Tecnologias web utilizadas para visualização de dados

Neste documento, você irá aprender a utilizar tecnologias web para visualização de dados. Inicialmente, você irá utilizar os fundamentos das tecnologias usadas para desenvolvimento de aplicações web: HTML, CSS e JavaScript. Você irá aprender a manipular arquivos em formatos JSON e CSV, como objetos JavaScript. Você irá ainda conhecer algumas das bibliotecas mais utilizadas para construção de visualizações de dados, como Chart.js e D3.js.

Sumário

Requisitos básicos ***2***

Fundamentos das tecnologias web ***3***

HTML **3**

CSS **4**

JavaScript **6**

Inspecionar elemento **6**

Chart.js ***8***

Criando a base de uma aplicação **8**

Exemplos práticos **9**

Gráfico de barras 9

Gráfico de linhas 15

Gráfico de pizza 18

Gráfico de donut 20

Gráfico de dispersão 23

Obtendo dados de arquivos **28**

Requisitos básicos

Para desenvolver uma aplicação web para visualização de dados, você precisará de alguns requisitos básicos, como:

- Um navegador de sua preferência, como o Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, etc. Neste documento, iremos utilizar o Chrome.
- Uma IDE (*Integrated Development Environment*) de desenvolvimento, como o Visual Studio Code, Sublime Text, Atom, WebStorm, etc. Essa IDE permitirá que você escreva e edite seu código, depure e execute sua aplicação. Neste documento, iremos utilizar o Visual Studio Code (https://code.visualstudio.com/).
- Um servidor web para executar sua aplicação (opcional). O servidor pode ser PHP, Python, Node.js ou outra tecnologia de servidor da sua escolha. O servidor é responsável por processar solicitações e fornecer respostas para o navegador. Alguns exemplos de servidores web populares incluem Apache, Nginx, IIS, Node.js HTTP Server, etc. Precisaremos do servidor apenas para carregar arquivos locais.

Fundamentos das tecnologias web

Esta seção apresenta uma breve revisão sobre as principais tecnologias web usadas neste documento. Sinta-se a vontade para saltar este capítulo caso já tenha familiaridade com os temas.

HTML

HTML é a sigla para *Hypertext Markup Language*, que é uma linguagem de marcação utilizada para criar páginas web e aplicações web. O HTML é uma das principais tecnologias usadas na web, juntamente com o CSS e o JavaScript.

A linguagem HTML utiliza tags para definir a estrutura e o conteúdo da página web. Cada tag é definida entre os caracteres "<" e ">". O conteúdo da tag é definido entre as tags de abertura e fechamento. Por exemplo, a tag " " é usada para definir um parágrafo e deve ser fechada com a tag " ".

Aqui está um exemplo básico de código HTML que define um parágrafo:

```
<!DOCTYPE html>
<html>
 <head>
  <title>Exemplo de página HTML</title>

  <!-- incluindo uma folha de estilo -->
  <link rel="stylesheet" type="text/css" href="estilo.css">
 </head>
 <body>
  <p>Este é um exemplo de parágrafo.</p>

  <!-- incluindo um arquivo JavaScript -->
  <script src="script.js"></script>

 </body>
</html>
```

Este código define uma página web básica que contém um único parágrafo. A primeira linha define o tipo de documento HTML que está sendo usado. As tags "<html>" e "<head>" definem o cabeçalho da página e a tag "<title>" define o título da página. A tag "<body>" define o corpo da página, que é onde o conteúdo real da página é definido. O parágrafo é definido usando a tag "<p>".

Para carregar um arquivo CSS e um script em um arquivo HTML, você pode usar as tags "link" e "script", respectivamente. Essas tags permitem que você especifique a localização do arquivo que deseja carregar.

Para carregar um arquivo CSS, você pode adicionar a seguinte tag "link" dentro da seção "head" do seu documento HTML. Nesse exemplo, o arquivo CSS está localizado em "estilo.css" na mesma pasta do arquivo HTML. O atributo "rel" especifica o tipo de link, que é um estilo CSS, e o atributo "href" especifica a localização do arquivo.

Para carregar um script em seu arquivo HTML, você pode adicionar a seguinte tag "script" dentro da seção "head" ou "body" do seu documento HTML. Nesse exemplo, o arquivo de script está localizado em "script.js" na mesma pasta do arquivo HTML. O atributo "src" especifica a localização do arquivo de script.

É importante lembrar que a ordem em que você carrega seus arquivos é importante. Normalmente, é recomendável carregar arquivos CSS antes de arquivos de script, para garantir que o estilo seja aplicado corretamente.

CSS

CSS é a sigla para *Cascading Style Sheets*, que é uma linguagem de estilo usada para controlar a apresentação e o layout de páginas web e aplicações web. O CSS permite separar o conteúdo e a estrutura de uma página HTML da sua apresentação visual.

A linguagem CSS utiliza seletores para aplicar estilos a elementos HTML específicos. Por exemplo, você pode usar um seletor de classe para aplicar um estilo a todos os elementos com uma determinada classe.

Aqui está um exemplo básico de código CSS que define um estilo para um parágrafo:

```
p {
  color: blue;
```

```
  font-size: 16px;
}
```

Nesse exemplo, o seletor "p" é usado para selecionar todos os elementos "p" (parágrafo) na página. As propriedades "color" e "font-size" são usadas para definir a cor do texto do parágrafo e o tamanho da fonte.

Para aplicar esse estilo a um parágrafo em sua página HTML, você pode adicionar uma classe ao seu elemento HTML e adicionar o estilo à classe no arquivo CSS. Aqui está um exemplo básico:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Minha página</title>
    <link rel="stylesheet" type="text/css" href="estilo.css">
  </head>
  <body>
    <p class="meu-estilo">Este é um exemplo de parágrafo com estilo.</p>
  </body>
</html>
```

Agora veja como ficará o CSS:

```
.meu-estilo {
  color: blue;
  font-size: 16px;
}
```

Nesse exemplo, a classe "meu-estilo" é adicionada ao parágrafo, e o estilo definido na classe é aplicado ao elemento. Isso permite que você reutilize estilos em vários elementos HTML em sua página.

JavaScript

JavaScript é uma linguagem de programação de alto nível, interpretada e orientada a objetos, que é amplamente usada no desenvolvimento web para criar interatividade e dinamismo em páginas web e aplicações web.

Aqui está um exemplo básico de código JavaScript que mostra uma mensagem de saudação personalizada:

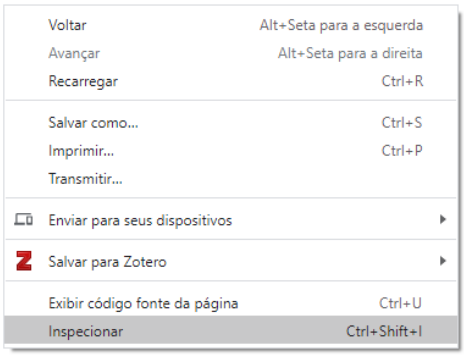
```
<!DOCTYPE html>
<html>
  <head>
    <title>Minha página</title>
  </head>
  <body>
    <script>
      let nome = prompt("Qual é o seu nome?");
      alert(`Olá, ${nome}! Bem-vindo à minha página.`);
    </script>
  </body>
</html>
```

Nesse exemplo, o código JavaScript é adicionado diretamente na seção "body" do documento HTML, usando a tag "script". Quando a página é carregada, o script exibe uma caixa de diálogo pedindo ao usuário para inserir seu nome e, em seguida, exibe uma mensagem de saudação personalizada usando o nome inserido.

O JavaScript é uma linguagem muito poderosa e versátil, e é usada em muitas áreas diferentes do desenvolvimento web, desde validação de formulários e animações até a criação de aplicativos web complexos e de grande escala.

Inspecionar elemento

Navegadores modernos fornecem ferramentas para auxílio no desenvolvimento de aplicações. Por exemplo, o Chrome fornece a ferramenta inspecionar. Você pode acessá-la clicando com o botão direito em qualquer lugar da página e clicando em seguida na opção “inspecionar”, conforme pode ser visto na figura a seguir:



Isso irá carregar uma barra inferior. Você pode visualizar o código fonte de sua aplicação clicando em "*Elements*" ou utilizar o "*Console*" para testar códigos em JavaScript (conforme pode ser visto na figura abaixo) ou analisar a saída dos scripts executados pela aplicação.

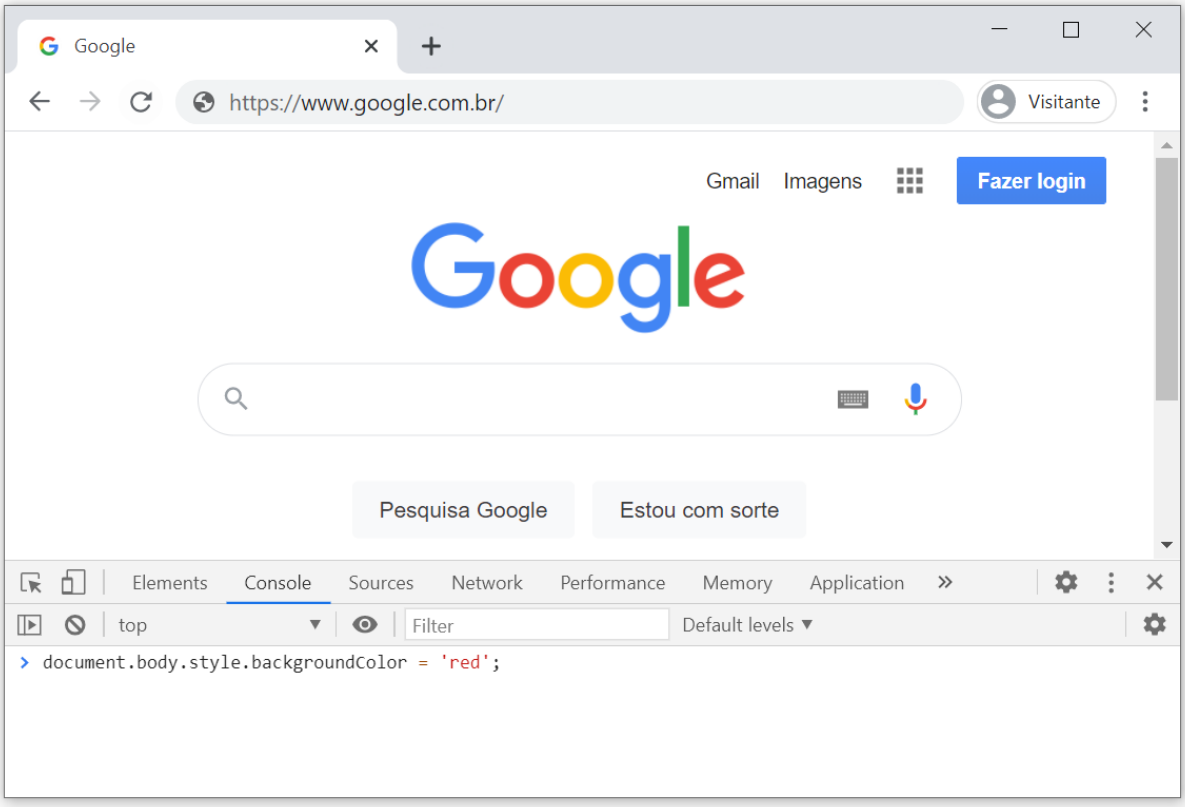


Chart.js

Chart.js é uma biblioteca JavaScript de código aberto que permite a criação de gráficos e visualizações de dados interativos na web. É uma das bibliotecas mais populares para visualização de dados e oferece suporte a vários tipos de gráficos, incluindo gráficos de linhas, barras, pizza, radar, polares, dispersão e outros.

A biblioteca é construída usando a tecnologia Canvas do HTML5 para renderizar gráficos na tela. Além disso, ela é altamente personalizável, permitindo que os usuários definam propriedades como cores, tamanho, estilo de fonte, legendas, tooltips e muito mais.

Chart.js é fácil de usar e inclui documentação detalhada e exemplos de código para ajudar os usuários a começar rapidamente. Ele também tem uma grande comunidade de desenvolvedores, tornando mais fácil encontrar suporte e recursos adicionais. É uma escolha popular para desenvolvedores que desejam criar visualizações de dados dinâmicas e interativas em seus projetos da web.

Criando a base de uma aplicação

Para criar um gráfico de barras simples com Chart.js, siga os seguintes passos:

- Adicione a biblioteca Chart.js à sua página HTML. Você pode fazer isso baixando a biblioteca e adicionando um link para o arquivo "chart.js" em sua página HTML ou usando um CDN (*Content Delivery Network*), como o seguinte:

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

Adicione um elemento "canvas" à sua página HTML. O gráfico será desenhado dentro desse elemento.

```
  <canvas id="meu-grafico"></canvas>
```

Agora, crie um arquivo chamado “script.js” e vamos colocar os códigos referentes ao Chart.js nesse arquivo. Veja como ficará ao final seu arquivo HTML:

```
<!DOCTYPE html>
<html>
 <head>
```

```
  <title>Chart.js</title>

 <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

 </head>
 <body>
  <canvas id="meu-grafico"></canvas>

  <script src="script.js"></script>
 </body>
</html>
```

No seu arquivo “script.js”, você deverá colocar todos os código necessários para criar os gráficos. Começaremos obtendo o local onde o gráfico será gravado:

```
const ctx = document.getElementById('meu-grafico');
```

Agora, para criar um gráfico, precisaremos utilizar os comandos:

```
new Chart(ctx, {
 // ...
});
```

Note que na linha 2 do código acima precisaremos colocar algumas propriedades, como por exemplo, *type* (tipo de gráfico, como barras, linhas, etc.), *data* (onde passaremos os dados) e *options* (onde faremos configurações avançadas). Vamos ver na prática como podemos implementar isso.

Exemplos práticos

Gráfico de barras

Crie um contexto para o elemento "canvas" e inicialize um novo gráfico de barras.

```
const ctx = document.getElementById('meu-grafico');
```

```
new Chart(ctx, {
  type: 'bar',
  data: {
    labels: ['Vermelho', 'Azul', 'Amarelo', 'Verde', 'Roxo', 'Laranja'],
    datasets: [{
      label: '# votos',
      data: [12, 19, 3, 5, 2, 3],
      borderWidth: 1
    }]
  },
  options: {
    scales: {
      y: {
        beginAtZero: true
      }
    }
  }
});
```

Nesse exemplo, criamos um gráfico de barras com seis barras correspondentes a seis cores. Veja como ficará o gráfico.

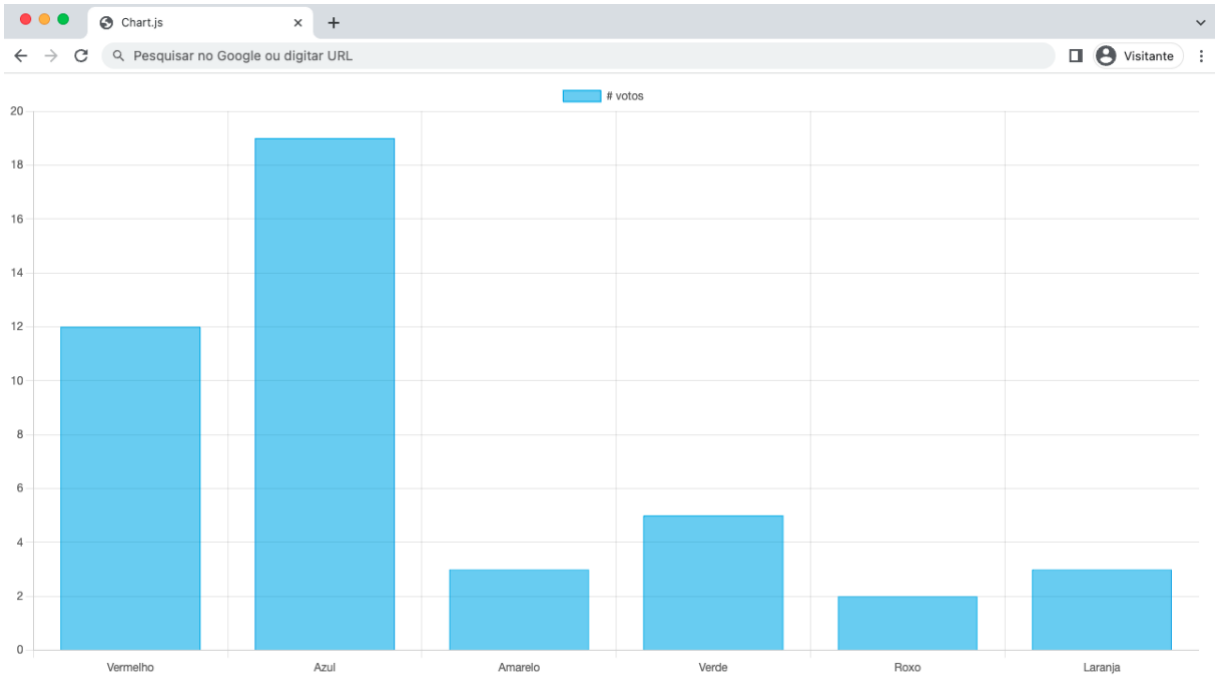


Gráfico de barras empilhadas

Agora, vamos aprender a construir um gráfico de barras empilhadas. Com Chart.js isso é bastante simples. Vamos começar criando dois elementos <canvas>: (1) para o gráfico de barras normal e (2) para o gráfico de barras empilhadas.

```
<!doctype html>
<html lang="pt-br">
<head>
    <title>Meu gráfico</title>
    <meta charset="utf-8">
    <!-- CSS -->
    <link rel="stylesheet" href="estilo.css">
</head>
<body>
    <div id="caixa">
        <canvas id="meu-grafico1"></canvas>
        <canvas id="meu-grafico2"></canvas>
    </div>
    <!-- SCRIPT -->
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

```
    <script src="script.js"></script>
</body>
</html>
```

Vamos configurar um arquivo de estilos:

```
#caixa{
    max-height: 500px;
    margin: 0 auto;
    width: 960px;
}

canvas{
    max-width: 400px;
    margin: 20px;
    display: inline !important
}
```

Agora, vamos construir o script:

```
let tipo = 'bar';
let dados = {
    // rótulos
    labels: ["Vermelho", "Azul", "Amarelo", "Verde", "Roxo", "Laranja"],
    // conjuntos de dados
    datasets: [
        {
            label: "# votos",
            data: [12, 19, 3, 5, 2, 3],
            borderWidth: 1,
            borderColor: "#1E90FF",
            backgroundColor: "#1E90FF"
        },
```

```
        {
            label: "objeto 2",
            data: [10, 20, 1, 10, 4, 3],
            borderWidth: 1,
            borderColor:  "#FF6347",
            backgroundColor: "#FF6347"
        }
    ]
};
let opcoes = {
    // escala
    scales: {
        y: {
            beginAtZero: true,
            stacked: true
        },
        x: {
            stacked: true
        }
    },
    plugins: {
        title: {
            display: true,
            text: "Gráfico de barras empilhadas"
        }
    }
};

// gráfico 1 - gráfico de barras
const ctx1 = document.querySelector("#meu-grafico1");

new Chart(ctx1, {
```

```
    // type = tipo do gráfico
    type: tipo,
    // data = recebe os dados
    data: dados,
    // options = options de configuração do gráfico
    options: {
        // escala
        scales: {
            y: {
                beginAtZero: true
            },
        },
        plugins: {
            title: {
                display: true,
                text: "Gráfico de barras"
            }
        }
    }
})

// gráfico 2 - gráfico de barras empilhadas
const ctx2 = document.querySelector("#meu-grafico2");

new Chart(ctx2, {
    // type = tipo do gráfico
    type: tipo,

    // data = recebe os dados
    data: dados,

    // options = options de configuração do gráfico
```

```
    options: opcoes
})
```

Veja uma comparação entre os gráficos lado a lado:

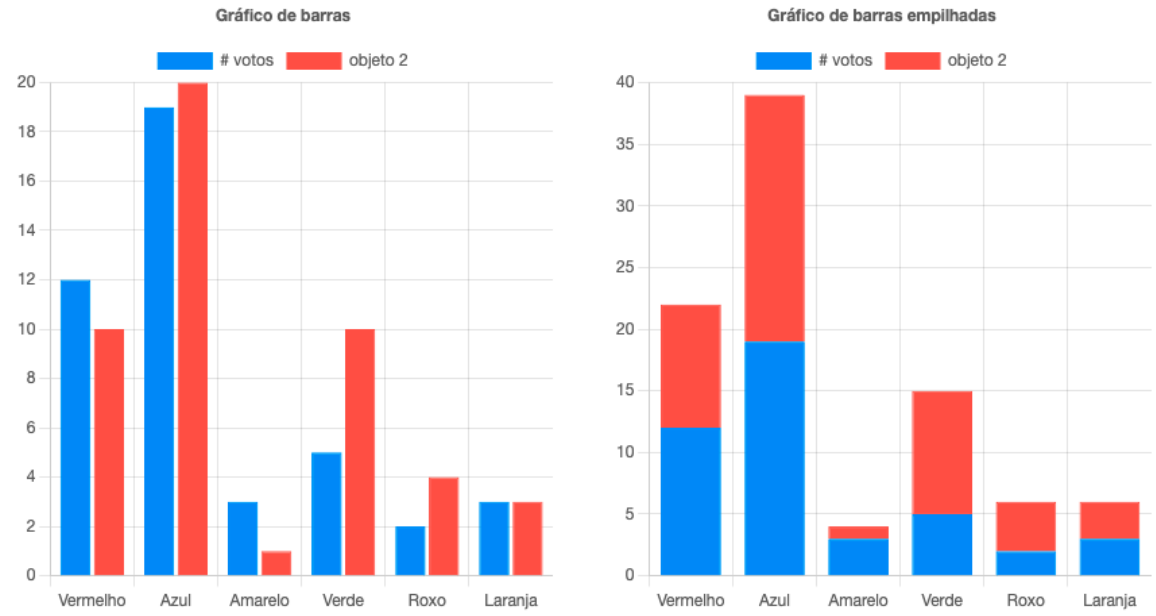


Gráfico de linhas

Agora, vamos construir um gráfico de linhas. Veja como ficou nosso arquivo HTML:

```
<!doctype html>
<html lang="pt-br">

<head>
    <title>Meu gráfico</title>
    <meta charset="utf-8">

    <!-- CSS -->
    <link rel="stylesheet" href="estilo.css">
</head>

<body>
```

```
    <div id="caixa">

        <canvas id="meu-grafico3"></canvas>

    </div>

    <!-- SCRIPT -->
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="script.js"></script>
</body>
</html>
```

Agora, veja como ficará no nosso “script.js”:

```
let dados = {
    // rótulos
    labels: ["Vermelho", "Azul", "Amarelo", "Verde", "Roxo", "Laranja"],
    // conjuntos de dados
    datasets: [
        {
            label: "# votos",
            data: [12, 19, 3, 5, 2, 3],
            borderWidth: 1,
            borderColor: "#1E90FF",
            backgroundColor: "#1E90FF"
        },
        {
            label: "objeto 2",
            data: [10, 20, 1, 10, 4, 3],
            borderWidth: 1,
            borderColor:  "#FF6347",
            backgroundColor: "#FF6347"
```

```
        }
    ]
};

// gráfico 3 - gráfico de linhas
const ctx3 = document.querySelector("#meu-grafico3");

new Chart(ctx3, {
    // type = tipo do gráfico
    type: 'line',

    // data = recebe os dados
    data: dados,

    // options = options de configuração do gráfico
    options: { plugins: {
        title: {
            display: true,
            text: "Gráfico de linhas"
        }
    }}
})
```

Seu gráfico de linhas será renderizado da seguinte forma:

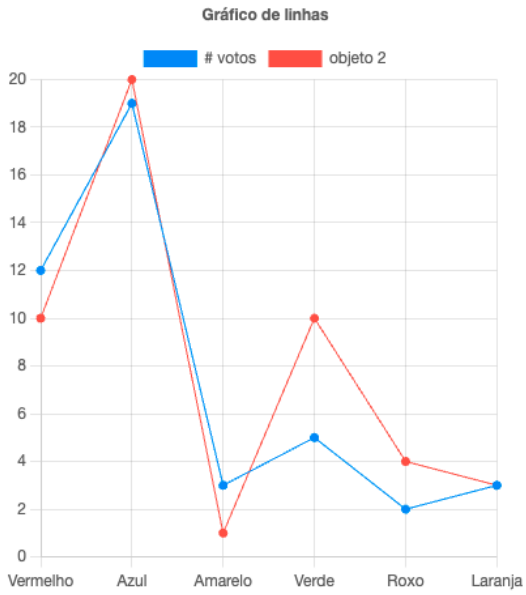


Gráfico de pizza

Código HTML:

```
<!doctype html>
<html lang="pt-br">
<head>
    <title>Meu gráfico</title>
    <meta charset="utf-8">
    <!-- CSS -->
    <link rel="stylesheet" href="estilo.css">
</head>
<body>

    <div id="caixa">
        <canvas id="meu-grafico4"></canvas>
    </div>
```

```
    <!-- SCRIPT -->
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="script.js"></script>
</body>
</html>
```

Script:

```
// gráfico 4 - gráfico de pizza
const ctx4 = document.querySelector("#meu-grafico4");

new Chart(ctx4, {
    // type = tipo do gráfico
    type: 'pie',

    // data = recebe os dados
    data: {
        labels: ["Vermelho", "Azul", "Amarelo", "Verde", "Roxo", "Laranja"],

        datasets: [
            {
                label: "# votos",
                data: [12, 19, 3, 5, 2, 3],
            }
        ]
    },

    // options = options de configuração do gráfico
    options: {
        responsive: true,
        plugins: {
```

```
        legend: {
            position: 'top',
        },
        title: {
            display: true,
            text: "Gráfico de pizza"
        }
    }}
})
```

Resultado:

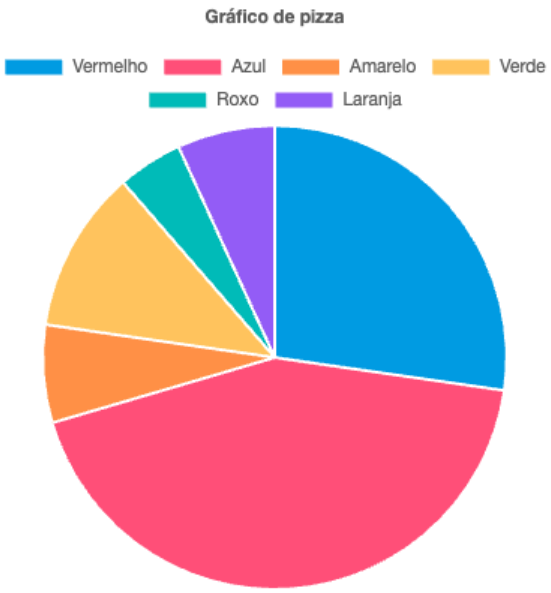


Gráfico de donut

Código HTML:

```
<!doctype html>
<html lang="pt-br">

<head>
    <title>Meu gráfico</title>
```

```
    <meta charset="utf-8">

    <!-- CSS -->
    <link rel="stylesheet" href="estilo.css">
</head>

<body>
    <div id="caixa">
        <canvas id="meu-grafico5"></canvas>
    </div>

    <!-- SCRIPT -->
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="script.js"></script>
</body>
</html>
```

Script:

```
// gráfico 5 - gráfico de donuts
const ctx5 = document.querySelector("#meu-grafico5");

new Chart(ctx5, {
    // type = tipo do gráfico
    type: 'doughnut',

    // data = recebe os dados
    data: {
        labels: ["Vermelho", "Azul", "Amarelo", "Verde", "Roxo", "Laranja"],

        datasets: [
            {
                label: "# votos",
```

```
            data: [12, 19, 3, 5, 2, 3],
            backgroundColor: ["red", "blue", "yellow", "green", "purple", "orange"]
        }
    ]
},

// options = options de configuração do gráfico
options: {
    responsive: true,
    plugins: {
        legend: {
            position: 'top',
        },
        title: {
            display: true,
            text: "Gráfico de donut"
        }
}}
})
```

Resultado:

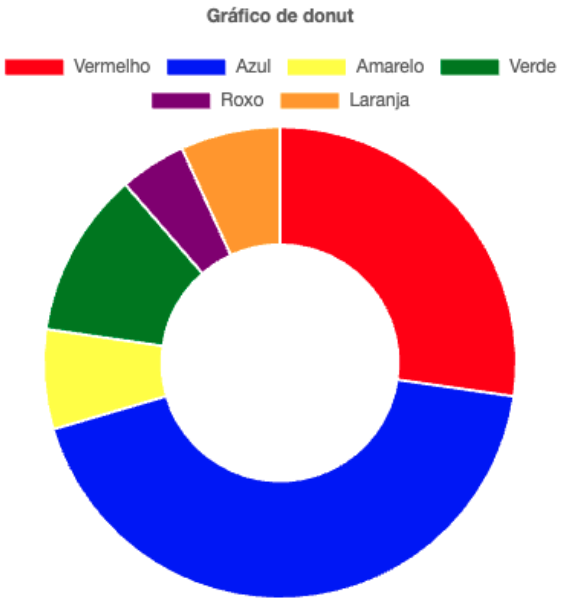
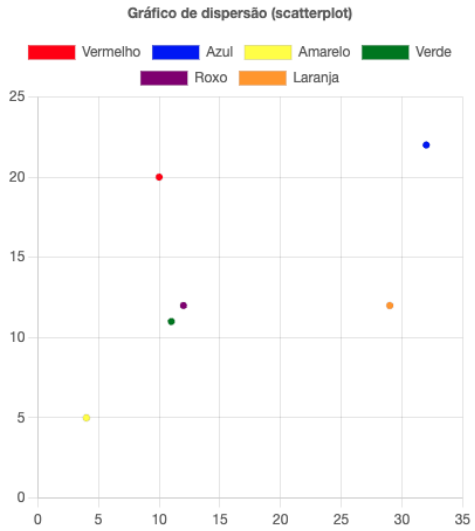


Gráfico de dispersão

Nesta seção, você irá aprender a construir este gráfico:



Código HTML:

```
<!doctype html>
<html lang="pt-br">

<head>
    <title>Meu gráfico</title>
    <meta charset="utf-8">

    <!-- CSS -->
    <link rel="stylesheet" href="estilo.css">
</head>

<body>

    <div id="caixa">

        <canvas id="meu-grafico6"></canvas>
    </div>
```

```
    <!-- SCRIPT -->
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="script.js"></script>
</body>
</html>
```

Script:

```
// gráfico 6 - gráfico de dispersão (scatterplot)
const ctx6 = document.querySelector("#meu-grafico6");

new Chart(ctx6, {
    // type = tipo do gráfico
    type: 'scatter',

    // data = recebe os dados
    data: {
        labels: ["Vermelho", "Azul", "Amarelo", "Verde", "Roxo", "Laranja"],

        datasets: [
            {
                label: "Vermelho",
                data: [{x: 10, y: 20}],
                backgroundColor: "red",
            },
            {label: "Azul", data: [{x:32, y:22}], backgroundColor: "blue"},
            {label: "Amarelo", data: [{x:4, y:5}], backgroundColor: "yellow"},
            {label: "Verde", data: [{x:11, y:11}], backgroundColor: "green"},
            {label: "Roxo", data: [{x:12, y:12}], backgroundColor: "purple"},
            {label: "Laranja", data: [{x:29, y:12}], backgroundColor: "orange"},

        ]
```

```
  },

  // options = options de configuração do gráfico
  options: {
    responsive: true,
    plugins: {
      legend: {
        position: 'top',
      },
      title: {
        display: true,
        text: "Gráfico de donut"
      }
    },
    scales: {
      y: {
        beginAtZero: true,
      }
    },

  }
})
```

Este é um código JavaScript que utiliza a biblioteca Chart.js para criar um gráfico de dispersão (scatterplot). Vamos analisar as diferentes partes do código:

Selecionando o elemento HTML que irá receber o gráfico:

```
const ctx6 = document.querySelector("#meu-grafico6");
```

Aqui, estamos selecionando o elemento HTML que possui o ID "meu-grafico6" e armazenando-o na variável "ctx6". Este elemento será o container onde o gráfico será renderizado.

Criando o gráfico:

```
new Chart(ctx6, {
    type: 'scatter',

    data: {
        ...
    },

    options: {
        ...
    }
})
```

Aqui, estamos criando um novo objeto Chart.js com o elemento HTML selecionado na variável "ctx6". O objeto recebe dois parâmetros: um objeto de configuração de dados (data) e um objeto de configuração de opções (options).

Configurando o objeto de dados:

```
data: {
        labels: ["Vermelho", "Azul", "Amarelo", "Verde", "Roxo", "Laranja"],

        datasets: [
            {
                label: "Vermelho",
                data: [{x: 10, y: 20}],
```

```
            backgroundColor: "red",
        },
        {label: "Azul", data: [{x:32, y:22}], backgroundColor: "blue"},
        {label: "Amarelo", data: [{x:4, y:5}], backgroundColor: "yellow"},
        {label: "Verde", data: [{x:11, y:11}], backgroundColor: "green"},
        {label: "Roxo", data: [{x:12, y:12}], backgroundColor: "purple"},
        {label: "Laranja", data: [{x:29, y:12}], backgroundColor: "orange"},


    ]
},
```

Aqui, estamos definindo os dados que serão exibidos no gráfico. O objeto de dados possui um array de labels (rótulos) e um array de datasets (conjunto de dados). Cada dataset possui uma label (rótulo), um array de dados e uma cor de fundo (backgroundColor).

Os dados para este gráfico de dispersão são definidos no array "data" dentro de cada dataset. Cada par ordenado de coordenadas "x" e "y" representa um ponto no gráfico. No exemplo, estamos definindo apenas um ponto para cada dataset.

Configurando o objeto de opções:

```
options: {
    responsive: true,
    plugins: {
        legend: {
            position: 'top',
        },
        title: {
            display: true,
            text: "Gráfico de dispersão (scatterplot)"
        }
    },
    scales: {
        y: {
            beginAtZero: true,
```

```
        }
    },


}
```

Aqui, estamos definindo as opções de configuração para o gráfico. Estas opções incluem a possibilidade de tornar o gráfico responsivo (responsive), configurar a posição da legenda (plugins.legend), adicionar um título ao gráfico (plugins.title) e definir o comportamento da escala vertical (scales.y).

No exemplo, estamos definindo a opção "beginAtZero" da escala vertical como *true*. Isso significa que a escala vertical começará no valor zero. Se não definirmos essa opção, a escala vertical começará no valor mínimo dos dados.

Obtendo dados de arquivos

Nos exemplos anteriores, os dados foram inseridos diretamente no código JS, mas nem sempre você poderá obter dados desta forma. Em geral, precisaremos ler arquivos externos. Em JavaScript podemos usar a função fetch() para ler arquivos.

Importante

Caso esteja tentando ler um arquivo local, você precisará de um servidor web configurado. Você pode obter um em https://www.apachefriends.org/pt_br/index.html.

Para executá-lo, abra o terminal de linhas de comando, navegue até a pasta desejada e inicie o servidor usando o comando:

php -S localhost:8000

Por fim, acesse o site no navegador usando o endereço: http://localhost:8000

Neste exemplo, vamos ler o arquivo: “autores.csv”. Esse arquivo está no formato CSV e contém uma lista de autores, número de citações, número de artigos e categoria (que pode ser “1000” para autores no top 1000, ou seja, os 1000 autores mais citados, “100” ou “10” para top100 e top10, respectivamente).

Fragmento do arquivo:

```
Name,Citations,Articles,Ranking
I B Rogozin,103,7,1000
R Stevens,48,7,1000
C A Goble,48,4,1000
S Bechhofer,48,4,1000
R Apweiler,189,16,1000
W Kaplan,73,1,1000
T G Littlejohn,73,2,1000
R Stevens,52,13,1000
...
```

O arquivo completo está disponível em:

https://raw.githubusercontent.com/dcbmariano/chartjs/main/autores.csv

Para este exemplo, vamos criar um arquivo HTML denominado: “lerarquivo.html”.

```
<!doctype html>
<html lang="pt-br">

<head>
   <title>Lendo arquivos externos</title>
   <meta charset="utf-8">

   <!-- CSS -->
   <link rel="stylesheet" href="estilo.css">
</head>

<body>

   <div id="caixa">
      <canvas id="meu-grafico1"></canvas>

   </div>

   <!-- SCRIPT -->
```

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script>

    const url = "autores.csv";
    //const url = "https://raw.githubusercontent.com/dcbmariano/chartjs/main/autores.csv";


    fetch(url)
        .then(resposta => resposta.text())
        .then(data => {
            data = data.split("\n")
            data = data.map(i=>i.split(","))

            let top10 = data.filter(i=>i[3] == '10')
            console.log(top10)

            // local
            const local = document.querySelector("#meu-grafico1");

            let dados = {
                // rótulos
                labels: top10.map(i=>i[0]),
                // conjuntos de dados
                datasets: [
                    {
                        label: "Citações",
                        data: top10.map(i=>i[1]),
                        borderWidth: 1,
                        borderColor: "#1E90FF",
                        backgroundColor: "#1E90FF"
                    },
                    {
                        label: "Artigos",
```

```
                    data: top10.map(i=>i[2]),
                    borderWidth: 1,
                    borderColor: "red",
                    backgroundColor: "red"
                }
            ],
        };

        new Chart(local, {
            type: "bar",
            data: dados,
            options: {
                scales: {
                    y: {
                        display: true,
                        type: 'logarithmic',
                    }
                }
            }
        })
    })
</script>
</body>
</html>
```

Este código é um exemplo de como ler um arquivo CSV (Comma-Separated Values) em JavaScript, utilizando o método fetch(), e utilizar os dados para gerar um gráfico com a biblioteca Chart.js.

Na seção <head>, há um link para o arquivo CSS externo estilo.css, que pode ser utilizado para definir a aparência da página.

Dentro do elemento <body>, há uma div com id "caixa" e um canvas com id "meu-grafico1". O objetivo deste exemplo é ler um arquivo CSV com informações de autores e suas citações e artigos, e gerar um gráfico de barras com as informações dos 10 autores com mais citações.

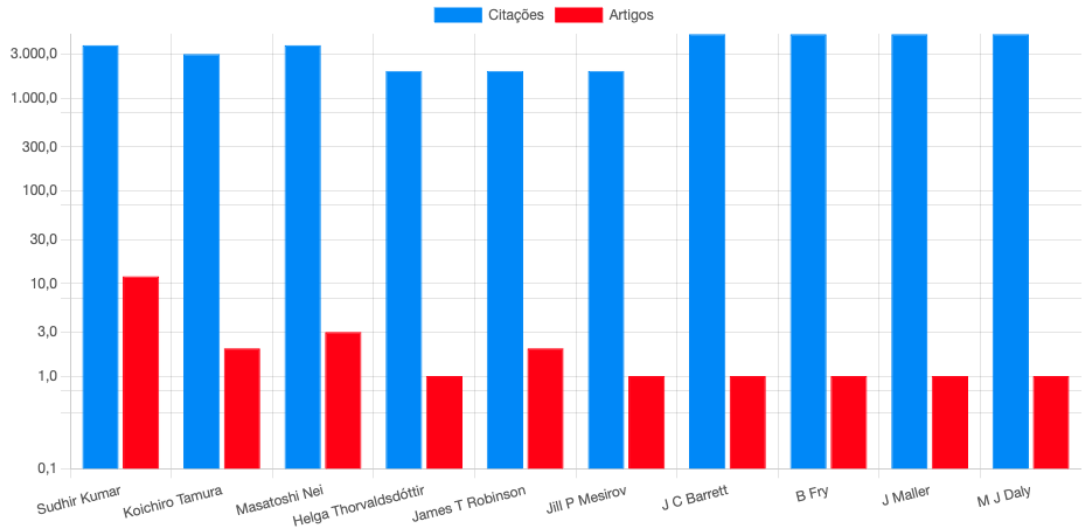
Para isso, é utilizado o método fetch() para fazer uma requisição do arquivo CSV, que é salvo na variável url. Quando a resposta é recebida, o método .text() é utilizado para transformar o conteúdo da resposta em uma string.

Em seguida, a string é dividida em linhas (cada linha representa um autor), utilizando o método .split("\n"). Cada linha também é dividida em colunas (separadas por vírgula), utilizando o método .map(i=>i.split(",")).

Os dados dos 10 autores com mais citações são selecionados com o método .filter(i=>i[3] == '10'), e são utilizados para gerar o gráfico. O canvas com id "meu-grafico1" é selecionado com o método document.querySelector("#meu-grafico1") e é utilizado para renderizar o gráfico.

Os dados do gráfico são definidos na variável dados, que possui rótulos e conjuntos de dados. O gráfico é criado com new Chart(local, {type: "bar", data: dados, options: {scales: {y: {display: true, type: 'logarithmic',}}}}), onde local é o canvas selecionado, "bar" é o tipo de gráfico escolhido, dados são os dados do gráfico, e options são as opções de exibição do gráfico (no caso, o eixo y é definido como escala logarítmica).

Ao final, este gráfico será produzido:



O código fonte desta seção está disponível em: https://github.com/dcbmariano/chartjs.

Desafio: tente ler o arquivo a seguir no formato JSON

```
[
    [
        "Sudhir Kumar",
        "3768",
        "12",
        "10"
    ],
    [
        "Koichiro Tamura",
        "3022",
        "2",
        "10"
    ],
    [
        "Masatoshi Nei",
        "3768",
        "3",
        "10"
    ],
    [
        "Helga Thorvaldsdóttir",
        "1980",
        "1",
        "10"
    ],
    [
        "James T Robinson",
        "1980",
        "2",
        "10"
    ],
    [
        "Jill P Mesirov",
        "1980",
        "1",
        "10"
    ],
    [
        "J C Barrett",
        "4959",
        "1",
        "10"
    ],
    [
        "B Fry",
        "4959",
        "1",
        "10"
    ],
```

```
    [
        "J Maller",
        "4959",
        "1",
        "10"
    ],
    [
        "M J Daly",
        "4959",
        "1",
        "10"
    ]
]
```