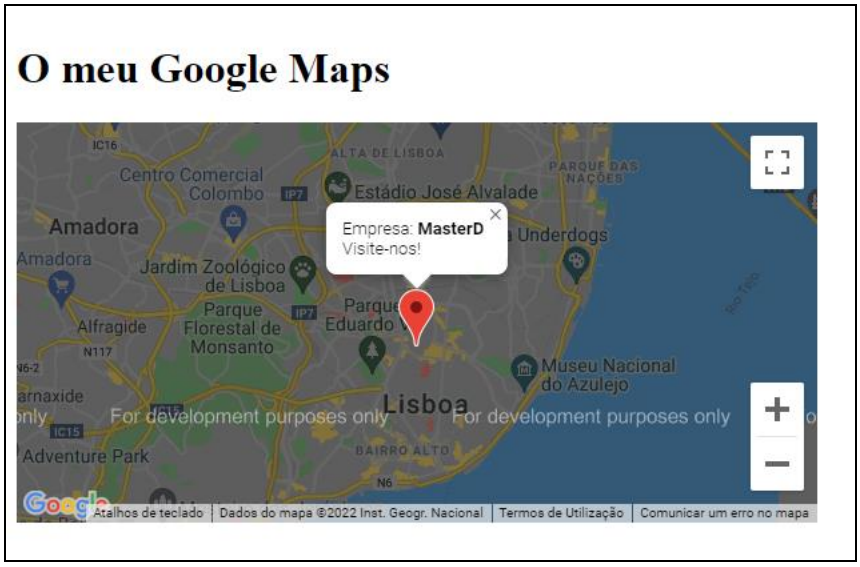
Exemplo:

```
<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<script type="text/JavaScript"></script>

<script

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCJ4UYojA3Ldu
ldgOlizCQJjjpNCjHkvn4&callback=myMap"></script>

<script type="text/JavaScript">

/* criar a variavel ponto com definição de coordenadas */

function carregarmapa() {

var ponto = new google.maps.LatLng(38.73376520878317, -
9.141150730691137);

/* criar as opções do mapa */

var opcoes = {
```

```
zoom: 12,

center: ponto,

mapTypeId: google.maps.MapTypeId.ROADMAP};

/* criar o mapa */

var m = new google.maps.Map(document.getElementById("mapa"),
opcoes);

/* criar a marca */

var marca = new google.maps.Marker({

position: ponto,

map: m,

});

var caixa = new google.maps.InfoWindow({

content: 'Empresa: <b>MasterD</b><br/>Visite-nos!'});

google.maps.event.addListener(

marca, 'click', function()

{

caixa.open(m,marca);

});

</script>

</head>

<body onload="carregarmapa()">

<h1>O meu Google Maps</h1>

<div id="mapa" style="width: 600px; height: 300px;"></div>

</body>

</html>
```

Como pode ver no exemplo, o texto passado para a caixa pode ser HTML e, como tal, poderia inserir tabelas, links, imagens, entre outros.

A visualização deste exemplo será:



1.2. GEOLOCATOR

Outra ferramenta amplamente utilizada na biblioteca do Google Maps é a geo-localização, que permite utilizar vários métodos de biblioteca para posicionar o mapa no local a partir da morada inserida.

Para isso é utilizado um objeto de biblioteca chamado **google.maps.Geocoder** que permite passar um parâmetro com a morada e outro parâmetro com uma função que é executada quando a pesquisa é finalizada. Esta pesquisa é iniciada com o método **geocode** do objeto.

Esta pesquisa devolverá um estado OK e um array de resultados com a localização, ou um estado de erros, indicando que nenhuma morada foi encontrada.

Exemplo 04

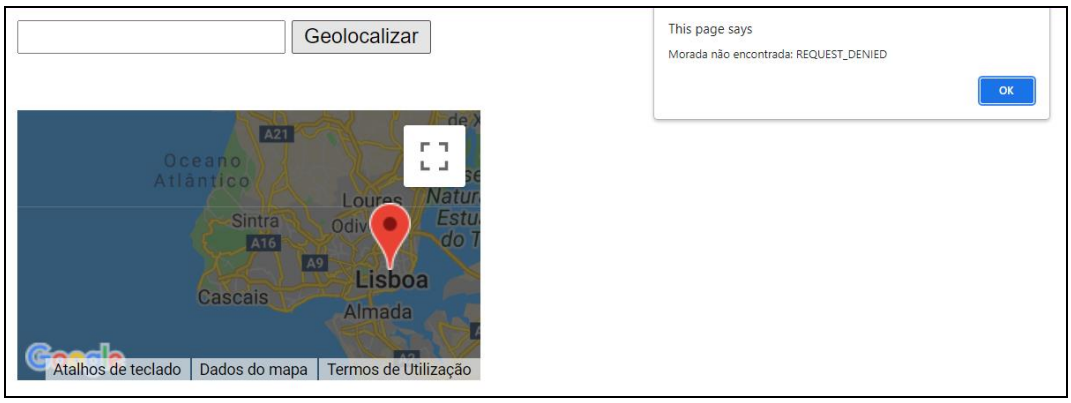
Neste exemplo, além do mapa, são criados um campo de texto e um botão para pesquisar uma morada.

Exemplo:

```
<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

    <script type="text/JavaScript"


src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCJ4UYojA3Ldu
ldgOlizCQJjjpNCjHkvn4&callback=myMap"></script>

    <script type="text/JavaScript">

    function carregarmapa() {   // criar o ponto das coordenadas

        var ponto = new google.maps.LatLng(

            38.73376520878317, -9.141150730691137

    );

        var opcoes = {   // criar as opções do mapa

            zoom: 12,

            center: ponto,

            mapTypeId: google.maps.MapTypeId.ROADMAP

        };

        var m = new
google.maps.Map(document.getElementById("mapa"), opcoes);   //
criar o mapa

        var marca = new google.maps.Marker({

            position: ponto,

            map: m

        });
```

```
        }
        function geo() {
            var geocoder = new google.maps.Geocoder();
            var direccao = $('#direccao').val();
            geocoder.geocode({'address': direccao},
            function(results, status) {
            if (status == 'OK') {
                m.setCenter(results[0].geometry.location);
                var marker = newgoogle.maps.Marker({
                map: m,
                position: results[0].geometry.location
                });
             } else {
                alert('Morada não encontrada: ' + status);
             }
            });
    }
    </script>
</head>
<body onload="carregarmapa()">
    <input id="direccao" type="textbox" value="Lisboa">
    <input type="button" value="Geolocalizar" onclick="geo()">
    <br /><br /><br />
    <div id="mapa" style="width:300px; height:175px"></div>
</body>
</html>
```

A visualização deste exemplo será:



1.3. CÁLCULO DE ROTA

Esta ferramenta permite calcular percursos entre dois pontos, muito úteis, por exemplo, para colocar nas páginas de contacto ou localização de uma empresa, permitindo ao utilizador não só saber a localização da empresa, mas também como lá chegar.

Para isso, são utilizados dois objetos da biblioteca: um chamado **DirectionsService**, que permite calcular essas rotas, e outro chamado **DirectionsRender**, que cria a rota pesquisada em modo de direções de texto. Vão ser utilizados quatro métodos dos objetos:

- **directionRenderer.setMap**: para indicar em que mapa vai calcular a rota.
- **directionRenderer.setPanel**: para indicar o id da div do HTML que incluirá a rota escrita para ir de um local a outro.
- **directionService.route**: o método que é chamado para encontrar a rota.
- **directionRenderer.setDirections**: o método que cria as direções da rota em modo texto a partir do resultado da route.

Exemplo 05

Neste exemplo, além do mapa, vão ser inseridos dois campos de texto e um botão; ao pressionar o botão irá pesquisar o caminho mais curto entre as duas moradas indicadas nos campos de texto.

Exemplo:

```
<!DOCTYPE   html>

<html>

<head>

<meta charset="UTF-8">

<script   type="text/JavaScript"

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCJ4UYojA3Ldu
ldgOlizCQJjjpNCjHkvn4&callback=myMap"></script>

<script type="text/JavaScript">

   var  mapa;

   var mostrarDireccao;

   var servicioRota=new google.maps.DirectionsService();

      function carregarmapa() {

         mostrarDireccao=new google.maps.DirectionsRenderer();

      var ponto = new google.maps.LatLng(

         38.733572717953415, -9.141140002274987);

      var opcoes = {

         zoom: 7,

         center: ponto,

         mapTypeId: google.maps.MapTypeId.ROADMAP};

         mapa = new
google.maps.Map(document.getElementById("mapa"),opcoes);
```

```
          mostrarDireccao.setMap(mapa);

          mostrarDireccao.setPanel(document.getElementById("rota"));

       }

    function calcularRota() {

          var partida = document.getElementById("partida").value;

          var destino = document.getElementById("destino").value;

          var opcoes = {

             origin:partida,

             destination:destino,

             travelMode:   google.maps.DirectionsTravelMode.DRIVING
//indica, neste caso, que a viagem será de carro/mota};

          servicioRota.route(opcoes, function(response, status) {

             if (status == google.maps.DirectionsStatus.OK) {

             mostrarDireccao.setDirections(response);}

    });

}

</script>

</head>

<body onload="carregarmapa()">

Partida: <input id="partida" type="textbox" value=""> Destino:
<input id="destino" type="textbox" value="">

<input type="button" value="Geolocalizar" onclick="calcularRota()">

<br/><br/><br/>

<div id="mapa"
style="position:relative;float:left;width:650px;height:575px"></div
>

<div id="rota"
style="position:relative;float:left;width:250px;"></div>

</body>
```

```
</html>
```

1.4. GOOGLE MAPS – STREET VIEW

Este serviço permite, para além de apresentar um mapa Google, apresentar as fotografias tiradas pela Google ao nível das ruas. Para isso, é utilizado o objeto **StreetViewPanorama** da biblioteca, que permite associar as fotografias das ruas, uma div HTML através do método **setStreetView** do objeto map, e mostrar essas imagens.

Exemplo 06

Este exemplo mostra as fotografias das ruas por onde passa.

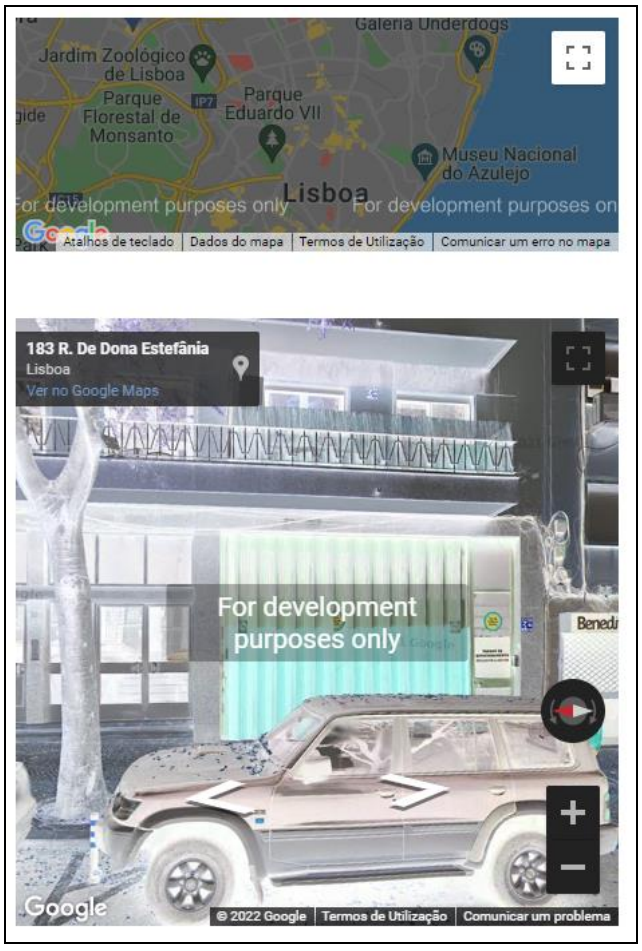
Exemplo:

```
<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

    <script type="text/JavaScript"


src="https://maps.googleapis.com/maps/api/js?key=AIzaSyCJ4UYojA3Ldu
ldgOlizCQJjjpNCjHkvn4&callback=myMap"></script>

    <script type="text/JavaScript">

    function carregarmapa() {

        var ponto = new google.maps.LatLng(

            38.73376520878317, -9.141150730691137
```

```
    );

        var opcoes = {

            zoom: 12,

            center: ponto,

            mapTypeId: google.maps.MapTypeId.ROADMAP

        };

        mapa = new google.maps.Map(document.getElementById("mapa"),
opcoes);

        var opcoesFotos = {

            position: ponto,

            pov: {

                heading: 90, // ângulo de rotação em relação ao
norte da posição

                pitch: 0, // variação do ângulo para cima ou para
baixo da câmara

                zoom: 2 // zoom de close-up da foto, sendo 0 sem
zoom

            }

        };

        var fotos = new
google.maps.StreetViewPanorama(document.getElementById("fotos") ,
opcoesFotos);

        mapa.setStreetView(fotos);

     }

    </script>

</head>

<body onload="carregarmapa()">

    <div id="mapa" style="width:450px; height:175px;margin-
bottom:50px;"></div>
```

```
    <div id="fotos"
style="position:relative;float:left;width:450px;height:455px;
"></div>

</body>

</html>
```

A visualização deste exemplo será:



2. GOOGLE CALENDAR

Outra biblioteca bastante utilizada é o calendário Google. De uma forma muito simples, permite incorporar calendários totalmente configuráveis às páginas web; é capaz de criar sistemas de agenda, alertas de eventos ou até mesmo um portefólio de clientes.

Para integrar o calendário Google será utilizada a biblioteca FullCalendar, que é gratuita e torna o código mais simples e leve. Pode ser descarregada em https://fullcalendar.io/.

Para a utilizar é necessário carregar um ficheiro CSS e dois ficheiros JS da biblioteca, e um ficheiro JS da biblioteca moment.js, que é uma biblioteca JavaScript para controlar as datas e facilitar o seu manuseio (mais informações em https://momentjs.com/).

Uma vez que as bibliotecas estiverem carregadas, é necessário criar uma div para conter o calendário. Esta div pode ter os estilos CSS que desejar (e para que possa ajustá-la ao tamanho que pretender).

Por fim, utilizando o jQuery (ready), inicia-se a criação do calendário assim que a página inteira for carregada, para que seja utilizado o método **fullCalendar**.

Este método permite criar o calendário, podendo passar parâmetros de configuração e eventos do calendário. Os parâmetros mais utilizados são:

- **header**: permite definir os elementos principais do calendário, por exemplo, os botões de navegação, o título do calendário e as opções de exibição. Os valores possíveis são:
  - title: texto que contém o mês/semana/dia atual.
  - prev: botão para retroceder o calendário um mês/semana/dia.
  - next: botão para avançar o calendário um mês/semana/dia.
  - prevYear: botão para mover o calendário para o ano anterior.
  - nextYear: botão para avançar o calendário um ano.
  - today: para mover o calendário para o mês/semana/dia atual.
  - Vistas possíveis: permite definir a visualização que o utilizador coloca no calendário.
    - month – mês.
    - basicWeek – semana.
    - basicDay – dia.
    - agendaWeek – semana na agenda (horas).
    - agendaDay – dia da agenda (horas).
    - listYear – lista de eventos sem calendário (visualização anual).
    - listMonth – lista de eventos sem calendário (visualização mensal).
    - listWeek – lista de eventos sem calendário (visualização semanal).
    - listDay – lista de eventos sem calendário (visualização diária).
- **defaultDate**: permite indicar em que dia inicial começa a visualização do calendário. Se não indicar nenhum, utiliza a biblioteca de momentos para estabelecer o dia atual.
- **navLinks**: permite definir se os nomes dos dias serão selecionáveis, ou seja, se for possível clicar neles, por defeito, abrirá no dia selecionado. Por defeito, este valor é "false". Ao definir como "true", esta funcionalidade passa a ser possível.
- **editable**: permite determinar se os eventos do calendário podem ser modificados, ou seja, se os eventos podem ser arrastados e redimensionados. Por defeito, o valor é "false".

- **eventLimit**: limita o número de eventos exibidos num dia. Quando há mais eventos do que o número definido, um link semelhante a "+X mais" é exibido. Um valor “false” (padrão) exibirá todos os eventos como estão. Um valor “true” limitará o número de eventos à altura da célula do dia.
- **events**: permite definir os eventos que irão aparecer no calendário. Existem duas formas de definir: através de um array que é passado como parâmetro, ou como dado retido de um ficheiro JSON.

  Por array:

```
events: [
{
title : 'evento1',
start : '2021-01-01'
 },
 {
title : 'evento2',
start : '2021-01-05',
end : '2021-01-07'
 }
```

  Por URL:

```
events: '/eventosJson.php'
```

  Normalmente, são criados por URL a partir de um banco de dados com PHP ou Java.

- **defaultView**: permite definir a vista inicial que o calendário terá. Por defeito, se o deixar vazio, será a vista mensal.

Exemplo 07

Neste exemplo, aprenderá a carregar as bibliotecas necessárias (serão utilizados os links das bibliotecas e não os ficheiros locais) e um calendário básico com alguns eventos. Este exemplo precisa de um servidor web para funcionar corretamente.

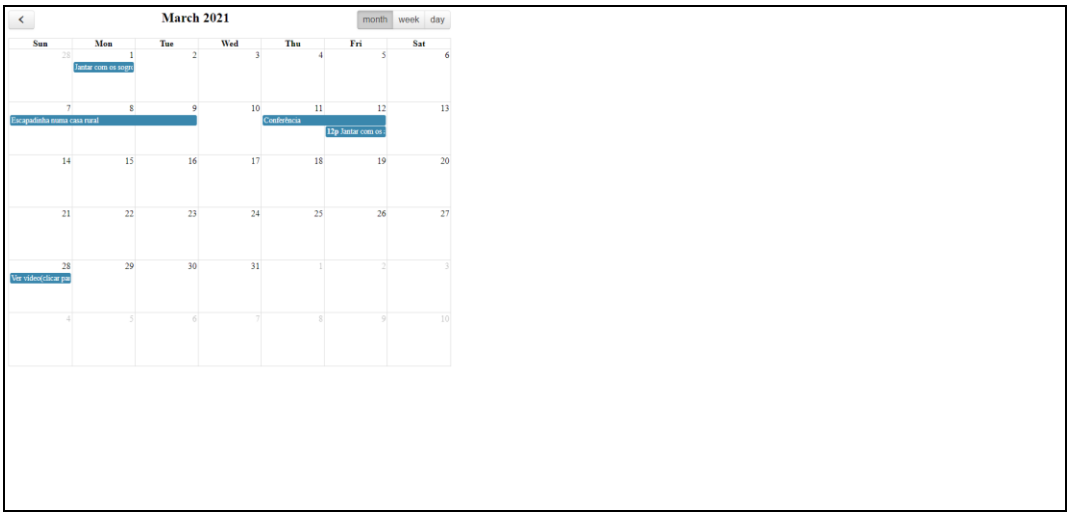
Exemplo:

```
<html>

<head>

<title>Exemplo Google Calendar</title>

<meta charset="UTF-8">

<link rel='stylesheet' type='text/css'
href='https://cdnjs.cloudflare.com/ajax/libs/fullcalendar/2.7.1/ful
lcalendar.min.css'>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

<script
src='https://momentjs.com/downloads/moment.min.js'></script>

<script src='http://fullcalendar.io/js/fullcalendar-
2.1.1/fullcalendar.min.js'></script>

</head>

<body >

<div style="width:800px;height:600px; position:relative;"
id="calendar">

</div>

<script>

    $(document).ready(function() {

          $('#calendar').fullCalendar({

                header: {
```

```
                        left: 'prev,nexttoday',
                        center: 'title',
                        right: 'month,basicWeek,basicDay'
                },
                defaultDate: '2021-03-12',
                navLinks: true,
                editable: true,
                eventLimit: true,
                events: [
                        {
                                title: 'Jantar com os sogros',
                                start: '2021-03-01'
                        },
                        {
                                title: 'Escapadinha numa casa rural',
                                start: '2021-03-07',
                                end: '2021-03-10'
                        },
                        {
                                title: 'Conferência',
                                start: '2021-03-11',
                                end: '2021-03-13'
                        },
                        {
                                title: 'Jantar com os amigos',
                                start: '2021-03-12T12:00:00'
```

```
                        },
                        {
                                title: 'Ver vídeo(clicar para ver)',
                                url: 'https://www.youtube.com/',
                                start: '2021-03-28'
                        }
                ]
        });
    });
</script>
</body>
</html>
```

A visualização deste exemplo será:



O verdadeiro poder desta biblioteca é que, juntamente com uma linguagem de programação, é possível utilizá-la como controlo de nomeação, modificação e criação de eventos. Para isso, conta-se com o sistema de eventos da própria biblioteca, que permite criar os eventos do rato que ocorrem no calendário. Os eventos mais usados são:

- **eventRender**: este evento é disparado quando um evento é processado; geralmente, é utilizado para modificar algum estilo por CSS ou lançar um alerta.
- **dayClick**: é executado quando o utilizador clica num dia; normalmente, é usado para exibir uma janela para que um formulário possa ser preenchido com os dados necessários para criar um compromisso ou entrada no calendário. A função que ele gera dá acesso ao valor do dia, ao evento e ao tipo de visualização que é ativada, o que nos permite realizar diferentes ações, dependendo de se for selecionado numa visualização ou noutra (por exemplo, criar um compromisso ao clicar no formato de semana, mas mostrar apenas a mensagem na visualização mensal).

```
dayClick: function(date, jsEvent, view) {

    alert('dia clicado: ' + date.format());

    alert('Coordenadas: ' + jsEvent.pageX + ',' + jsEvent.pageY);

    alert('Vista atual: ' + view.name);

}
```

- **eventClick**: iniciado quando um evento do calendário é selecionado; geralmente, é usado para visualizar os detalhes do evento ou exibir um formulário para modificar o evento.

  A função criada devolve tudo o que for passado e definido no parâmetro como array ou json, funcionando como array, e pode devolver, por exemplo, o elemento jsEvent, a visão atual e os dados do evento, título, datas, etc.

```
eventClick: function(calEvent, jsEvent, view) {

    alert('Titulo do evento: ' + calEvent.title);

    alert('Coordenadas: ' + jsEvent.pageX + ',' + jsEvent.pageY);

    alert('Vista: ' + view.name);

}
```

- **eventDrop**, lançado quando um evento do calendário é arrastado para um dia diferente do indicado; a função devolve um objeto de evento, outro objeto delta, uma função para devolver o evento ao ponto inicial revertFunc, o objeto jsEvent e a visualização:

```
eventDrop: function(event, delta, revertFunc) {

    alert(event.title + " movido para " + event.start.format());

    if (!confirm("De certeza que quer mover?")) {

           revertFunc();

    }

}
```

3. GOOGLE CHARTS

Esta biblioteca gratuita do Google permite criar gráficos a partir de dados dos websites para mostrar os dados visualmente. Para os utilizar é necessário carregar a API a partir do seguinte URL: https://www.gstatic.com/charts/loader.js

```
<script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
```

Para fazer a chamada de carregamento, primeiro tem de carregar o pacote gráfico. Para isso usará a seguinte instrução:

- **google.charts.load (versão, pacotes, idioma)**: o primeiro parâmetro permite definir a versão da API gráfica a ser carregada. Se indicar "current" como valor, será a versão atual do Google naquele momento.

  No segundo parâmetro é indicado o pacote gráfico que pretende utilizar (podendo passar mais do que um), por padrão será passado **corechart** como um valor que permite criar todos os gráficos básicos do Google (barras, colunas, linhas, circulares, de bolhas, entre outros).

  O terceiro parâmetro permite definir o idioma no qual o gráfico é exibido; por defeito está em inglês. Para o colocar em português basta indicar language como "pt", deixando a chamada completa assim:

  - google.charts.load('current', {'packages':['corechart'], 'language': 'pt'});

A instrução seguinte a utilizar sempre é:

- **google.charts.setOnLoadCallback(função)**: esta instrução é muito semelhante ao ready do Ajax, ou seja, chama a função JavaScript criada para carregar os gráficos assim que a página web for carregada.

  Os dados do gráfico são criados usando um **DataTable**, que é uma classe JavaScript que simula tabelas de dados HTML divididas em linhas e colunas.

  A maneira padrão de preencher a tabela é indicando colunas com a instrução addColumn e linhas com addRows.

  Para criar uma tabela que relaciona as profissões ao número de horas de trabalho semanais como esta:

Profissão	N.º de horas
Cozinheiro	56
Professor	42
Programador	48

  A forma padrão de criar esta tabela utilizando o **DataTable** seria:

```
var data = newgoogle.visualization.DataTable ();

data.addColumn ('string', 'Profissão');

data.addColumn ('number', 'Nº de Horas');

data.addRows ([

['Cozinheiro', 56],

 ['Professor', 42],

 ['Programador', 48]

]);
```

Estas tabelas de dados são normalmente retiradas de bancos de dados e utilizam outras linguagens de programação como Java, PHP, C#, entre outras. É por isso que esta biblioteca tem um sistema para criar o objeto DataTable diretamente de um array, cuja instrução é:

- **arrayToDataTable**

  Para criar a mesma tabela do exemplo anterior, mas com esta instrução, seria:

```
var data = google.visualization.arrayToDataTable ([

 ['Profissão', 'Nº de Horas'],

 ['Cozinheiro', 56],

 ['Professor', 42],

 ['Programador', 48]

],

 false); // 'false' significa que a primeira linha contém rótulos,
não dados.
```

O passo seguinte seria personalizar o gráfico. Para o fazer, é possível definir um título, as dimensões, cores, etc. Dependendo do gráfico, terá uma ou outra opção, e poderá ver todas as opções disponíveis em https://developers.google.com/chart/interactive/docs/.

Neste exemplo, serão colocados um título e as dimensões do gráfico:

```
var options = {‘title’: 'Horário de trabalho', 'width': 400,
'height': 300};
```

Por fim, para desenhar o gráfico, é necessário criar uma div e fazer a chamada para a classe do gráfico que pretende criar; cada gráfico tem a sua própria classe, por exemplo, o gráfico circular é PieChart, o gráfico de barras é BarChart, etc.

No URL https://developers.google.com/chart/interactive/docs/gallery poderá encontrar todos os tipos de gráficos..

Para desenhar o gráfico, primeiro é necessário criar a classe e associá-la à div (neste exemplo, será um gráfico circular).

```
var chart = newgoogle.visualization.PieChart($('chart'));
```

E para terminar, passar à classe a ordem de desenhar o gráfico com os dados e as opções criados:

```
chart.draw (dados, opções);
```

Exemplo 08

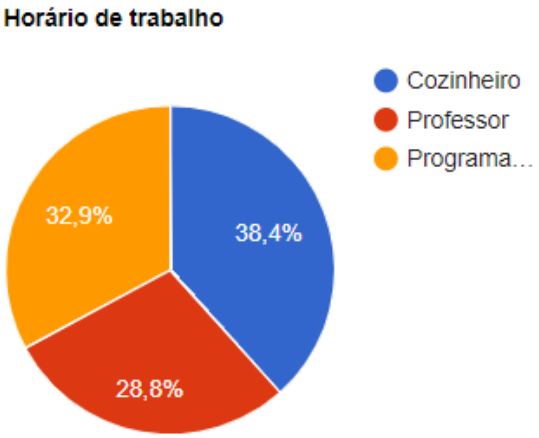
Neste exemplo reúne-se tudo o que foi referido acerca da criação de gráficos.

Exemplo:

```
<html>

<head>

<meta charset="UTF-8">

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.
js"></script>

<script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>

<script type="text/javascript">

google.charts.load('current', {'packages':['corechart'], 'language'
: 'pt'});

google.charts.setOnLoadCallback(desenharGrafico);

function desenharGrafico() {

 var data = newgoogle.visualization.DataTable();

data.addColumn ('string', 'Profissão');

    data.addColumn ('number', 'Nº de horas');

    data.addRows ([

            ['Cozinheiro', 56],

            ['Professor', 42],

            ['Programador', 48]
```

```
     ]);

 var opcoes = {'title':'Horário de
trabalho','width':400,'height':300};

 var chart =
newgoogle.visualization.PieChart(document.getElementById('caixa'));

chart.draw(data, opcoes);

 }

</script>

</head>

<body>

<div id="caixa" style="width:400; height:300"></div>

</body>

</html>
```

O resultado deverá ser o seguinte:



CONCLUSÃO

As bibliotecas externas fornecem várias melhorias que pode incorporar no website de forma rápida e fácil. Graças a elas, obtêm-se efeitos ou conteúdos que, de outra forma, demorariam muito tempo a preparar, como mapas ou conteúdos interativos e dinâmicos.

Bibliotecas de mapas, gráficos, calendários, etc. são bibliotecas gratuitas que pode utilizar para criar uma página web completa e profissional.

Existem também inúmeras bibliotecas no GitHub que pode importar nos projetos para realizar mil e uma funções, como a integração de controlos deslizantes de fotografias (“carrosséis”), efeitos de animações dinâmicas ou interativas, etc.

AUTOAVALIAÇÃO

1. **Onde é colocada a chamada para a API do Google Maps?**

   a) No <head>.

   b) No final da página.

   c) Não é necessário colocá-lo pois já vem incluído no Chrome.

   d) Em qualquer lado no código, desde que dentro da tag<script>.

2. **Qual é opção da biblioteca do Google Maps que permite indicar o tipo de mapa a ser utilizado?**

   a) mapType.

   b) typeOfMap.

   c) mapValue.

   d) mapTypeId.

3. **Qual é o objeto da biblioteca do Google Maps que permite criar um ponto de coordenadas para utilizar no mapa?**

   a) google.maps.coordenate.

   b) google.maps.point.

   c) google.maps.LatLng.

   d) google.maps.pointLaLn.

4. O que permite realizar o google.maps.Marker?

a) Criar uma marca ou ponto no mapa.

b) Encontrar as marcações no mapa.

c) Selecionar o mapa com o qual pretende trabalhar.

d) Esse método não existe.

5. Qual é o método da biblioteca do Google Maps que permite localizar geograficamente uma morada a partir de uma string?

a) google.maps.geo.

b) google.maps.geocoder.

c) google.maps.localice.

d) google.maps.locate.

6. Qual é o método da biblioteca do Google Maps que permite criar direções de cálculo de rota em modo de texto?

a) directionsService.route.

b) directionsService.setText.

c) directionsRenderer.route.

d) directionsRenderer.setDirections.

7. Qual é o evento do calendário do Google que permite detetar quando o botão é pressionado num dia?

a) clickCalendar.

b) onDayPress.

c) dayClick.

d) onDayClick.

8. **Quais são os parâmetros necessários para passar a função google.charts.load?**

   a) Nenhuns, não necessita de parâmetros.

   b) O tipo de gráfico a ser carregado.

   c) O tamanho e o nome do gráfico a ser carregado.

   d) A versão, os pacotes e o idioma.

9. **Que instrução é que permite converter um determinado array numa dataTable da API do Google Chart?**

   a) convertArray.

   b) arrayToTable.

   c) tableFromArray.

   d) arrayToDataTable.

10. **Qual é a instrução final necessária para carregar o gráfico na página web?**

    a) grafico.Print.

    b) grafico.draw.

    c) grafico.show.

    d) grafico.photo.

SOLUÇÕES

1.	a	**2.**	d	**3.**	c	**4.**	a	**5.**	b
6.	d	**7.**	c	**8.**	d	**9.**	d	**10.**	b

PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

Visite as seguintes páginas da web:

- Para saber mais sobre a biblioteca do Google Maps: https://developers.google.com/maps/
- Para saber mais sobre a biblioteca Google Charts: https://developers.google.com/chart
- Para saber mais sobre a biblioteca do Google Calendar: https://developers.google.com/google-apps/calendar
- Para conhecer as bibliotecas gratuitas do Google: https://developers.google.com/

BIBLIOGRAFIA

- Google Developers (2021). "Plataforma Google Maps" Disponível em:

  https://developers.google.com/maps/. Consultado a 18 de fevereiro de 2021.

- Google Developers (2021). "Google Calendar for Developers". Disponível em:

  https://developers.google.com/google-apps/calendar/. Consultado a 18 de fevereiro de 2021.

- Google Developers (2021). "Google Charts". Disponível em:

  https://developers.google.com/chart/. Consultado a 18 de fevereiro de 2021.