

# **PROGRAMMING IN HTML5 WITH JS AND CSS3**

## **Exame 70-480**

### **Ray Carneiro**

<https://www.youtube.com/watch?v=vmJ896cWXCQ&list=PLwftZeDnOzt1Bbkn0Ue87GsYAe4TcUSSI>

Resumo feito por Roberto Pinheiro

## **CAPÍTULO 1 - ESTRUTURAS DE DOCUMENTOS HTML5**

Os desenvolvedores Web hoje precisam entender as complexidades da construção de aplicativos interativos e dinâmicos com HTML e JavaScript. A introdução do HTML5 trouxe um novo padrão para definir a estrutura de suas páginas da web como bem como mudanças em como você interage com eles via script.

Este capítulo demonstra como criar documentos HTML com a nova marcação semântica HTML5. Você vai explorar o processo de criação do código necessário para manipular e interagir com a marcação HTML5 e aplicar estilos para elementos HTML5.

OBS.: A marcação HTML é conhecida como tags HTML e elementos HTML. Esses termos são frequentemente usados indistintamente. Este livro se refere à marcação HTML como elementos.

Objetivos neste capítulo:

- Aula 1: Criar a estrutura do documento
- Aula 2: Escrever código que interagem com controles UI
- Aula 3: Aplicar estilos para elementos HTML programaticamente
- Aula 4: Implementar APIs HTML5
- Aula 5: Estabelecer o escopo de objetos e variáveis
- Aula 6: Criar e implementar objetos e métodos

# Aula 1 - Criando a estrutura do documento

O objetivo da estrutura de um documento é dizer ao navegador como o conteúdo deve ser exibido. Sem qualquer estrutura declarativa em sua página, o navegador não detectará nenhuma estrutura, então ele irá organizar seu conteúdo de acordo com as regras implementadas por seu mecanismo de renderização. Ao usar a marcação HTML5 apresentada neste objetivo, você está dizendo ao navegador para levar em consideração sua semântica ao exibir a página. Novos lançamentos de navegadores irão incorporar mais e mais dos padrões HTML5 em seus motores de renderização

Nesta aula veremos como:

- Usar marcação semântica do HTML5
- Criar um conteúdo de layout em HTML
- Otimizar para mecanismos de pesquisa
- Otimizar para leitores de tela

## Usando marcação semântica do HTML5

**TABLE 1-1** HTML5 semantic markup

HTML5 element	Description
<article>	Defines self-contained areas on a page
<aside>	Defines smaller content areas outside the flow of a webpage
<figcaption>	Defines the caption of a figure element
<figure>	Defines content that contains a figure, such as an image, chart, or picture
<footer>	Defines the bottom of a section or page
<header>	Defines the top of a section or page
<hgroup>	Defines a group of headings (H1–H6 elements)
<mark>	Defines text that should be highlighted
<nav>	Defines navigation to other pages in the site
<progress>	Defines the progress of the task
<section>	Defines the distinct content of a document

## Compreendendo a estrutura central de uma página HTML5

O código HTML a seguir, demonstra o modelo básico de uma página HTML5:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title></title>
  </head>
  <body>
    <!-- page content goes here -->
  </body>
</html>
```

Nesta seção, você começará com uma estrutura de documento HTML simples que você usará em muitos outros exemplos para destacar vários conceitos.

### Elementos `<header>` e `<footer>`

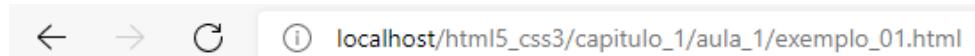
A maioria dos documentos de página da web contém conteúdo comum na parte superior e inferior de todas as páginas. Embora o uso dos elementos `<header>` e `<footer>` não forneça automaticamente essa funcionalidade, os elementos fornecem a capacidade de definir o conteúdo no cabeçalho e rodapé do site.

Normalmente, o cabeçalho de uma página da Web contém conteúdo, como o logotipo ou banner de uma empresa. Em alguns casos, ele também pode conter um menu de navegação. Comece a página de exemplo adicionando o elemento `<header>` à sua página:

#### **aula1\exemplo\_01.html**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>TITULO DA PAGINA</title>
  </head>

  <body>
    <header>
      <h1>Some fictional company Website</h1>
    </header>
  </body>
</html>
```



## Some fictional company Website

O elemento <header> não se limita apenas ao início de sua página, ele fornece uma forma semântica de declarar o cabeçalho para qualquer área da página da web. Você pode usar o elemento <header> como um cabeçalho para um elemento <section> ou para um elemento <article>. O elemento <header> é destinado a conter um elemento H1 a H6 conforme necessário; no entanto, você pode preencher um cabeçalho com qualquer marcação que atenda às suas necessidades para criar o melhor cabeçalho para aquela área específica do site.

### aula1\exemplo\_02.html

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <title>TITULO DA PAGINA</title>
    </head>

    <body>
        <header>
            <h1>Some fictional company Website</h1>
        </header>
        <article>
            <header>
                <h1>Our first new Article</h1>
            </header>
            <section>
                <h1>Section 1</h1>
            </section>
        </article>
    </body>
</html>
```

← → ⌂ ⓘ localhost/html5\_css3/capitulo\_1/aula\_1/exemplo\_02.html

# Some fictional company Website

## Our first new Article

### Section 1

## Elemento <nav>

Usar o elemento <nav> em um documento HTML5 fornece aos usuários navegação através dos elementos principais do documento da web ou do aplicativo da web como um todo. Esta navegação principal pode ser representada como uma lista de links na parte superior da página para navegar no site atual. Também pode listar seus sites favoritos na lateral da página, como em um blog onde você lista outros blogs favoritos que segue.

Normalmente, a lista de links na parte superior, comumente conhecida como o menu principal, está contido no cabeçalho (mas não precisa estar). Uma lista de URLs favoritos provavelmente será colocada em um <aside> para que a lista possa ser colocada ao lado, longe do conteúdo principal, mas facilmente acessível. Para o exemplo atual, você cria um principal menu na parte superior da página.

**aula1\exemplo\_03.html**

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <title>TITULO DA PAGINA</title>
    </head>

    <body>
        <header>
            <h1>Some fictional company Website</h1>
            <nav>
                <a href="Home.html">Document Structure</a>
                <a href="Blog.html">Writing Code</a>
                <a href="About.html">Styles</a>
            </nav>
        </header>
    </body>
</html>
```

← → ⌂ ⓘ localhost/html5\_css3/capitulo\_1/aula\_1/exemplo\_03.html

# Some fictional company Website

[Document Structure](#) [Writing Code](#) [Styles](#)

## Elemento <hgroup>

Outro elemento principal comumente usado no elemento <header> é o denominado <hgroup>.

O elemento <hgroup> é um método semântico que organiza cabeçalhos e subcabeçalhos. Esse elemento normalmente contém os elementos <h1> a <h6> padrão e familiares. O elemento <hgroup> agrupa cabeçalhos relacionados em sequência.

**aula1\exemplo\_04.html**

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <title>TITULO DA PAGINA</title>
    </head>

    <body>
        <header>
            <hgroup>
                <h1>Some fictional company Website</h1>
            </hgroup>
            <nav>
                <a href="Home.html">Document Structure</a>
                <a href="Blog.html">Writing Code</a>
                <a href="About.html">Styles</a>
            </nav>
        </header>
    </body>
</html>
```



# Some fictional company Website

[Document Structure](#) [Writing Code](#) [Styles](#)

## Elemento <article>

Um elemento <article> representa uma composição ou entrada inteira e completa. Exemplos de um elemento <article> pode ser um artigo de revista ou uma postagem de blog, onde o conteúdo pode ser redistribuído de forma independente e não perder o seu significado.

## Elemento <section>

O elemento <section> subdivide as páginas em seções. Você pode continuar a quebrar a página de exemplo com elementos <article> adicionais; no entanto, o objetivo do elemento <article> não é quebrar uma página em detalhes mais granulares. É aqui que o elemento <section> torna-se útil. Cada elemento <article> contém zero ou mais elementos <section> para denotar as diferentes seções de conteúdo dentro do elemento <article>. O primeiro elemento dentro de um elemento <section> é normalmente um cabeçalho ou um grupo de cabeçalhos.

### aula1\exemplo\_05.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>TITULO DA PAGINA</title>
  </head>
  <body>
    <header>
      <h1>Some fictional company Website</h1>
      <nav>
        <a href="Home.html">Document Structure</a>
        <a href="Blog.html">Writing Code</a>
        <a href="About.html">Styles</a>
      </nav>
    </header>
    <article>
      <header>
        <hgroup>
          <h1>Our first new Article</h1>
        </hgroup>
      </header>
      <section>
        <h1>Section 1</h1>
        <p>Some details about section 1</p>
      </section>
      <section>
        <h1>Section 2</h1>
      </section>
    </article>
    <article>
      <header>
        <hgroup>
          <h1>Second huge article</h1>
        </hgroup>
      </header>
      <p>Provide some useful information in the article</p>
    </article>
```

```
<article>
  <header>
    <hgroup>
      <h1>Third huge article</h1>
    </hgroup>
  </header>
  <p>Provide some useful information in the third article</p>
</article>
</body>
</html>
```



## Some fictional company Website

[Document Structure](#) [Writing](#) [Code](#) [Styles](#)

### Our first new Article

#### Section 1

Some details about section 1

#### Section 2

### Second huge article

Provide some useful information in the article

### Third huge article

Provide some useful information in the third article

## Elemento <aside>

O elemento `<aside>` define qualquer conteúdo que não se enquadre no fluxo principal ou no conteúdo principal da página atual - por exemplo, uma barra lateral, uma nota, um alerta ou um anúncio. O elemento `<aside>` não se posiciona automaticamente em nenhum lado específico da página da web; serve apenas como uma forma de definir semanticamente uma seção de texto ou gráfico como um aparte. Mais tarde você verá como posicionar um aparte usando estilos. Por enquanto, adicione o seguinte elemento `<aside>` à sua página para uso posterior:

## aula1\exemplo\_06.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>TITULO DA PAGINA</title>
    </head>

    <body>
        <header>
            <h1>Some fictional company Website</h1>
            <nav>
                <a href="Home.html">Document Structure</a>
                <a href="Blog.html">Writing Code</a>
                <a href="About.html">Styles</a>
            </nav>
        </header>

        <article>
            <header>
                <hgroup>
                    <h1>Our first new Article</h1>
                </hgroup>
            </header>
            <section>
                <h1>Section 1</h1>
                <p>Some details about section 1</p>
                <aside>Did you know that 7/10 is 70%</aside>
            </section>
            <section>
                <h1>Section 2</h1>
            </section>
        </article>

        <article>
            <header>
                <hgroup>
                    <h1>Second huge article</h1>
                </hgroup>
            </header>
            <p>Provide some useful information in the article</p>
        </article>

        <article>
            <header>
                <hgroup>
                    <h1>Third huge article</h1>
                </hgroup>
            </header>
            <p>Provide some useful information in the third article</p>
        </article>

    </body>
</html>
```

# Some fictional company Website

[Document Structure](#) [Writing](#) [Code](#) [Styles](#)

## Our first new Article

### Section 1

Some details about section 1

Did you know that 7/10 is 70%

### Section 2

## Second huge article

Provide some useful information in the article

## Third huge article

Provide some useful information in the third article

## Elementos <figure> e <figcaption>

Os elementos <figcaption> e <figure>, novos no HTML5, fornecem os elementos semânticos necessários para adicionar gráficos e figuras às páginas da web. Esses gráficos e figuras normalmente fornecem uma representação visual da informação no conteúdo textual e referenciada pelo texto.

### aula1\exemplo\_07.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>TITULO DA PAGINA</title>
  </head>

  <body>
    <header>
      <h1>Some fictional company Website</h1>
      <nav>
        <a href="Home.html">Document Structure</a>
        <a href="Blog.html">Writing Code</a>
        <a href="About.html">Styles</a>
      </nav>
    </header>

    <article>
      <header>
        <hgroup>
          <h1>Our first new Article</h1>
        </hgroup>
      </header>
      <section>
        <h1>Section 1</h1>
        <p>Some details about section 1</p>
        <aside>Did you know that 7/10 is 70%</aside>
      </section>
      <section>
        <h1>Section 2</h1>
      </section>
    </article>

    <article>
      <header>
        <hgroup>
          <h1>Second huge article</h1>
        </hgroup>
      </header>
      <p>Provide some useful information in the article</p>
    </article>

    <article>
      <header>
        <hgroup>
          <h1>Third huge article</h1>
        </hgroup>
      </header>
      <p>Provide some useful information in the third article</p>
    </article>
  </body>

```

```

<figure>
    
    <figcaption>Fig 1: A really juicy orange.</figcaption>
</figure>
</article>
</body>
</html>

```

localhost/html5\_css3/capitulo\_1/aula\_1/exemplo\_07.html

# Some fictional company Website

[Document Structure](#) [Writing Code](#) [Styles](#)

## Our first new Article

### Section 1

Some details about section 1

Did you know that 7/10 is 70%

### Section 2

## Second huge article

Provide some useful information in the article

## Third huge article

Provide some useful information in the third article



Fig 1: A really juicy orange.

## Elemento <progress>

O elemento `<progress>` representa o progresso de um objetivo ou tarefa. Há dois tipos de elementos `<progress>`: determinados e indeterminados. Use um elemento `<progress>` determinado quando você sabe com antecedência a quantidade de trabalho a ser completado; em outras palavras, você conhece os valores inicial e final. Cenários de exemplo para este caso, incluem o download de um arquivo do qual você sabe o tamanho exato ou a exibição do progresso de uma arrecadação de fundos. Em ambas as situações, você sabe o status exato da tarefa em qualquer momento específico, e você também sabe qual é o objetivo final - seja o número de bytes para o download do arquivo ou o número de dólares para a arrecadação de fundos. Nestes casos determinados, você pode especificar a marcação HTML5 como esta:

### aula1\exemplo\_08.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>TITULO DA PAGINA</title>
  </head>

  <body>

    <p>Our goal is to have 1000 users:</p>
    <span>0</span>
    <progress value="50" max="1000"></progress>
    <span>1000</span>
  </body>
</html>
```



Our goal is to have 1000 users:

0  1000

Conforme mostrado no código anterior, o elemento `<progress>` tem dois atributos que você precisa saber: valor e máx. O atributo `value` permite que você especifique o valor atual ou a posição do elemento `<progress>` em um ponto específico no tempo. O atributo `max` diz ao navegador qual é o valor máximo possível para o elemento. O navegador usa esses dois valores para determinar quanto do elemento deve ser colorido. Normalmente, o atributo de valor é atualizado dinamicamente usando JavaScript. Na Figura a seguir, você pode ver como a exibição do elemento `<progress>` muda quando o atributo de valor é atualizado para 750.

### aula1\exemplo\_09.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>TITULO DA PAGINA</title>
  </head>

  <body>

    <p>Our goal is to have 1000 users:</p>
    <span>0</span>
    <progress value="750" max="1000"></progress>
    <span>1000</span>
  </body>
</html>
```

Our goal is to have 1000 users:

0 1000

Você usa tarefas indeterminadas quando não sabe quanto tempo uma tarefa levará para ser concluída mas ainda assim deseja mostrar aos usuários que algum trabalho está ocorrendo e que eles devem esperar. Nesse caso, use o elemento <progress>, mas remova o atributo value. Quando você não especifica o atributo value, o navegador entende que o elemento <progress> é uma tarefa do tipo indeterminada. Isso pode ser útil para dados recebidos de um serviço sobre o qual você não tem controle ou conhecimento da rapidez com que a solicitação será concluída ou do tamanho dos resultados da solicitação.

A seguinte marcação HTML5 demonstra uma tarefa indeterminada:

#### aula1\exemplo\_10.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>TITULO DA PAGINA</title>
  </head>

  <body>
    <p>Data download is in progress, please wait patiently:</p>
    <progress max="5"></progress>
  </body>
</html>
```

Data download is in progress, please wait patiently:



---

Data download is in progress, please wait patiently:



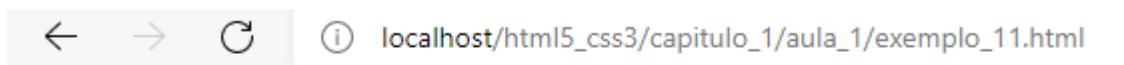
## Elemento <mark>

Com o elemento <mark>, você pode destacar facilmente informações importantes ou qualquer texto que você queira enfatizar. Tem essencialmente a mesma função de um marcador de texto. Envolvendo o texto em um elemento <mark> e fornecendo um atributo background-color para seu elemento style, você pode obter o efeito de realce desejado. O seguinte código HTML demonstra o elemento <mark>:

### aula1\exemplo\_11.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>TITULO DA PAGINA</title>
  </head>

  <body>
    <p>Some very <mark style="background-color:yellow;">important</mark> information provided here!</p>
  </body>
</html>
```



Some very **important** information provided here!

## Elemento <div>

O objetivo desses elementos do HTML5 é substituir o método mais antigo de estruturação de páginas - antes do HTML5 - que usava elementos <div> e nomenclatura de acordo com sua função. Use os novos elementos semânticos conforme apropriado, mas lembre-se que o elemento <div> ainda é bastante útil para estilizar o conteúdo.

## Criando layout de container no HTML

Você pode definir o layout de uma página da Web de várias maneiras. Um objetivo importante aqui é fazê-lo pensar seriamente no layout para que a apresentação da página seja amigável. Se os usuários não conseguirem encontrar o que procuram porque a página inteira é estilizada como um único elemento <p> dentro do elemento <body>, eles provavelmente não voltarão mais. Os dois métodos mais comuns de criação de um layout em HTML envolvem o uso de <div> e elementos <table>. Em ambos os casos, é mais do que provável que você ainda use CSS para ajudar no posicionamento e dimensionamento.

```
<div id="PageHeader"></div>
<div id="LeftSide"></div>
<div id="RightSide"></div>
<div id="Footer"></div>
```

O mecanismo de renderização exibe cada `<div>` de acordo com suas regras. Para posicionar as divisões dinamicamente usa-se CSS. O principal problema com o uso de elementos `<div>` para estruturar o documento é a sua incapacidade de dar significado semântico padrão a cada seção.

O elemento `<div>` permite recursos mais dinâmicos no layout da página. Para um layout mais estático declarado diretamente na página HTML, o elemento `<table>` é mais apropriado. No exemplo a seguir é definida uma tabela que fornece um formato de site de blog comum, com uma seção de cabeçalho, à esquerda barra lateral, uma área de conteúdo, uma barra lateral direita e uma área de rodapé:

### **aula1\exemplo\_12.html**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>TITULO DA PAGINA</title>
        <style>
            table {
                border:1px solid black;
                width:100%;
                height:600px;
                border-collapse: collapse;
            }
            tr, td {
                border:1px solid black;
            }
        </style>
    </head>
    <body>
        <table>
            <tr>
                <td colspan="3" id="Header"></td>
            </tr>
            <tr>
                <td rowspan="3" id="LeftBar"></td>
                <td rowspan="3" id="MainContent"></td>
                <td id="RightSideTop"></td>
            </tr>
            <tr>
                <td id="RightSideMiddle"></td>
            </tr>
            <tr>
                <td id="RightSideBottom"></td>
            </tr>
            <tr>
                <td colspan="3" id="Footer"></td>
            </tr>
        </table>
    </body>
</html>
```


O elemento `<table>` é muito flexível. Elementos adicionais como `<thead>` e `<tfoot>` fornecem uma abordagem mais semântica para rotular as células da tabela. A preocupação em usar o elemento `<table>` é a natureza estática da estrutura. Para mudar a estrutura geral de um site que usa tabelas para layout, você precisa ir a cada página e fazer as alterações.

## Otimizando para mecanismos de busca

Você deve garantir que seu site possa ser encontrado entre os milhões de sites que já existem. A otimização do mecanismo de pesquisa (SEO) é uma técnica usada para tornar os elementos do site facilmente detectáveis e apropriadamente indexados por mecanismos de pesquisa, de modo que quando os usuários pesquisarem conteúdo relacionado ao seu site, eles encontrem suas páginas. Mecanismos de busca como Bing e Google vasculham constantemente a Internet em busca de conteúdo. Quando encontram páginas da web, eles passam pelo HTML e indexam o conteúdo, como página e metadados de imagem. Eles usam os dados indexados para permitir que os usuários pesquisem basicamente qualquer coisa na Internet e recebam resultados relevantes. Claramente, então, existe um relacionamento entre o conteúdo de seus sites e a facilidade com que os usuários podem encontrar seus sites usando um motor de busca.

Antes do HTML5, os web designers costumavam usar elementos `<div>` para segmentar a página. Esses elementos não fornecem muito contexto sobre o que eles pretendem conter. Mas com a marcação semântica disponível em HTML5, você pode usar elementos mais descritivos para as seções da página. Como você viu no exemplo de layout da página do blog, os elementos HTML por si só, deixe clara a intenção de cada segmento da página. Conforme os mecanismos de pesquisa vasculham as páginas da web, eles detectam a marcação e sabem o que tirar dela para indexar corretamente a página.

Os elementos `<article>` e `<section>` são os principais usados pelo algoritmo de SEO. Esses elementos são conhecidos por conter o corpo principal da página. Que uma página tenha mais de um o elemento `<article>` e / ou `<section>` é aceitável; todos eles são indexados. Dentro de cada elemento `<article>`, o motor procura elementos como `<hgroup>` ou `<h1>` para obter o principal tópico do elemento `<article>` para indexação relevante. No entanto, isso não significa que se um site não tem elementos `<article>` ou `<section>`, não será indexado e não será pesquisável. Refere-se apenas à qualidade da indexação que os motores de busca podem realizar para tornar o seu site mais pesquisável por usuários finais.

SEO é uma ótima técnica para entender ao projetar sites. Criar um site apenas e deixá-lo no escuro, com dificuldade para ser encontrado não serve a muitos propósitos. Ser encontrado é muito importante e, quando o site for encontrado, você não quer limitar o seu público. Acessibilidade também é muito importante.

## Otimizando para leitores de tela

Os leitores de tela contam com o esboço do documento para analisar a estrutura e apresentar as informações para o usuário. Os programas leitores de tela podem ler o texto na página e convertê-lo em áudio por meio de um algoritmo de conversão de texto em voz. Isso é útil para usuários que podem ter dificuldade para visualizar a página da web.

Conforme discutido anteriormente, a maneira como o documento é delineado em HTML5 tem mudado. Antes do HTML5, uma página era delineada usando apenas os elementos do cabeçalho (`<h1>` a `<h6>`). A posição relativa de cada elemento do cabeçalho em relação ao elemento anterior do cabeçalho na página criou a hierarquia. Os leitores de tela podem usar essas informações para apresentar um índice para os usuários. No entanto, o HTML5 introduziu elementos semânticos para criar novas seções. Isso significa que os elementos `<section>`, `<article>`, `<nav>` e `<aside>` definem novas seções. A introdução dos elementos semânticos altera a forma como o esboço do documento é criado.

### **aula1\exemplo\_13.html**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>TITULO DA PAGINA</title>
  </head>

  <body>
    <h1>Fruits and Vegetables</h1>
    <h2>Fruit</h2>
    <h3>Round Fruit</h3>
    <h3>Long Fruit</h3>
    <h2>Vegetables</h2>
    <h3>Green</h3>
    <h3>Colorful</h3>
  </body>
</html>
```

← → ⌂ ⓘ localhost/html5\_css3/capitulo\_1/aula\_1/exemplo\_13.html

## Fruits and Vegetables

**Fruit**

**Round Fruit**

**Long Fruit**

**Vegetables**

**Green**

**Colorful**

## **aula1\exemplo\_14.html**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>TITULO DA PAGINA</title>
  </head>

  <body>
    <section>
      <h1>Fruits and Vegetables</h1>
      <section>
        <h1>Fruit</h1>
        <section>
          <h1>Round Fruit</h1>
        </section>
        <section>
          <h1>Long Fruit</h1>
        </section>
      </section>
      <section>
        <h1>Vegetables</h1>
        <section>
          <h1>Green</h1>
        </section>
        <section>
          <h1>Colorful</h1>
        </section>
      </section>
    </section>
  </body>
</html>
```

localhost/html5\_css3/capitulo\_1/aula\_1/exemplo\_14.html

## **Fruits and Vegetables**

### **Fruit**

**Round Fruit**

**Long Fruit**

### **Vegetables**

**Green**

**Colorful**

Esse HTML produz a mesma saída mostrada anteriormente. A diferença é que agora cada elemento `<section>` cria uma nova seção de página em vez de usar elementos de cabeçalho para criar as seções. Os leitores de tela podem analisar os elementos semânticos para criar o esboço do documento e, eventualmente, pode fornecer uma experiência do usuário muito mais rica devido à forma de como o HTML5 permite que os designers de páginas da web façam o layout das páginas.

## Resumo

- HTML5 introduziu novos elementos semânticos para definir mais claramente as seções de uma página HTML. Esses elementos incluem `<section>`, `<article>`, `<nav>`, `<header>`, `<footer>`, `<aside>`, `<progress>`, `<mark>`, `<figure>` e `<figcaption>`.
- Os elementos em uma página HTML podem ter seu layout controlado quando são incluídos em estruturas como elementos `<div>` e/ou tabelas HTML.
- Os elementos semânticos HTML5 fornecem os mecanismos necessários para estruturar a página mais facilmente para acessibilidade por meio de leitores de tela.
- Os mecanismos de pesquisa aproveitam a semântica do HTML5, aproveitando o elemento `<article>` para determinar o propósito da página.

## Objective review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the “Answers” section at the end of this chapter.

1. Which of the following elements aren't introduced in HTML5?

- A. `<article>`
- B. `<footer>`
- C. `<hgroup>`
- D. `<input>`

A. Incorreto: O elemento `<article>` é um novo elemento semântico HTML5.

B. Incorreto: O elemento `<footer>` é um novo elemento semântico HTML5.

C. Incorreto: O elemento `<hgroup>` é um novo elemento semântico HTML5.

D. Correto: o elemento `<input>` não é novo no HTML5. No entanto, novos tipos de entrada foram introduzidos na especificação.

2. Which element(s) does the `<hgroup>` element contain?

- A. `<h1>` to `<h6>`
- B. `<header>`
- C. `<nav>`
- D. All of the above

A. Correto: O elemento `<hgroup>` deve conter qualquer um ou todos os elementos `<h1>` a `<h6>`.

B. Incorreto: O elemento `<header>` é usado separadamente para definir a seção do cabeçalho de um elemento `<article>` ou um elemento `<section>`.

C. Incorreto: O elemento `<nav>` é usado para definir uma estrutura de menu. Não seria incluído dentro de um elemento `<hgroup>`.

D. Incorreto: O elemento `<hgroup>` deve conter apenas elementos `<h1>` a `<h6>`.

3. Which HTML5 element would you use to organize content so that the page maximizes a search engine's algorithm?

- A. `<div id="CompanyNews">`
- B. `<header>Company News</header>`
- C. `<article>Company News</article>`
- D. All of the above

- A. Incorreto: O elemento <div> não fornece nenhum contexto adicional para um motor de pesquisa.
- B. Incorreto: O elemento <header> está contido dentro de um elemento <article> e irá ajudar, mas não é o elemento principal para fornecer otimização de mecanismo de pesquisa.
- C. Correto: O elemento <article> informa ao mecanismo de pesquisa que conteúdo específico nesta área é relevante para o que os usuários estão procurando. Este elemento fornece o melhor Motor de Otimização de Busca.
- D. Incorreto: O elemento <article> é usado especificamente para SEO.

4. Which HTML5 element should you use to create a more structured layout?

- A. <div>
- B. <p>
- C. <table>
- D. <form>

- A. Incorreto: Os elementos <div> fluem da esquerda para a direita e de cima para baixo e não fornecem qualquer estrutura.
- B. Incorreto: O elemento <p> denota um parágrafo. Este elemento não fornece qualquer estrutura de layout para a página.
- C. Correto: você usa o elemento <table> para fornecer um layout estruturado, usando seus elementos de linhas e colunas conforme necessário para criar o layout desejado.
- D. Incorreto: O elemento <form> é usado para denotar uma área onde o usuário pode enviar dados. O elemento <form> não fornece nenhum mecanismo para controlar o layout.

## Aula 2 - Escrever código para interagir com controles UI

Nesta aula, analisaremos como interagir com páginas da web no navegador usando código. Web browsers incluem um ambiente poderoso no qual você pode controlar o comportamento das páginas da web. E alguns novos elementos HTML5 fornecem interatividade aprimorada para usuários finais. Veremos como modificar o modelo de objeto do documento dinamicamente, usando JavaScript. Analisaremos como implementar vídeo e áudio em páginas da web e como controlá-los programaticamente. Finalmente, analisaremos como renderizar gráficos dinamicamente ou permitir que os usuários desenhem seus próprios gráficos.

Nesta aula veremos como:

- Adicionar ou modificar elementos HTML
- Implementar controles de mídia
- Implementar gráficos com HTML (canvas) e SVG

### Adicionando ou modificando elementos HTML

A capacidade de modificar um documento HTML em tempo de execução é muito poderosa. Em muitos casos, você deve modificar o layout de suas páginas da web em tempo de execução, dependendo do que seus usuários fazem. É aqui que você pode tirar proveito do poder do JavaScript. JavaScript fornece o kit de ferramentas de que você precisa para escrever um código que interaja com os elementos da página da web depois que eles já estiverem renderizados no navegador. Antes de começar a modificar a página da web, você precisa saber como acessar ou fazer referência aos elementos para que você possa manipulá-los.

### DOM - Document Object Model

O Document Object Model (DOM) é uma representação da estrutura da sua página HTML que permite que você interaja com ela. Uma página HTML é um hierarquia. O navegador produz um esboço com base na hierarquia HTML apresentada a ele e exibe isso no navegador para o usuário. Nos bastidores, sem o conhecimento do usuário, o navegador constrói um DOM. A interface de programação de aplicativos (API) do DOM é exposta como objetos com propriedades e métodos, permitindo que você escreva código JavaScript para interagir com elementos HTML renderizados na página. Essa noção é muito poderosa. Você pode adicionar novos elementos à página que nem mesmo existiam em sua página HTML original. Você pode modificar os elementos para mudar seu comportamento, layout, aparência e conteúdo. Teoricamente, embora isso raramente seja uma prática recomendada, você pode renderizar uma página HTML em branco para o navegador, construir a página inteira usando JavaScript e produzir exatamente os mesmos resultados. Ter esse poder sobre suas páginas da web é muito empolgante, mesmo depois de serem renderizados. Você começa selecionando itens no DOM para obter uma referência a eles.

### Selecionando itens no DOM

Para manipular o DOM, você precisa saber como acessá-lo e obter referências ao elementos que você deseja manipular. Ou seja, primeiro você precisa obter elementos do DOM para depois poder trabalhar com eles.

**Nota:** O Modelo de Objeto do Documento com uma árvore familiar, o DOM é essencialmente uma coleção de nós organizados em uma árvore. Todos os nós estão relacionados um para o outro. Eles são uma grande família feliz de filhos, irmãos, pais, avós, netos e assim por diante. Esta essência de uma árvore genealógica representa a hierarquia do DOM e é importante entender como você manipula o DOM por meio do código.

Existem algumas opções quando se trata de usar JavaScript para acessar o DOM. Você pode acessar elementos DOM por meio de um objeto global fornecido pelo navegador, denominado **document**, ou através dos próprios elementos depois de obter uma referência a um. A Tabela, a seguir, descreve os métodos nativos principais usados para selecionar elementos no DOM.

**TABLE 1-2** Methods available for selecting DOM elements

<b>Method</b>	<b>Usage description</b>
getElementById	Gets an individual element on the page by its unique id attribute value
getElementsByClassName	Gets all the elements that have the specified CSS class applied to them
getElementsByTagName	Gets all the elements of the page that have the specified tag name or element name
querySelector	Gets the first child element found that matches the provided CSS selector criteria
querySelectorAll	Gets all the child elements that match the provided CSS selector criteria

Na maioria das vezes, os métodos são diretos de usar.

## aula2\exemplo\_01.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>
    </head>
    <body>
        <div id="outerDiv">
            <p class='mainPara'>Main Paragraph</p>
            <ul>
                <li>First List Item</li>
                <li>Second List Item</li>
                <li>Third List Item</li>
                <li>Fourth List Item</li>
            </ul>
            <div id="innerDiv">
                <p class='subPara' id='P1'>Paragraph 1</p>
                <p class='subPara' id='P2'>Paragraph 2</p>
                <p class='subPara' id='P3'>Paragraph 3</p>
                <p class='subPara' id='P4'>Paragraph 4</p>
            </div>
            <table>
                <tr>
                    <td>Row 1 </td>
                </tr>
                <tr>
                    <td>Row 2 </td>
                </tr>
                <tr>
                    <td>Row 3 </td>
                </tr>
                <tr>
                    <td>Row 4 </td>
                </tr>
                <tr>
                    <td>Row 5 </td>
                </tr>
            </table>
            <input type="text"/><input type="submit" value="Submit"/>
        </div>
    </body>

<script>
    window.onload = function () {
        <!-- Pega todos o conteúdo do elemento com Id especificado -->
        var element = document.getElementById("outerDiv");
        alert(element.innerHTML);
    }
</script>

</html>
```

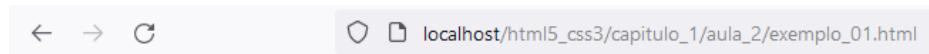
Esta página de exemplo é muito simples, mas serve ao propósito de demonstrar várias maneiras para acessar os elementos por meio de código.

⊕ localhost

```
<p class="mainPara">Main Paragraph</p>
<ul>
  <li>First List Item</li>
  <li>Second List Item</li>
  <li>Third List Item</li>
  <li>Fourth List Item</li>
</ul>
<div id="innerDiv">
  <p class="subPara" id="P1">Paragraph 1</p>
  <p class="subPara" id="P2">Paragraph 2</p>
  <p class="subPara" id="P3">Paragraph 3</p>
  <p class="subPara" id="P4">Paragraph 4</p>
</div>
<table>
  <tbody><tr>
    <td>Row 1 </td>
  </tr>
  <tr>
    <td>Row 2 </td>
  </tr>
  <tr>
    <td>Row 3 </td>
  </tr>
  <tr>
    <td>Row 4 </td>
  </tr>
  <tr>
    <td>Row 5 </td>
  </tr>
</tbody></table>
```

OK

O método de alerta JavaScript, que exibe uma caixa de mensagem, é usado aqui para mostrar se você realmente acessou o DOM com sucesso. O alerta não é tão útil no mundo real, mas é utilizado para fins de desenvolvimento. Ao executar a página, observe a caixa de mensagem do navegador com todo o conteúdo `innerHTML` do `<div>` que você selecionou do DOM com o seu código.



Main Paragraph

- First List Item
- Second List Item
- Third List Item
- Fourth List Item

Paragraph 1

Paragraph 2

Paragraph 3

Paragraph 4

Row 1

Row 2

Row 3

Row 4

Row 5

## Método getElementById

O método `getElementById` é ótimo quando você sabe o ID de um elemento específico na página com a qual deseja trabalhar. Quando você pode deseja fazer algo para todos os elementos de um determinado tipo (por exemplo, parágrafos), o método `getElementsByName` é mais apropriado.

### aula2\exemplo\_02.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>
  </head>
  <body>
    <div id="outerDiv">
      <p class='mainPara'>Main Paragraph</p>
      <ul>
        <li>First List Item</li>
        <li>Second List Item</li>
        <li>Third List Item</li>
        <li>Fourth List Item</li>
      </ul>
      <div id="innerDiv">
        <p class='subPara' id='P1'>Paragraph 1</p>
        <p class='subPara' id='P2'>Paragraph 2</p>
        <p class='subPara' id='P3'>Paragraph 3</p>
        <p class='subPara' id='P4'>Paragraph 4</p>
      </div>
      <table>
        <tr>
          <td>Row 1 </td>
        </tr>
        <tr>
          <td>Row 2 </td>
        </tr>
        <tr>
          <td>Row 3 </td>
        </tr>
        <tr>
          <td>Row 4 </td>
        </tr>
        <tr>
          <td>Row 5 </td>
        </tr>
      </table>
      <input type="text"/><input type="submit" value="Submit"/>
    </div>
  </body>
  <script>
    window.onload = function () {
      // Conta todos os elementos <p>
      var paragraphs = document.getElementsByName("p");
      alert("Número de parágrafos: " + paragraphs.length);
    }
  </script>
</html>
```

⊕ localhost

Número de parágrafos: 5

OK

## Método getElementTagName

Da mesma forma que você pode usar o método `getElementsByTagName` para obter todos elementos do mesmo tipo, você pode usar o método `getElementsByClassName` para obter todos elementos da mesma classe CSS. Isso é útil quando você tem muitos elementos com o mesmo estilo, mas talvez queira modificá-los em tempo de execução. Este método também retorna um NodeList.

### aula2\exemplo\_03.html

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>
    </head>

    <body>
        <div id="outerDiv">
            <p class='mainPara'>Main Paragraph</p>
            <ul>
                <li>First List Item</li>
                <li>Second List Item</li>
                <li>Third List Item</li>
                <li>Fourth List Item</li>
            </ul>
            <div id="innerDiv">
                <p class='subPara' id='P1'>Paragraph 1</p>
                <p class='subPara' id='P2'>Paragraph 2</p>
                <p class='subPara' id='P3'>Paragraph 3</p>
                <p class='subPara' id='P4'>Paragraph 4</p>
            </div>
            <table>
                <tr>
                    <td>Row 1 </td>
                </tr>
                <tr>
                    <td>Row 2 </td>
                </tr>
                <tr>
                    <td>Row 3 </td>
                </tr>
                <tr>
                    <td>Row 4 </td>
                </tr>
                <tr>
                    <td>Row 5 </td>
                </tr>
            </table>
        </div>
    </body>
</html>
```

```

</table>
<input type="text"/><input type="submit" value="Submit"/>
</div>
</body>

<script>
window.onload = function () {
    var paragraphs = document.getElementsByClassName("subPara");
    alert("Elementos com a classe subPara: " + paragraphs.length);
}
</script>
</html>

```

 localhost

Elementos com a classe subPara: 4

OK

## Métodos querySelector e querySelectorAll

Todos os métodos que você examinou até agora para encontrar elementos no DOM fornecem uma implementação para um propósito específico. Se você quiser um único elemento por seu ID exclusivo, use o método `getElementById`; se você quiser encontrar um elemento ou todos os elementos de um determinado Classe CSS, você usa o método `getElementsByClassName`.

Os métodos `querySelector` e `querySelectorAll` são muito mais flexíveis, permitem que você alcance muito do que você já feito com os outros métodos. Ambos os métodos recebem um parâmetro na forma de um seletor CSS. O método `querySelector` retorna o primeiro elemento encontrado que corresponde aos critérios do seletor passados a ele, enquanto o método `querySelectorAll` retorna todos os elementos que correspondem aos critérios do seletor passados. Os elementos ainda são retornados na forma de um objeto `NodeList`.

Ambos os métodos existem não apenas no próprio documento, mas também em cada elemento. Portanto, quando você tem uma referência a um elemento, você pode usar esses métodos para pesquisar seus filhos sem ter que percorrer todo o documento.

Para buscar todos os elementos `<p>` numa página, você poderia usar esta sintaxe:  
`document.querySelectorAll("p");`

Para buscar um elemento com um único ID, você poderia usar esta sintaxe:  
`document.querySelector("#outerDiv");`

## Dica para exames

jQuery é provavelmente a biblioteca mais popular disponível até hoje para simplificar e estender os principais recursos do JavaScript. Embora jQuery não seja uma tecnologia da Microsoft, é essencialmente um padrão da indústria e totalmente suportado pela Microsoft. O exame espera que você possa usar jQuery efetivamente no lugar de métodos seletores.

## Alterando o DOM

Ter acesso ao DOM por meio de JavaScript pode fornecer uma rica experiência ao usuário ao criar páginas web dinâmicas. Até agora, tudo que você fez foi obter referências aos elementos, o que não é particularmente útil por si só. O objetivo de recuperar elementos do DOM é permitir realizar algo com eles. Nesta seção, veremos como manipular o DOM usando código JavaScript para adicionar e remover itens.

Depois de ter uma referência a um elemento de container, você pode adicionar elementos filho a ele dinamicamente. Você pode remover elementos dele ou simplesmente ocultar elementos. Quando você remove um elemento do DOM, ele será excluído. Porém, se você quiser tornar algo invisível para o usuário, mas ser capaz de usá-lo novamente mais tarde, você pode simplesmente ocultá-lo usando o CSS apropriado em vez de removê-lo. Aqui está um exemplo:

### aula2\exemplo\_04.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>
  </head>
  <body>
    <div id="outerDiv">
      <p class='mainPara'>Main Paragraph</p>
      <ul>
        <li>First List Item</li>
        <li>Second List Item</li>
        <li>Third List Item</li>
        <li>Fourth List Item</li>
      </ul>
      <div id="innerDiv">
        <p class='subPara' id='P1'>Paragraph 1</p>
        <p class='subPara' id='P2'>Paragraph 2</p>
        <p class='subPara' id='P3'>Paragraph 3</p>
        <p class='subPara' id='P4'>Paragraph 4</p>
      </div>
      <table>
        <tr>
          <td>Row 1 </td>
        </tr>
        <tr>
          <td>Row 2 </td>
        </tr>
        <tr>
          <td>Row 3 </td>
        </tr>
        <tr>
          <td>Row 4 </td>
        </tr>
        <tr>
          <td>Row 5 </td>
        </tr>
      </table>
      <input type="text"/><input type="submit" value="Submit"/>
    </div>
```

```
</body>
<script>
window.onload = function () {
    var element = document.getElementById("innerDiv");
    alert(element.innerHTML);
    document.removeChild(element);
    var afterRemove = document.getElementById("innerDiv");
    alert(afterRemove);
}
</script>
</html>
```

⊕ localhost

```
<p class="subPara" id="P1">Paragraph 1</p>
<p class="subPara" id="P2">Paragraph 2</p>
<p class="subPara" id="P3">Paragraph 3</p>
<p class="subPara" id="P4">Paragraph 4</p>
```

OK

O primeiro alerta mostra corretamente a propriedade innerHTML do innerDiv, mas o código nunca atinge o segundo alerta porque o ID do elemento especificado não existe mais no documento.

## Adicionando elementos no DOM

### Método `document.createElement`

Esteja ciente dos vários métodos quando se trata de adicionar elementos e removê-los do DOM. O primeiro método a ser examinado é `document.createElement`. Você usa este método do objeto de documento para criar um novo elemento HTML. O método recebe um único parâmetro — o nome do elemento que você deseja criar. O código a seguir cria um novo Elemento `<article>` para usar em sua página:

```
var element = document.createElement("article");
element.innerText = "My new <article> element";
```

Este novo elemento `<article>` não é visível para ninguém; ele simplesmente existe no DOM para uso em sua página. Porém não é muito útil criar elementos se não podemos utilizá-los em seguida.

### Método `appendChild`

Há métodos disponíveis para obter seu novo elemento `<article>` na sua página. O primeiro desses métodos é `appendChild`. Você usa este método para adicionar um novo elemento HTML à coleção de elementos filhos pertencentes ao container de chamada. O nó é adicionado ao final da lista de filhos que o nó pai já contém.

O método `appendChild` existe no objeto do documento, bem como em outros elementos de container HTML. Retorna uma referência ao nó recém-adicionado. Este exemplo anexa um novo elemento `<article>` ao outerDiv:

## aula2\exemplo\_05.html

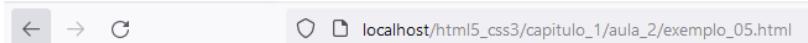
```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>
    </head>

    <body>
        <div id="outerDiv">
            <p class='mainPara'>Main Paragraph</p>
            <ul>
                <li>First List Item</li>
                <li>Second List Item</li>
                <li>Third List Item</li>
                <li>Fourth List Item</li>
            </ul>
            <div id="innerDiv">
                <p class='subPara' id='P1'>Paragraph 1</p>
                <p class='subPara' id='P2'>Paragraph 2</p>
                <p class='subPara' id='P3'>Paragraph 3</p>
                <p class='subPara' id='P4'>Paragraph 4</p>
            </div>
            <table>
                <tr>
                    <td>Row 1 </td>
                </tr>
                <tr>
                    <td>Row 2 </td>
                </tr>
                <tr>
                    <td>Row 3 </td>
                </tr>
                <tr>
                    <td>Row 4 </td>
                </tr>
                <tr>
                    <td>Row 5 </td>
                </tr>
            </table>
            <input type="text"/><input type="submit" value="Submit"/>
        </div>

        </body>

        <script>
            window.onload = function () {
                var outerDiv = document.getElementById("outerDiv");
                var element = document.createElement("article");
                element.innerText = "My new <article> element";
                outerDiv.appendChild(element);
            }
        </script>
    </html>
```



#### Main Paragraph

- First List Item
- Second List Item
- Third List Item
- Fourth List Item

Paragraph 1

Paragraph 2

Paragraph 3

Paragraph 4

Row 1

Row 2

Row 3

Row 4

Row 5

My new <article> element

O método **appendChild** retorna uma referência ao novo elemento anexado aos elementos filho. Esta é uma boa maneira de garantir que você sempre tem uma referência a um elemento para uso futuro, especialmente ao excluir elementos. Também permite simplificar ou reestruturar o código. O código a seguir produz o mesmo resultado:

#### aula2\exemplo\_06.html

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>
    </head>

    <body>
        <div id="outerDiv">
            <p class='mainPara'>Main Paragraph</p>
            <ul>
                <li>First List Item</li>
                <li>Second List Item</li>
                <li>Third List Item</li>
                <li>Fourth List Item</li>
            </ul>
            <div id="innerDiv">
                <p class='subPara' id='P1'>Paragraph 1</p>
                <p class='subPara' id='P2'>Paragraph 2</p>
                <p class='subPara' id='P3'>Paragraph 3</p>
                <p class='subPara' id='P4'>Paragraph 4</p>
            </div>
            <table>
                <tr>
                    <td>Row 1 </td>
                </tr>
                <tr>
                    <td>Row 2 </td>
                </tr>
                <tr>
```

```

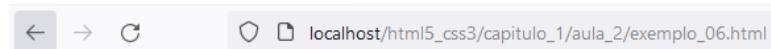
        <td>Row 3 </td>
    </tr>
    <tr>
        <td>Row 4 </td>
    </tr>
    <tr>
        <td>Row 5 </td>
    </tr>
</table>
<input type="text"/><input type="submit" value="Submit"/>
</div>

</body>

<script>
window.onload = function () {
    var element = document.getElementById("outerDiv").appendChild(document.
        createElement("article"));
    element.innerText = "My new <article> element";
}
</script>

</html>

```



Main Paragraph

- First List Item
- Second List Item
- Third List Item
- Fourth List Item

Paragraph 1

Paragraph 2

Paragraph 3

Paragraph 4

Row 1

Row 2

Row 3

Row 4

Row 5

Submit

My new <article> element

## Método insertBefore

Perceba que o elemento `<article>` foi colocado no final da página. O método `appendChild` sempre adiciona o novo elemento ao final do filho do elemento pai lista de nós. Para inserir o novo elemento `<article>` em algum lugar mais preciso, o método `insertBefore` poderia ser mais adequado. Este método leva dois parâmetros: o novo elemento em si, e o nó antes do qual você deseja anexar o novo elemento. Por exemplo, para inserir seu novo artigo antes do elemento `innerDiv`, você pode escrever o seguinte código:

### aula2\exemplo\_07.html

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>
    </head>

    <body>
        <div id="outerDiv">
            <p class='mainPara'>Main Paragraph</p>
            <ul>
                <li>First List Item</li>
                <li>Second List Item</li>
                <li>Third List Item</li>
                <li>Fourth List Item</li>
            </ul>
            <div id="innerDiv">
                <p class='subPara' id='P1'>Paragraph 1</p>
                <p class='subPara' id='P2'>Paragraph 2</p>
                <p class='subPara' id='P3'>Paragraph 3</p>
                <p class='subPara' id='P4'>Paragraph 4</p>
            </div>
            <table>
                <tr>
                    <td>Row 1 </td>
                </tr>
                <tr>
                    <td>Row 2 </td>
                </tr>
                <tr>
                    <td>Row 3 </td>
                </tr>
                <tr>
                    <td>Row 4 </td>
                </tr>
                <tr>
                    <td>Row 5 </td>
                </tr>
            </table>
            <input type="text"/><input type="submit" value="Submit"/>
        </div>

    </body>

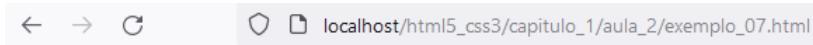
    <script>
        window.onload = function () {
```

```

var element = document.getElementById("outerDiv").insertBefore(
  document.createElement("article"),
  document.getElementById("innerDiv"));
element.innerText = "My new <article> element";
}
</script>

</html>

```



#### Main Paragraph

- First List Item
- Second List Item
- Third List Item
- Fourth List Item

My new <article> element

Paragraph 1

Paragraph 2

Paragraph 3

Paragraph 4

Row 1

Row 2

Row 3

Row 4

Row 5

Submit

Este exemplo usa o método `getElementById` para obter uma referência ao nó antes do qual você queira inserir seu elemento `<article>` no DOM.

Cada elemento ou nó tem as propriedades listadas na tabela 1-3.

**TABLE 1-3** Properties available on a DOM element

Property	Description
<code>childNodes</code>	A collection of all child nodes of the parent element.
<code>firstChild</code>	A reference to the very first child node in the list of child nodes of the parent node.
<code>lastChild</code>	A reference to the very last child node in the list of the child nodes of the parent node.
<code>hasChildNodes</code>	A useful property that returns <code>true</code> if the parent element has any child nodes at all. A good practice is to check this property before accessing other properties, such as <code>firstChild</code> or <code>lastChild</code> .

Para obter um exemplo dessas propriedades, você pode alterar o código anterior para inserir seu elemento `<article>` como o primeiro elemento no elemento `innerDiv`:

## aula2\exemplo\_08.html

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>
    </head>

    <body>
        <div id="outerDiv">
            <p class='mainPara'>Main Paragraph</p>
            <ul>
                <li>First List Item</li>
                <li>Second List Item</li>
                <li>Third List Item</li>
                <li>Fourth List Item</li>
            </ul>
            <div id="innerDiv">
                <p class='subPara' id='P1'>Paragraph 1</p>
                <p class='subPara' id='P2'>Paragraph 2</p>
                <p class='subPara' id='P3'>Paragraph 3</p>
                <p class='subPara' id='P4'>Paragraph 4</p>
            </div>
            <table>
                <tr>
                    <td>Row 1 </td>
                </tr>
                <tr>
                    <td>Row 2 </td>
                </tr>
                <tr>
                    <td>Row 3 </td>
                </tr>
                <tr>
                    <td>Row 4 </td>
                </tr>
                <tr>
                    <td>Row 5 </td>
                </tr>
            </table>
            <input type="text"/><input type="submit" value="Submit"/>
        </div>

        </body>

        <script>
            window.onload = function () {
                var inner = document.getElementById("innerDiv");
                var element = inner.insertBefore(document.createElement("article"), inner.firstChild);
                element.innerHTML = "My new <article> element";
            }
        </script>
    </html>
```

Main Paragraph

- First List Item
- Second List Item
- Third List Item
- Fourth List Item

My new <article> element

Paragraph 1

Paragraph 2

Paragraph 3

Paragraph 4

Row 1

Row 2

Row 3

Row 4

Row 5

## Removendo elementos no DOM

Assim como você pode adicionar novos elementos ao DOM por meio de código, também pode remover elementos do DOM usando código.

### Método removeChild()

O método `removeChild()` remove um nó filho do container de chamada. Este método existe no objeto de documento, bem como em outros elementos de container HTML. O `removeChild()` método retorna uma referência ao nó removido. Isso é especialmente útil se você planeja retornar esse nó para o DOM - talvez em resposta a alguma outra interação do usuário com a página. Lembre-se, no entanto, que se você não mantiver a referência retornada ao nó removido, você não tem como adicionar o elemento de volta sem recriá-lo completamente. O exemplo a seguir remove o primeiro elemento `<p>` de seu elemento `innerDiv`:

### aula2\exemplo\_09.html

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>
    </head>

    <body>
        <div id="outerDiv">
            <p class='mainPara'>Main Paragraph</p>
            <ul>
                <li>First List Item</li>
                <li>Second List Item</li>
                <li>Third List Item</li>
                <li>Fourth List Item</li>
            </ul>
        </div>
    </body>
</html>
```

```

<div id="innerDiv">
    <p class='subPara' id='P1'>Paragraph 1</p>
    <p class='subPara' id='P2'>Paragraph 2</p>
    <p class='subPara' id='P3'>Paragraph 3</p>
    <p class='subPara' id='P4'>Paragraph 4</p>
</div>
<table>
    <tr>
        <td>Row 1 </td>
    </tr>
    <tr>
        <td>Row 2 </td>
    </tr>
    <tr>
        <td>Row 3 </td>
    </tr>
    <tr>
        <td>Row 4 </td>
    </tr>
    <tr>
        <td>Row 5 </td>
    </tr>
</table>
<input type="text"/><input type="submit" value="Submit"/>
</div>

</body>

<script>
    window.onload = function () {
        var innerDiv = document.getElementById("innerDiv");
        var p = innerDiv.removeChild(document.getElementById("P1"));
    }
</script>

</html>

```

The screenshot shows a browser window with the URL `localhost/html5_css3/capitulo_1/aula_2/exemplo_09.html`. The page content is as follows:

- Main Paragraph**
- First List Item
- Second List Item
- Third List Item
- Fourth List Item
- Paragraph 2
- Paragraph 3
- Paragraph 4
- Row 1
- Row 2
- Row 3
- Row 4
- Row 5

A text input field and a submit button are at the bottom.

## Método remove

O método **remove()** remove o elemento especificado do DOM.

### **aula2\exemplo\_10.html**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>
  </head>
  <body>
    <div id="outerDiv">
      <p class='mainPara'>Main Paragraph</p>
      <ul>
        <li>First List Item</li>
        <li>Second List Item</li>
        <li>Third List Item</li>
        <li>Fourth List Item</li>
      </ul>
      <div id="innerDiv">
        <p class='subPara' id='P1'>Paragraph 1</p>
        <p class='subPara' id='P2'>Paragraph 2</p>
        <p class='subPara' id='P3'>Paragraph 3</p>
        <p class='subPara' id='P4'>Paragraph 4</p>
      </div>
      <table>
        <tr>
          <td>Row 1 </td>
        </tr>
        <tr>
          <td>Row 2 </td>
        </tr>
        <tr>
          <td>Row 3 </td>
        </tr>
        <tr>
          <td>Row 4 </td>
        </tr>
        <tr>
          <td>Row 5 </td>
        </tr>
      </table>
      <input type="text"/><input type="submit" value="Submit"/>
    </div>
  </body>
  <script>
    window.onload = function () {
      var innerDiv = document.getElementById("innerDiv");
      innerDiv.remove();
    }
  </script>
</html>
```



### Main Paragraph

- First List Item
- Second List Item
- Third List Item
- Fourth List Item

Row 1

Row 2

Row 3

Row 4

Row 5

Submit

## Alterando o conteúdo do DOM

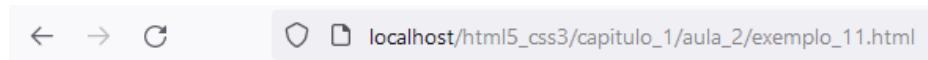
### aula2\exemplo\_11.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>
  </head>
  <body>
    <div id="outerDiv">
      <p class='mainPara'>Main Paragraph</p>
      <ul>
        <li>First List Item</li>
        <li>Second List Item</li>
        <li>Third List Item</li>
        <li>Fourth List Item</li>
      </ul>
      <div id="innerDiv">
        <p class='subPara' id='P1'>Paragraph 1</p>
        <p class='subPara' id='P2'>Paragraph 2</p>
        <p class='subPara' id='P3'>Paragraph 3</p>
        <p class='subPara' id='P4'>Paragraph 4</p>
      </div>
      <table>
        <tr>
          <td>Row 1 </td>
        </tr>
        <tr>
          <td>Row 2 </td>
        </tr>
        <tr>
          <td>Row 3 </td>
        </tr>
        <tr>
          <td>Row 4 </td>
        </tr>
        <tr>
          <td>Row 5 </td>
        </tr>
```

```

        </table>
        <input type="text"/><input type="submit" value="Submit"/>
    </div>
</body>
<script>
    window.onload = function () {
        var innerDiv = document.getElementById("innerDiv");
        var newDiv = document.createElement("div");
        for (var i = 0; i < innerDiv.childNodes.length; i++) {
            var anchor = newDiv.appendChild(document.createElement("a"));
            anchor.setAttribute("href", "http://www.bing.ca");
            anchor.textContent = innerDiv.childNodes[i].textContent;
            newDiv.appendChild(document.createElement("br"));
        }
        innerDiv.innerHTML = newDiv.innerHTML;
    }
</script>
</html>

```



## Main Paragraph

- First List Item
- Second List Item
- Third List Item
- Fourth List Item

[Paragraph 1](#)

[Paragraph 2](#)

[Paragraph 3](#)

[Paragraph 4](#)

Row 1

Row 2

Row 3

Row 4

Row 5

## Implementando controles de mídia

Incorporar elementos multimídia em páginas da web não é um conceito novo. Esta capacidade existe há muito tempo e apresenta desafios em diversas situações. Um desafio chave muitas vezes tem sido a dependência de um objeto de terceiros integrado com o navegador para processar os meios de comunicação. Nesta seção, você verá dois novos elementos adicionados à especificação HTML5 que trabalham com multimídia nativamente no navegador da web: os elementos `<video>` e `<audio>`.

### Usando o elemento `<video>`

Incorporar vídeo em uma página da web se tornou muito popular e muitos sites agora incluem um elemento de vídeo em seu design. HTML5 tornou a inclusão de vídeo em suas páginas da web muito mais fácil do que antes.

**TABLE 1-4** Attributes available on the `<video>` element

Attribute	Description
<code>src</code>	This attribute specifies the video to play. It can be a local resource within your own website or something exposed through a public URL on the Internet.
<code>autoplay</code>	This attribute tells the browser to start playing the video as soon as it loads. If this attribute is omitted, the video plays only when told to through player controls or JavaScript.
<code>controls</code>	This attribute tells the browser to include its built-in video controls, such as play and pause. If this is omitted, the user has no visible way to play the content. You would use <code>autoplay</code> or provide some other mechanism through JavaScript to play the video.
<code>height/width</code>	These attributes control the amount of space the video will occupy on the page. Omitting these causes the video to display in its native size.
<code>loop</code>	This attribute tells the browser to continuously play the video after it completes. If this attribute is omitted, the video stops after it plays through completely.
<code>poster</code>	This attribute specifies an image to show in the place allocated to the video until the user starts to play the video. Use this when you're not using <code>autoplay</code> . It's very useful for providing a professional image or artwork to represent the video. If it's omitted, the poster appears in the first frame of the video.

- Incorporar um vídeo na página é tão simples quanto adicionar a seguinte marcação:

## aula2\exemplo\_12.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>
  </head>
  <body>

    <h1>Vídeo</h1>
    <video id="sampleVideo" autoplay muted controls height="400" width="600"
poster="http://blog.teamtreehouse.com/wp-content/uploads/2015/05/InternetSlowdown_Day.gif">
      <source src="sample_video.ogv" type="video/ogg"/>
      <source src="sample_video.mp4" type="audio/mp4"/>
      <object>
        <p>Video is not supported by this browser.</p>
      </object>
    </video>
  </body>
</html>
```

Este exemplo removeu o atributo src do elemento `<video>` e em vez disso, adicionou os elementos filhos `<source>`. O elemento `<video>` oferece suporte a vários elementos `<source>`, então você pode incluir um para cada tipo de vídeo. Um navegador passa pelos elementos `<source>` de cima para baixo e reproduz o primeiro que suporta.

localhost/html5\_css3/capitulo\_1/aula\_2/exemplo\_12.html

### Vídeo



### Vídeo



**TABLE 1-5** Methods and properties on the `<video>` object

Method/property	Description
<code>play()</code>	Plays the video from its current position.
<code>pause()</code>	Pauses the video at its current position.
<code>volume</code>	Allows the user to control the volume of the video.
<code>currentTime</code>	Represents the current position of the video. Increase or decrease this value to move forward or backward in the video.

## Usando o elemento `<audio>`

O elemento `<audio>` é essencialmente idêntico ao elemento `<video>`. Possui todos os mesmos atributos e os mesmos métodos. A única diferença real é como ele é exibido no navegador. Como nenhum vídeo está disponível para exibição, o elemento `<audio>` não ocupa espaço na tela. No entanto, você pode mostrar os controles padrão ou pode escolher novamente não mostrar os controles padrão e criar seu próprio mecanismo para controlar o áudio.

### aula2\exemplo\_13.html

```
<!DOCTYPE html>
<html>

  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>
  </head>

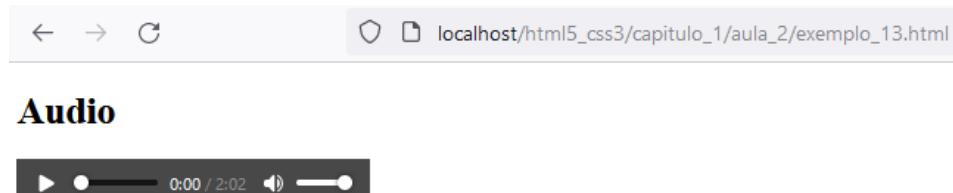
  <body>

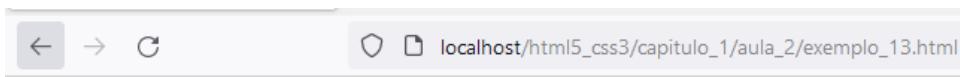
    <h1>Audio</h1>

    <audio controls>
      <source src="sample.mp3" type="audio/mp3"/>
      <source src="sample.ogg" type="audio/ogg"/>
      <p>Your browser does not support HTML5 audio.</p>
    </audio>

  </body>

</html>
```





## Audio



## Implementando gráficos com HTML5 (Canvas) e SVG

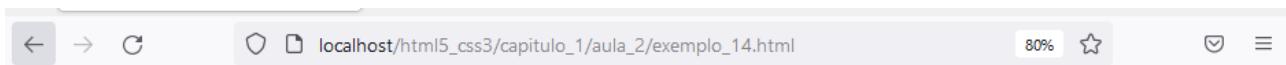
### Canvas

HTML5 fornece um novo mecanismo para trabalhar com gráficos em suas páginas da web. O HTML5 introduz o elemento <canvas>, que fornece uma tela em branco no qual você pode desenhar dinamicamente. Você pode desenhar linhas, texto e imagens na tela e manipulá-los com JavaScript. Adicionar uma tela à sua página é tão simples quanto declarar uma tela no HTML. O elemento <canvas> é semelhante ao elemento <div>. No entanto, é um container para gráficos em oposição a elementos baseados em texto. Aqui está a marcação para um elemento <canvas>:

O HTML é muito simples. Você simplesmente precisa definir um <canvas> e especificar um tamanho. Além disso, se o navegador do usuário não suportar o elemento <canvas>, você pode colocar um texto substituto dentro do elemento <canvas> a ser exibido em seu lugar. Quando você executa este HTML no navegador, você não deve notar absolutamente nada! Isso ocorre porque - assim como com um elemento <div> ou qualquer outro container - o elemento <canvas> não tem visibilidade padrão; em outras palavras, é visível, mas é branco sem bordas e, portanto, é invisível em uma página HTML em branco. O próximo exemplo adiciona um estilo simples ao seu elemento <canvas> para que você possa ver suas bordas:

### aula2\exemplo\_14.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>
    <style>
      canvas{
        border: 1px solid black;
      }
    </style>
  </head>
  <body>
    <h1>Canvas</h1>
    <canvas id="drawingSurface" width="600" height="400">Your browser does not support HTML5.</canvas>
  </body>
  <script>
    window.onload = function () {
    }
  </script>
</html>
```



## Canvas

Uma tela em branco não nada empolgante. Mas agora que você tem uma tela básica instalada e em execução, pode trabalhar com todos os vários métodos para criar gráficos na tela. Você deve criar um evento onload para sua janela encapsular seu código e fazer com que os gráficos sejam renderizados quando a página for carregada.

Para desenhar na tela, você precisa entender o sistema de coordenadas que a tela usa. A tela fornece um sistema de coordenadas fixo (x, y) no qual o canto superior esquerdo da tela é (0,0). Nesse caso, o canto inferior esquerdo da tela é (0,400), o canto superior direito é (600,0) e o canto inferior direito é (600,400). No entanto, a posição da tela na janela do navegador é irrelevante para os métodos de desenho usados para desenhar na tela. As coordenadas para desenhar na tela são sempre com base nas coordenadas dentro da própria tela, onde o pixel superior esquerdo é (0,0).

**TABLE 1-6** Methods for drawing lines

Method	Description
<i>beginPath</i>	Resets/begins a new drawing path
<i>moveTo</i>	Moves the context to the point set in the <i>beginPath</i> method
<i>lineTo</i>	Sets the destination end point for the line
<i>stroke</i>	Strokes the line, which makes the line visible

## aula2\exemplo\_15.html

```
<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>
        <style>
            canvas{
                border: 1px solid black;
            }
        </style>
    </head>

    <body>
        <h1>Canvas</h1>
        <canvas id="drawingSurface" width="600" height="400">Your browser does not support HTML5.</canvas>
    </body>

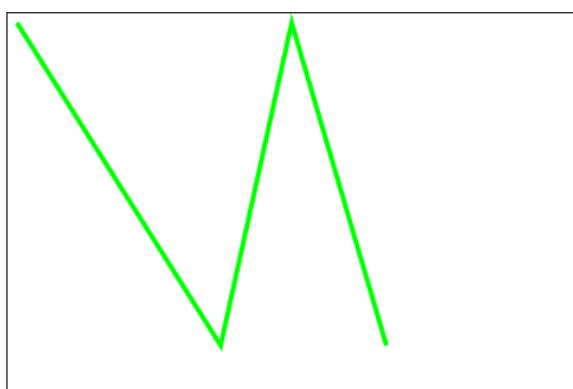
    <script>
        window.onload = function () {
            var drawingSurface = document.getElementById("drawingSurface");
            var ctxt = drawingSurface.getContext("2d");

            ctxt.lineWidth = 5;
            ctxt.strokeStyle = '#0f0';

            ctxt.beginPath();
            ctxt.moveTo(10, 10);
            ctxt.lineTo(225, 350);
            ctxt.lineTo(300, 10);
            ctxt.lineTo(400, 350);
            ctxt.stroke();
        }
    </script>
</html>
```

localhost/htm5\_css3/capitulo\_1/aula\_2/exemplo\_15.html

### Canvas



## Desenho de curvas

**TABLE 1-7** Methods for drawing curves

Method	Description
<i>arc</i>	A standard arc based on a starting and ending angle and a defined radius
<i>quadraticCurveTo</i>	A more complex arc that allows you to control the steepness of the curve
<i>bezierCurveTo</i>	Another complex arc that you can skew

**TABLE 1-8** Parameters required to draw an arc

Parameter	Description
<i>X, Y</i>	The first two parameters are the X and Y coordinates for the center of the circle.
<i>radius</i>	The third parameter is the radius. This is the length of the distance from the center point of the circle to the curve.
<i>startAngle, endAngle</i>	The fourth and fifth parameters specify the starting and ending angles of the arc to be drawn. This is measured in radians, not in degrees.
<i>counterclockwise</i>	The final parameter specifies the drawing direction of the arc.

### aula2\exemplo\_16.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>
    <style>
      canvas{
        border: 1px solid black;
      }
    </style>
  </head>
  <body>
    <h1>Canvas</h1>
    <canvas id="drawingSurface" width="600" height="400">Your browser does not support HTML5.</canvas>
  </body>

  <script>
    window.onload = function () {
      var drawingSurface = document.getElementById("drawingSurface");
      var ctxt = drawingSurface.getContext("2d");

      ctxt.lineWidth = 5;
      ctxt.strokeStyle = '#0f0';

      ctxt.beginPath();
      ctxt.arc(150,100,75,0,2 * Math.PI, false);
    }
  </script>

```

```

        ctxt.lineWidth = 25;
        ctxt.strokeStyle = '#0f0';
        ctxt.stroke();

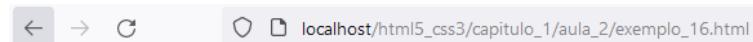
        ctxt.beginPath();
        ctxt.arc(450, 100, 75, 1.5 * Math.PI, 2 * Math.PI, false);
        ctxt.lineWidth = 25;
        ctxt.strokeStyle = 'blue';
        ctxt.stroke();

        ctxt.beginPath();
        ctxt.arc(150, 300, 75, 1 * Math.PI, 1.5 * Math.PI, false);
        ctxt.lineWidth = 25;
        ctxt.strokeStyle = '#0ff';
        ctxt.stroke();

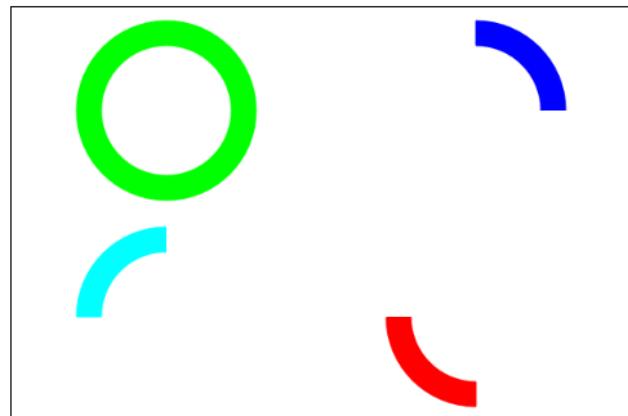
        ctxt.beginPath();
        ctxt.arc(450, 300, 75, .5 * Math.PI, 1 * Math.PI, false);
        ctxt.lineWidth = 25;
        ctxt.strokeStyle = '#f00';
        ctxt.stroke();
    }
</script>

</html>

```



## Canvas



## Curvas quadráticas

**TABLE 1-9** Parameters required for the *quadraticCurveTo* method

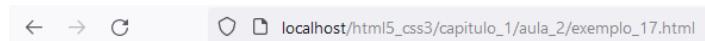
Parameter	Description
<i>controlX, controlY</i>	These parameters define the control point, relative to the top left of the canvas, that is used to "stretch" the curve away from the line formed by the start and end points.
<i>endX, endY</i>	This is the point where the curve should end.

## aula2\exemplo\_17.html

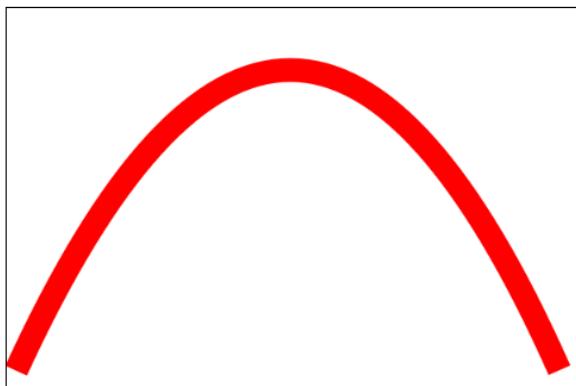
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>
    <style>
      canvas{
        border: 1px solid black;
      }
    </style>
  </head>
  <body>
    <h1>Canvas</h1>
    <canvas id="drawingSurface" width="600" height="400">Your browser does not support HTML5.</canvas>
  </body>
  <script>
    window.onload = function () {
      var drawingSurface = document.getElementById("drawingSurface");
      var ctxt = drawingSurface.getContext("2d");

      ctxt.lineWidth = 5;
      ctxt.strokeStyle = '#0f0';

      ctxt.beginPath();
      ctxt.moveTo(10,380);
      ctxt.quadraticCurveTo(300,-250,580,380);
      ctxt.lineWidth = 25;
      ctxt.strokeStyle = '#f00';
      ctxt.stroke();
    }
  </script>
</html>
```



**Canvas**



## Método bezierCurveTo()

O método `bezierCurveTo` segue uma chamada de método `moveTo` da mesma maneira que o método `quadraticCurveTo`. Você precisa passar três conjuntos de coordenadas para o Método `bezierCurveTo`, conforme listado na Tabela 1-10:

**TABLE 1-10** Parameters required for the `bezierCurveTo` method

Parameter	Description
<code>controlX, controlY</code>	The first two parameters specify the first control point that is used to stretch out the curve.
<code>Control2X, control2Y</code>	The second two parameters specify the second control point that is used to stretch out the curve.
<code>endX, endY</code>	The final two parameters specify the end point for the curve.

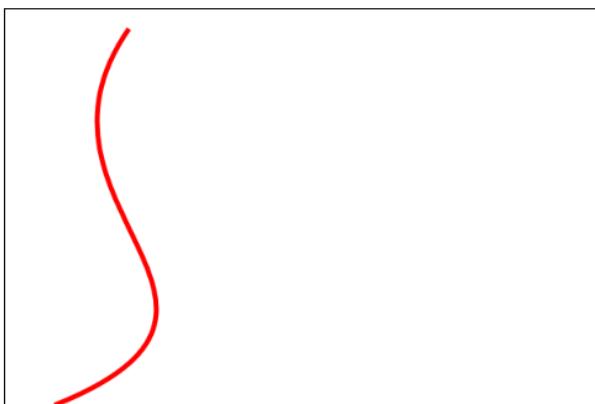
### aula2\exemplo\_18.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>
        <style>
            canvas{
                border: 1px solid black;
            }
        </style>
    </head>
    <body>
        <h1>Canvas</h1>
        <canvas id="drawingSurface" width="600" height="400">Your browser does not support HTML5.</canvas>
    </body>

    <script>
        window.onload = function () {
            var drawingSurface = document.getElementById("drawingSurface");
            var ctxt = drawingSurface.getContext("2d");

            ctxt.beginPath();
            ctxt.moveTo(125, 20);
            ctxt.bezierCurveTo(0, 200, 300, 300, 50, 400);
            ctxt.lineWidth = 5;
            ctxt.strokeStyle = '#f00';
            ctxt.stroke();
        }
    </script>
</html>
```

## Canvas



## Gráficos vetoriais escaláveis (SVG)

Scalable Vector Graphics (SVG) é uma linguagem baseada em XML para a criação de gráficos. É implementado usando tags definidas pelo namespace SVG XML e incorporado em documentos HTML5 dentro de elementos <svg> de abertura e fechamento.

Os objetos SVG não perdem a qualidade conforme os usuários aumentam ou diminuem o zoom. Você pode acessar objetos SVG por meio do DOM e - semelhante aos elementos HTML - os elementos SVG oferecem suporte a atributos, estilos, e manipuladores de eventos.

### aula2\exemplo\_19.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Test Web Page</title>
    <script language="javascript">
      function Red(evt) {
        var circle = evt.target;
        circle.setAttribute("style", "fill: red");
      }
      function Green(evt) {
        var circle = evt.target;
        circle.setAttribute("style", "fill: green");
      }
    </script>
  </head>
  <body>
    <svg>
      <circle id="Circle" cx="50" cy="50" r="50" fill="green" onmouseover="Red(evt)"
        onmouseout="Green(evt)"/>
    </svg>
  </body>
</html>
```



- Passando o mouse sobre o elemento:



## Dicas para exames

Em alguns casos, usar gráficos SVG é mais simples do que usar o elemento <canvas>. Você pode criar imagens SVG declarativamente diretamente no HTML em si. No entanto, conforme se aumenta o número de objetos em uma renderização SVG, o desempenho pode se tornar uma preocupação. Nos casos em que o desempenho é uma consideração, usar o elemento <canvas> é preferível.

O SVG também oferece suporte à renderização de gráficos existentes na forma de arquivos de imagem externos.

## Resumo

- JavaScript é uma ferramenta poderosa que permite aos desenvolvedores manipular o DOM programaticamente no navegador.
- HTML5 oferece suporte a controles de rich media para incorporar vídeo usando o elemento <video> e áudio usando o elemento <audio>.
- O elemento <video> oferece suporte a vários formatos de mídia usando o elemento <source>.
- Os elementos HTML5 <canvas> e <svg> oferecem suporte a uma API rica para criar gráficos complexos no navegador.
- Os motores gráficos <canvas> e <svg> podem desenhar texto, linhas, formas, fontes, preenchimentos, e gradientes.
- O elemento <canvas> é desenhado via JavaScript, obtendo uma referência ao contexto.
- O elemento <svg> renderiza gráficos usando uma sintaxe declarativa.

## Objective Review

1. Which of the following JavaScript methods can't be used to select an element in the DOM?
- A. getElementById
  - B. querySelector
  - C. getElementByClassName
  - D. queryAll**
- A. Incorreto: o método getElementById recupera um elemento por seu id único.  
B. Incorreto: o método querySelector recupera um único elemento que corresponde ao seletor especificado.  
C. Incorreto: O método getElementsByClassName recupera todos os elementos que têm a classe CSS especificada atribuída a eles.  
D. Correto: o método queryAll não está disponível para pesquisar o DOM.
2. Which line of JavaScript successfully retrieves only the image element with the ID myDog from the following HTML? Choose all that apply.
- ```
<form>
  <div id="main" class="mainStyle">
    <p id="dogs">
      This is a web page about dogs. Here is my dog picture:
      
      Here is a picture of my friend's dog:
      
    </p>
  </div>
</form>
```
- A. document.getElementById("myDog");**
  - B. <p>.getFirstChild("img");
  - C. document.getElementById("dogs").querySelector ("thumb");
  - D. document.querySelectorAll("thumb");
- A. Correto: document.getElementById ("myDog"); recupera apenas a única imagem com o ID myDog.  
B. Incorreto: <p> .getFirstChild ("img"); não é uma sintaxe válida.  
C. Incorreto: document.getElementById ("dogs") .querySelector ("thumb"); falha porque a página não possui nenhum elemento denominado "dogs".  
D. Incorreto: document.querySelector.querySelectorAll ("thumb"); está incorreto porque ele retorna todos os elementos da página com o nome de classe especificado.
3. To hide an element in the DOM and still be able to add it back later, what should you do?
- A. Nothing, because the DOM is always available in a static form.
  - B. Keep a reference to the removed node to be able to add it back.**
  - C. Call the document.restoreNodes method.
  - D. You can't add an element back after it's removed.
- A. Incorreto: O DOM é dinâmico e muda fisicamente quando os itens são removidos a partir dele.  
B. Correto: quando um elemento é removido e necessário novamente, uma referência ao nó removido que deve ser mantido para poder adicioná-lo de volta.  
C. Incorreto: Não existe tal método.

D. Incorreto: Um nó pode ser adicionado de volta ao DOM se uma referência foi mantida. Isto é realizado usando os vários métodos disponíveis para inserir nós no DOM.

4. When implementing the HTML5 video element, how do you ensure that the rendering of the element can function in different browsers?

A. You need to do nothing, because HTML5 is now a standard specification.

B. Specify all the source video types in the src attribute of the video element.

C. **Include the <source> element for each video type so that each browser can play the version that it supports.**

D. Include the <object> element for each video type so that the browser can play the version that it supports.

A. Incorreto: navegadores diferentes suportam formatos de mídia diferentes. O elemento <source> é usado para disponibilizar as especificações nos vários formatos.

B. Incorreto: O atributo src permite que você especifique apenas uma fonte de vídeo. Não funciona em vários navegadores, a menos que o formato de vídeo seja compatível com os navegadores.

C. Correto: Os elementos <source> especificam vários formatos de vídeo para que o navegador possa escolher o correto.

D. Incorreto: O elemento <object> é compatível para fornecer um mecanismo de fallback no evento no qual o navegador não suporta o elemento HTML5 <video>.

5. When drawing on the HTML5 <canvas> element, what method is used on the context to begin drawing at a new point?

A. moveTo

B. lineAt

C. **beginPath**

D. stroke

A. Incorreto: O método moveTo move o contexto atual para um novo ponto, mas não começa um desenho.

B. Incorreto: O método lineAt desenha uma linha dentro do contexto posicional atual.

C. Correto: O método beginPath diz ao contexto para iniciar um novo desenho a partir de seu ponto atual.

D. Incorreto: O método stroke informa ao contexto para desenhar os gráficos que são aplicado ao contexto.

6. When performance is critical for an HTML5 graphics application, what should you use?

A. <canvas> using a declarative syntax to create the graphics

B. <svg> using a declarative syntax to create the graphics

C. **<canvas> using JavaScript to create the graphics**

D. <svg> and <canvas> combination to leverage the best performance of both

A. Incorreto: O elemento <canvas> não suporta nenhum elemento declarativo.

B. Incorreto: O elemento <svg> tem desempenho pior do que o elemento <canvas> quando muita atualização gráfica é necessária.

C. Correto: O elemento <canvas> oferece desempenho superior em comparação com o elemento <svg>.

D. Incorreto: Os elementos <canvas> e <svg> combinados não fornecem melhores atuações. O elemento <canvas> oferece desempenho superior quando os gráficos exigem muita atualização.

## Aula 3 - Aplicar estilos em elementos HTML programaticamente

Nesta aula veremos como:

- Alterar a localização de um elemento
- Aplicar uma transformação
- Mostrar e ocultar elementos

### Alterando a posição de um elemento

Usando os métodos para recuperar um elemento do DOM em JavaScript é possível aplicar estilos dinamicamente através de código que pode alterar a posição do elemento na página. Algumas opções determinam como os elementos HTML são posicionados em uma página da Web.

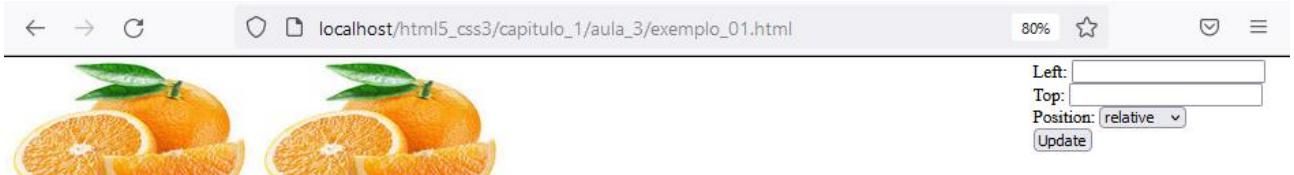
Por padrão, todos os elementos HTML fluem estaticamente da esquerda para a direita na mesma ordem que eles são declarados na página HTML. No entanto, o CSS fornece um mecanismo para especificar algumas opções avançadas de posição do elemento.

Você pode posicionar elementos usando posicionamento absoluto ou posicionamento relativo. Com posicionamento absoluto, o elemento é colocado no local exato especificado, em relação às fronteiras de seu container. No entanto, com o posicionamento relativo, o elemento é posicionado relativamente às coordenadas imediatas do vizinho imediatamente anterior (irmão esquerdo).

Ao usar posicionamento absoluto ou relativo, as configurações padrão de borda ou margem são ignoradas.

## aula\_3\exemplo\_01.html

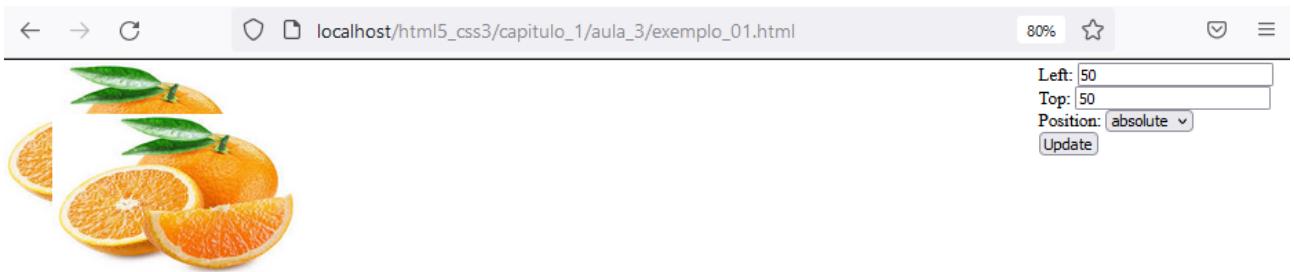
```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="utf-8"/>
        <title></title>
        <style>
            html, body {
                height: 100%;
                width: 100%;
                margin: 0px;
            }
            img {
                height: 150px;
                width: 225px;
            }
        </style>
        <script>
            window.onload = function () {
                var top = document.getElementById("topText");
                var left = document.getElementById("leftText");
                var pos = document.getElementById("positioning");
                document.getElementById("btnPosition").onclick = function () {
                    var img = document.getElementById("orange2");
                    img.style.position = pos.value;
                    img.style.left = left.value + "px";
                    img.style.top = top.value + "px";
                }
            }
        </script>
    </head>
    <body>
        <table style="width: 100%; height: 100%; border: 1px solid black;">
            <tr>
                <td style="vertical-align: top; width: 80%">
                    
                    
                </td>
                <td style="vertical-align: top;">Left:
                    <input type="text" id="leftText"/><br/>
                    Top:
                    <input type="text" id="topText"/><br/>
                    Position:
                    <select id="positioning">
                        <option>relative</option>
                        <option>absolute</option>
                    </select><br/>
                    <input type="button" id="btnPosition" value="Update"/>
                </td>
            </tr>
        </table>
    </body>
</html>
```



Quando o topo e a esquerda estiverem definidos em 50px e o posicionamento for relativo, teremos:



Mantendo os valores iguais, mas mudando o posicionamento para absoluto, teremos:



## Aplicando uma transformação

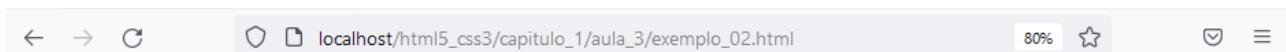
Aplicar transformações é uma maneira de alterar um objeto na página web. Permite que você mude a aparência de um elemento. Você pode fazer um elemento maior ou menor, rodá-lo, e assim por diante. Existem alguns métodos de transformação disponíveis. Para adicionar uma transformação a um elemento, você deve declará-lo no CSS para o elemento adicionando a propriedade `transform`.

### Método rotate

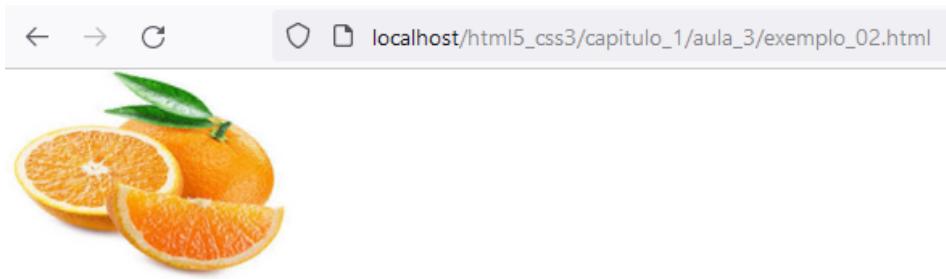
O método `rotate` permite girar um objeto por um número especificado de Graus. O método aceita um único parâmetro que especifica o número de graus.

#### aula\_3\exemplo\_02.html

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="utf-8"/>
        <title></title>
        <style>
            #orange1 {
                height: 150px;
                width: 225px;
            }
            .trans {
                transform: rotate(15deg);
            }
        </style>
        <script>
            window.onload = function () {
                document.getElementById("orange1").onclick = function () {
                    this.classList.add("trans");
                }
            }
        </script>
    </head>
    <body>
        
    </body>
</html>
```



Ao clicar na imagem:

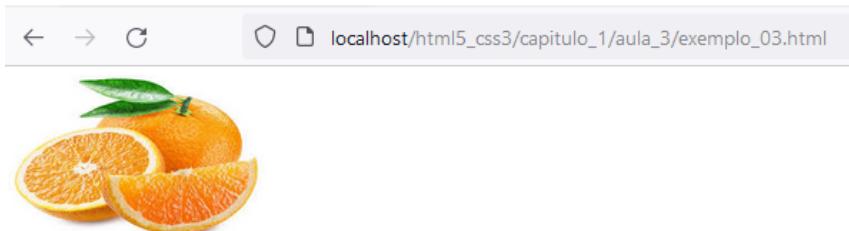


## Método translate

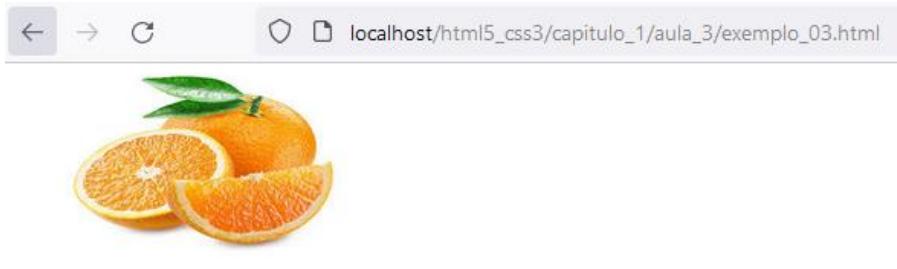
O método de tradução permite mover um elemento HTML alterando sua posição relativa X e Y na página.

### aula\_3\exemplo\_03.html

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="utf-8"/>
        <title></title>
        <style>
            #orange1 {
                height: 150px;
                width: 225px;
            }
            .trans {
                transform: translate(50px,0px);
            }
        </style>
        <script>
            window.onload = function () {
                document.getElementById("orange1").onclick = function () {
                    this.classList.add("trans");
                }
            }
        </script>
    </head>
    <body>
        
    </body>
</html>
```



Ao clicar na imagem:

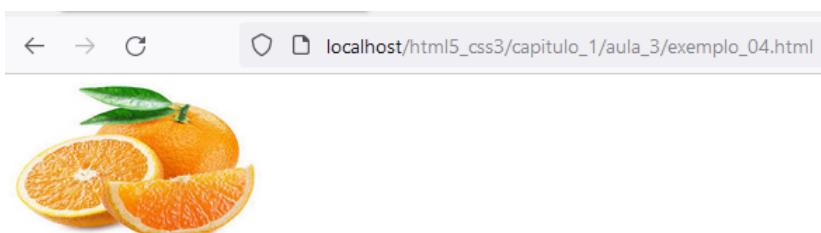


## Método skew

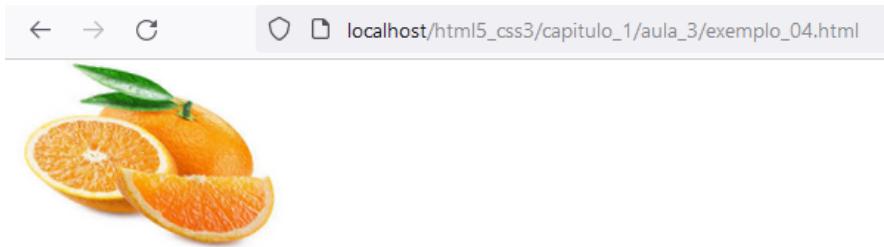
Você pode distorcer um elemento HTML usando o método skew da propriedade transform. Skew inclina o objeto para que ele não seja paralelo ao eixo vertical ou horizontal.

### aula\_3\exemplo\_04.html

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="utf-8"/>
        <title></title>
        <style>
            #orange1 {
                height: 150px;
                width: 225px;
            }
            .trans {
                transform: skew(10deg, 10deg);
            }
        </style>
        <script>
            window.onload = function () {
                document.getElementById("orange1").onclick = function () {
                    this.classList.add("trans");
                }
            }
        </script>
    </head>
    <body>
        
    </body>
</html>
```



Ao clicar na imagem:

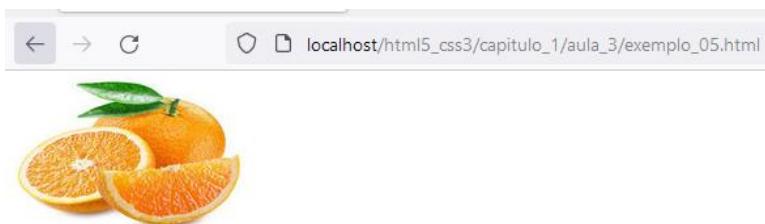


## Método scale

O método `scale` permite que você redimensione os elementos por uma razão especificada. Leva um parâmetro: um valor decimal que representa a porcentagem da escala. Um valor maior que 1 torna o objeto maior; um valor menor que 1, torna o objeto menor. Especificar um valor de -1 vira o objeto sobre seu eixo horizontal.

### aula\_3\exemplo\_05.html

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="utf-8"/>
        <title></title>
        <style>
            #orange1 {
                height: 150px;
                width: 225px;
            }
            .trans {
                transform: scale(1.5);
            }
        </style>
        <script>
            window.onload = function () {
                document.getElementById("orange1").onclick = function () {
                    this.classList.add("trans");
                }
            }
        </script>
    </head>
    <body>
        
    </body>
</html>
```



Ao clicar na imagem:

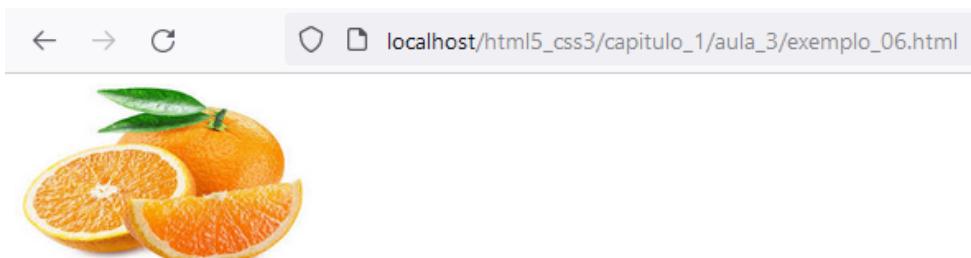


## Combinando transformações

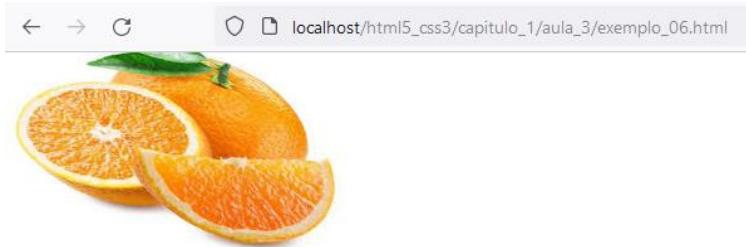
O estilo de transformação não se limita a especificar um único método de transformação. Você pode combinar os métodos para aplicar múltiplos efeitos ao elemento.

### aula\_3\exemplo\_06.html

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="utf-8"/>
        <title></title>
        <style>
            #orange1 {
                height: 150px;
                width: 225px;
            }
            .trans {
                transform: translate(50px,0px) scale(1.5) skew(10deg, 10deg);
            }
        </style>
        <script>
            window.onload = function () {
                document.getElementById("orange1").onclick = function () {
                    this.classList.add("trans");
                }
            }
        </script>
    </head>
    <body>
        
    </body>
</html>
```



Ao clicar na imagem:



## Exibindo e escondendo elementos

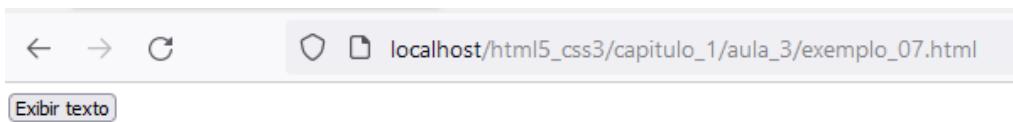
Você pode exibir e ocultar elementos declarativamente na marcação HTML ou programáticamente modificando as propriedades CSS do objeto através do JavaScript.

### Propriedade display

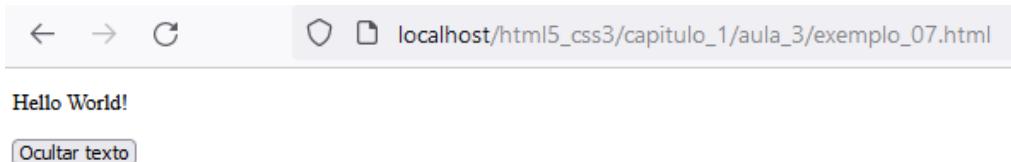
**aula\_3\exemplo\_07.html**

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="utf-8"/>
        <title></title>
        <style>
            #innerDiv{
                display: none;
            }
        </style>
        <script>
            window.onload = function () {
                document.getElementById("btnHideAnElement").onclick = function () {
                    if (document.getElementById("innerDiv").style.display == 'inline') {
                        document.getElementById("innerDiv").style.display = 'none';
                        btnHideAnElement.innerText = "Exibir texto";
                    } else {
                        document.getElementById("innerDiv").style.display = 'inline';
                        btnHideAnElement.innerText = "Ocultar texto";
                    }
                }
            }
        </script>
    </head>
    <body>
        <form>
            <div id="innerDiv">
                <p>Hello World!</p>
            </div>

            <button type="button" id="btnHideAnElement">Exibir texto</button>
        </form>
    </body>
</html>
```



Ao clicar no botão:



Este código modifica o bloco de script e adiciona um novo botão na parte inferior da página. O botão está conectado a um evento de onclick após o término da janela de carregamento. Neste caso, você modifica programáticamente a visibilidade dos elementos HTML.

Quando você usa a propriedade `display` e a define com o valor `none`, o elemento HTML fica escondido. Mas escondendo o elemento desta maneira, você também o remove do layout. Todos os elementos ao redor realinham-se como se o elemento não existisse. Quando `display` é definido como `inline`, o elemento é mostrado novamente e todos os elementos ao redor se movem para fora do caminho, de volta para onde eles estavam originalmente.

## Propriedade visibility

TABLE 1-12 Values available for the *visibility* property

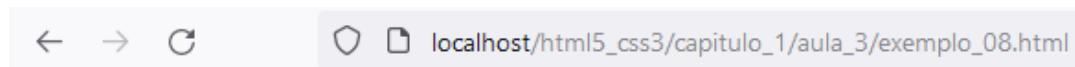
Value	Effect
<code>visible</code>	Sets the property to <code>visible</code> to show the element
<code>hidden</code>	Hides the element
<code>collapse</code>	Collapses the element where applicable, such as in a table row
<code>inherit</code>	Inherits the value of the <code>visibility</code> property from the parent

A propriedade `visibility` se comporta ligeiramente diferente. Definindo-a como `hidden`, o elemento é escondido, mas os elementos ao redor do elemento oculto agem como se ele ainda estivesse lá. O espaço do elemento ocupado é mantido intacto, mas o elemento e seu conteúdo ficam escondidos. Quando `visibility` é definida como `visible`, o elemento reaparece exatamente onde estava, sem afetar nenhum elemento ao redor.

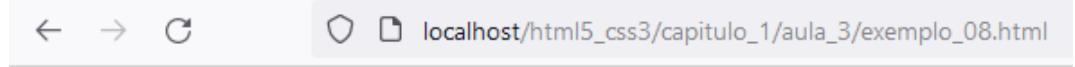
### aula\_3\exemplo\_08.html

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="utf-8"/>
        <title></title>
        <style>
            #innerDiv{
                visibility: hidden;
            }
        </style>
        <script>
            window.onload = function () {
                document.getElementById("btnHideAnElement").onclick = function () {
                    if (document.getElementById("innerDiv").style.visibility == 'visible') {
                        document.getElementById("innerDiv").style.visibility = 'hidden';
                        btnHideAnElement.innerText = "Exibir texto";
                    } else {
                        document.getElementById("innerDiv").style.visibility = 'visible';
                        btnHideAnElement.innerText = "Ocultar texto";
                    }
                }
            }
        </script>
    </head>
    <body>
        <form>
            <div id="innerDiv">
                <p>Hello World!</p>
            </div>

            <button type="button" id="btnHideAnElement">Exibir texto</button>
        </form>
    </body>
</html>
```



Exibir texto



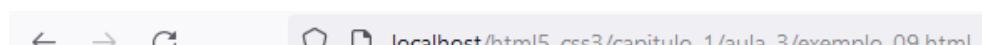
Hello World!

Ocultar texto

### aula\_3\exemplo\_09.html

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="utf-8"/>
        <title></title>
        <style>
            #innerDiv{
                visibility: collapse;
            }
            table{
                border-collapse: collapse;
                margin-bottom: 10px;
            }
            td{
                padding: 5px;
            }
        </style>
        <script>
            window.onload = function () {
                document.getElementById("btnHideAnElement").onclick = function () {
                    if (document.getElementById("innerDiv").style.visibility == 'visible') {
                        document.getElementById("innerDiv").style.visibility = 'collapse';
                        btnHideAnElement.innerText = "Exibir tabela";
                    } else {
                        document.getElementById("innerDiv").style.visibility = 'visible';
                        btnHideAnElement.innerText = "Ocultar tabela";
                    }
                }
            }
        </script>
    </head>
    <body>
        <form>
            <div id="innerDiv">
                <table border="1">
                    <tr>
                        <td>Linha 1</td>
                        <td>Texto da linha 1</td>
                    </tr>
                    <tr>
                        <td>Linha 2</td>
                        <td>Texto da linha 2</td>
                    </tr>
                    <tr>
                        <td>Linha 3</td>
                        <td>Texto da linha 3</td>
                    </tr>
                    <tr>
                        <td>Linha 4</td>
                        <td>Texto da linha 4</td>
                    </tr>
                    <tr>
                        <td>Linha 5</td>
                        <td>Texto da linha 5</td>
                    </tr>
                </table>
            </div>
            <button type="button" id="btnHideAnElement">Exibir tabela</button>
        </form>
    </body>
</html>
```

```
</form>
</body>
</html>
```



## Dica para exames

Se você precisar preservar o layout da página quando alterar a visibilidade, use a propriedade **visibility** com valor **hidden**. Se você não precisa preservar o layout, você pode usar a propriedade **display** com o valor **none** ou **visibility** com valor **collapse**.

## Resumo

- Você pode usar o CSS para definir efeitos de transformação.
- Você pode aplicar transformações via JavaScript para manipular o DOM com efeitos como girar, distorcer, aumentar/diminuir e mover.
- A propriedade **visibility** oferece opções para controlar a visibilidade de um elemento dentro da página.

## Objective Review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the “Answers” section at the end of this chapter.

1. Absolute positioning positions an object relative to what?

- A. The top-left corner of the browser window.
- B. The top-left corner of its parent element.**
- C. Centered inside the window.
- D. Centered inside its parent element.

- A. Incorreto: O elemento não está posicionado em relação à janela do navegador.
- B. Correto: O posicionamento absoluto posiciona o elemento em relação ao seu elemento pai.
- C. Incorreto: O elemento não se centraliza dentro da janela.
- D. Incorreto: O posicionamento absoluto não centraliza um elemento.

2. Which transformation enables you to change the size of an element?

- A. rotate
- B. skew
- C. translate
- D. scale**

- A. Incorreto: rotate gira um objeto no sentido horário ou no sentido anti-horário.
- B. Incorreto: skew inclina um objeto.
- C. Incorreto: translate move um objeto.
- D. Correto: scale altera o tamanho de um objeto.

3. Which syntax preserves the layout of the page when hiding an element in the DOM?

- A. display='hidden'
- B. display='inline'
- C. visibility='none'
- D. visibility='hidden'**

- A. Incorreto: display='hidden' não é uma opção válida.
- B. Incorreto: display='inline' mostra um objeto que anteriormente não estava sendo mostrado.
- C. Incorreto: visibility ='none' não é uma opção válida.
- D. Correto: visibility ='hidden' esconde um elemento, e seus elementos circundantes permanecem no lugar como se o elemento ainda estivesse lá.

## Aula 4 - Implementar API's HTML5

As APIs javaScript fornecem algumas novas funcionalidades poderosas, como a capacidade de armazenar mais dados localmente e disponibilizar esses dados para a página da Web através da **API web storage**.

A **API appCache** permite que você utilize aplicativos da Web de forma off-line. A **API geolocation** fornece métodos para trabalhar com posicionamento global dentro do aplicativo.

Nesta aula veremos como:

- Usar API de armazenamento
- Usar API appCache
- Usar API de Geolocalização

### API de armazenamento (Web Storage)

Web Storage é uma nova API para armazenar dados de páginas da Web localmente. **Web Storage substitui o conceito de cookies**.

OBS.: Claro, você deve levar em conta se o seu navegador tem suporte a Web Storage antes de usá-lo.

Existem duas formas de armazenamento na Web: local e de sessão.

O armazenamento local é persistente; dados armazenados em **localStorage** estão disponíveis para a página web, mesmo que o usuário feche completamente o navegador e depois o reabra; **sessionStorage** está disponível apenas para a duração da sessão atual, portanto, se o usuário fechar o navegador, o armazenamento da sessão será limpo automaticamente e não estará mais disponível.

A API Web Storage está disponível como um objeto global. Para acessar local storage, use o **objeto localStorage**; para acessar session storage, use o **objeto sessionStorage**.

### Dica para exames

Os objetos **localStorage** e **sessionStorage** fornecem exatamente a mesma API. Todos os exemplos mostrados nesta seção funcionam exatamente da mesma forma com qualquer um dos objetos. A única diferença é a vida útil do armazenamento; **sessionStorage** é liberado quando a sessão é encerrada, enquanto **localStorage** ainda estará acessível após o fechamento de uma sessão e a abertura de uma nova sessão.

**TABLE 1-13** Methods available on storage objects

Method	Description
<i>setItem</i>	Adds a key/value pair into storage. If no item with the specified key exists, the item is added; if that key does exist, its value is updated.
<i>getItem</i>	Retrieves data from storage based on a specified key value or index.
<i>clear</i>	Clears all storage that has been saved. Use this method to clear out the storage as needed.
<i>key</i>	Retrieves the key at a specified index. You can use the resultant key to pass as a parameter to one of the other methods that accepts a key.
<i>removeItem</i>	Removes the specified key/value pair from storage.

Além dos métodos descritos na Tabela 1-13, storage objects dispõem da propriedade **length** que devolve o número de pares chave/valor armazenados.

#### aula\_4\exemplo\_01.html

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta charset="utf-8"/>
        <title></title>
        <style>
            section {
                margin-top: 15px;
            }
        </style>
        <script>
            window.onload = function () {

                LoadFromStorage();

                document.getElementById("btnAdd").onclick = function () {
                    localStorage.setItem(document.getElementById("toStorageKey").value,
                        document.getElementById("toStorageValue").value);
                    LoadFromStorage();
                }

                document.getElementById("btnRemove").onclick = function () {
                    localStorage.removeItem(document.getElementById("toStorageKey").value);
                    LoadFromStorage();
                }

                document.getElementById("btnClear").onclick = function () {
                    localStorage.clear();
                    LoadFromStorage();
                }

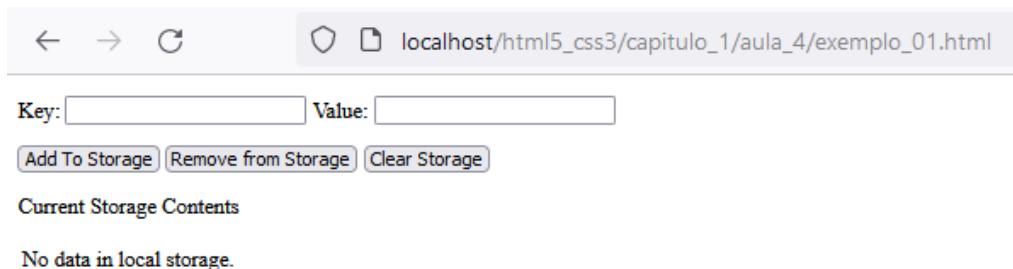
                function LoadFromStorage() {
                    var storageDiv = document.getElementById("storage");
                    var tbl = document.createElement("table");
                    tbl.id = "storageTable";
                    if (localStorage.length > 0) {
                        for (var i = 0; i < localStorage.length; i++) {

```

```

        var row = document.createElement("tr");
        var key = document.createElement("td");
        var val = document.createElement("td");
        key.innerText = localStorage.key(i);
        val.innerText = localStorage.getItem(key.innerText);
        row.appendChild(key);
        row.appendChild(val);
        tbl.appendChild(row);
    }
} else {
    var row = document.createElement("tr");
    var col = document.createElement("td");
    col.innerText = "No data in local storage.";
    row.appendChild(col);
    tbl.appendChild(row);
}
if (document.getElementById("storageTable")) {
    document.getElementById("storageTable").replaceNode(tbl);
} else {
    storageDiv.appendChild(tbl);
}
}
}
</script>
</head>
<body>
<section>
    Key:
    <input type="text" id="toStorageKey"/>
    Value:
    <input type="text" id="toStorageValue"/><br/>
</section>
<section>
    <button type="button" id="btnAdd">Add To Storage</button>
    <button type="button" id="btnRemove">Remove from Storage</button>
    <button type="button" id="btnClear">Clear Storage</button>
</section>
<div id="storage">
    <p>Current Storage Contents</p>
</div>
</body>
</html>

```



O código cria caixas de texto para aceitar uma chave e um valor, respectivamente. Botões permitem que você adicione itens ao armazenamento, remova um item, ou limpe completamente o armazenamento. Para exibir o conteúdo do armazenamento, a página contém uma div que mostra o conteúdo do armazenamento anexado a ele.

Para consultar o que há armazenado no web storage, faça o seguinte:

No browser, selecione Ferramentas de Desenvolvedor (<F12>). Em seguida, selecione a aba "Aplicativo":

The screenshot shows the Microsoft Edge DevTools interface with the 'Aplicativo' (Application) tab selected. In the main pane, there is a table titled 'Current Storage Contents' with two columns: 'Chave' (Key) and 'Valor' (Value). Below this, a message says 'No data in local storage.' At the bottom right of the pane, a tooltip reads 'Selecione um valor a ser visualizado' (Select a value to be displayed). On the left, there are several sections: 'Aplicativo' (Manifest, Service Workers, Armazenamento), 'Armazenamento' (Armazenamento Local, http://localhost, Armazenamento de Sessão, IndexedDB, Web SQL, Cookies, Tokens de Confiança), 'Cache' (Armazenamento em cache, Cache do Aplicativo), 'Serviços em Segundo plano' (Busca em Segundo Plano, Sincronização em Segundo Plano, Notificações, Manipulador de Pagamentos, Sincronização periódica em seguran..., Mensagens por Push), and 'Quadros' (top).

O método `LoadFromStorage` é chamado em cada operação para atualizar a página com os dados disponíveis no armazenamento. Todos os exemplos seguintes utilizam o armazenamento local, mas funcionariam da mesma forma com o armazenamento de sessões.

Se você quiser testar estes exemplos usando o armazenamento de sessão, simplesmente substitua a referência `localStorage` por uma referência à `sessionStorage`.

O código anterior implementa o evento `onclick` de cada botão. Um usuário pode agora adicionar itens para o armazenamento local e ver o que está armazenado. O usuário pode continuar adicionando ao armazenamento local nesta aplicação até que o armazenamento esteja cheio. A disponibilidade do armazenamento local é limitada, e o armazenamento disponível não é consistente em todos os navegadores.

A documentação afirma que O Microsoft Internet Explorer 10 suporta até cerca de 10 MB de armazenamento. Entretanto, isso poderia e pode não ser o mesmo em outros navegadores; alguns agora suportam apenas 5 MB de armazenamento. Tenha isto em mente ao projetar aplicações web que tirem proveito do armazenamento web.

Execute o exemplo anterior e adicione os seguintes itens ao `localStorage`:  
("Red", "FF0000"), ("Green", "00FF00"), ("Blue", "0000FF").

The screenshot shows the Microsoft Edge DevTools interface with the 'Aplicativo' (Application) tab selected. In the main pane, there is a table titled 'Current Storage Contents' with two columns: 'Chave' (Key) and 'Valor' (Value). The table now contains three entries: "Blue" "0000FF", "Green" "00FF00", and "Red" "FF0000". Below this, a message says 'No data in local storage.' On the left, there are several sections: 'Aplicativo' (Manifest, Service Workers, Armazenamento), 'Armazenamento' (Armazenamento Local, http://localhost, Armazenamento de Sessão, IndexedDB, Web SQL, Cookies, Tokens de Confiança), 'Cache' (Armazenamento em cache, Cache do Aplicativo), 'Serviços em Segundo plano' (Busca em Segundo Plano, Sincronização em Segundo Plano, Notificações, Manipulador de Pagamentos, Sincronização periódica em seguran..., Mensagens por Push), and 'Quadros' (top).

The screenshot shows the Microsoft Edge DevTools interface with the 'Aplicativo' (Application) tab selected. On the left, there's a sidebar with sections for 'Aplicativo' (Manifest, Service Workers, Armazenamento), 'Armazenamento' (Armazenamento Local), and a network entry for 'http://localhost'. The main area is a table titled 'Aplicativo' with columns 'Chave' (Key) and 'Valor' (Value). It contains three items: 'Red' with value 'FF0000', 'Green' with value '00FF00', and 'Blue' with value '0000FF'.

Chave	Valor
"Red"	"FF0000"
"Green"	"00FF00"
"Blue"	"0000FF"

Agora, se você fechar o navegador e depois reabrir sua página, os itens ainda estão disponíveis em localStorage. Tente substituir todos os usos do localStorage por sessionStorage. Desta vez, observe que o fechamento do navegador limpa automaticamente quaisquer dados no armazenamento.

O benefício de usar o Web Storage API ao invés de cookies é que os dados são armazenados localmente. Os dados não são enviados de e para o servidor, como no caso dos cookies. Os dados armazenados em Web Storage são organizados por domínio raiz. Por exemplo, domínios como o localhost ou microsoft.com cada um obtém seu próprio espaço de armazenamento seguro na web.

Web storage permite o armazenamento apenas de pares chave/valor. Se você precisar armazenar objetos mais complexos em web storage, você pode usar algumas técnicas. Por exemplo, adicione o seguinte código logo antes a primeira chamada para LoadFromStorage no evento onload:

## aula\_4\exemplo\_02.html

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta charset="utf-8"/>
    <title></title>
    <style>
      section {
        margin-top: 15px;
      }
    </style>
    <script>
      window.onload = function () {

        var customer = new Object();
        customer.firstName = "Rick";
        customer.lastName= "Delorme";
        customer.shirtSize = "XL";
        localStorage.setItem("cart1", JSON.stringify(customer));
        LoadFromStorage();

        document.getElementById("btnAdd").onclick = function () {
          localStorage.setItem(document.getElementById("toStorageKey").value,
            document.getElementById("toStorageValue").value);
          LoadFromStorage();
        }

        document.getElementById("btnRemove").onclick = function () {
          localStorage.removeItem(document.getElementById("toStorageKey").value);
          LoadFromStorage();
        }

        document.getElementById("btnClear").onclick = function () {
          localStorage.clear();
          LoadFromStorage();
        }

        function LoadFromStorage() {
          var storageDiv = document.getElementById("storage");
          var tbl = document.createElement("table");
          tbl.id = "storageTable";
          if (localStorage.length > 0) {
            for (var i = 0; i < localStorage.length; i++) {
              var row = document.createElement("tr");
              var key = document.createElement("td");
              var val = document.createElement("td");
              key.innerText = localStorage.key(i);
              val.innerText = localStorage.getItem(key.innerText);
              row.appendChild(key);
              row.appendChild(val);
              tbl.appendChild(row);
            }
          } else {
            var row = document.createElement("tr");
            var col = document.createElement("td");
            col.innerText = "No data in local storage.";
            row.appendChild(col);
            tbl.appendChild(row);
          }
        }
      }
    </script>
  </head>
  <body>
    <section>
      <table border="1">
        <tr>
          <td>First Name</td>
          <td>Last Name</td>
          <td>Shirt Size</td>
        </tr>
        <tr>
          <td>Rick</td>
          <td>Delorme</td>
          <td>XL</td>
        </tr>
      </table>
      <p>Add item to storage:</p>
      <input type="text" id="toStorageKey" value="customer.firstName"/>
      <input type="text" id="toStorageValue" value="John"/>
      <button type="button" id="btnAdd">Add</button>
      <button type="button" id="btnRemove">Remove</button>
      <button type="button" id="btnClear">Clear</button>
    </section>
  </body>
</html>
```

```

        if (document.getElementById("storageTable")) {
            document.getElementById("storageTable").replaceNode(tbl);
        } else {
            storageDiv.appendChild(tbl);
        }
    }
}
</script>
</head>
<body>
<section>
    Key:
    <input type="text" id="toStorageKey"/>
    Value:
    <input type="text" id="toStorageValue"/><br/>
</section>
<section>
    <button type="button" id="btnAdd">Add To Storage</button>
    <button type="button" id="btnRemove">Remove from Storage</button>
    <button type="button" id="btnClear">Clear Storage</button>
</section>
<div id="storage">
    <p>Current Storage Contents</p>
</div>
</body>
</html>

```

localhost/html5\_css3/capitulo\_1/aula\_4/exemplo\_02.html

Key:  Value:

[Add To Storage](#) [Remove from Storage](#) [Clear Storage](#)

Current Storage Contents

```
cart1 {"firstName": "Rick", "lastName": "Delorme", "shirtSize": "XL"}
```

Aplicativo

- Manifesto
- Service Workers
- Armazenamento

Aplicativo

Chave	Valor
cart1	{"firstName": "Rick", "lastName": "Delorme", "s...

Armazenamento

- Armazenamento Local
  - http://localhost
- Armazenamento de Sessão
  - IndexedDB
  - Web SQL
- Cookies
- Tokens de Confiança

Cache

- Armazenamento em cache
- Cache do Aplicativo

```
{firstName: "Rick", lastName: "Delorme", shirtSize: "XL"}
firstName: "Rick"
lastName: "Delorme"
shirtSize: "XL"
```

Este código cria um objeto personalizado para representar um cliente que navega no site e define qual será o tamanho da camisa do cliente. Esta informação deve ser guardada e utilizada localmente, para que não precise ser postado no servidor. O armazenamento local é uma ótima solução para isso. Entretanto, para armazenar os costumes do usuário em dados locais, você precisa de um método para converter o objeto personalizado em uma string que corresponda ao modelo de local storage. É aqui que pode entrar a Notificação de Objeto JavaScript (JSON) à mão. Você pode serializar o objeto em JSON, dar-lhe uma chave, e depois armazená-lo na web storage. Quando você executa esta aplicação agora, ela mostra o objeto do cliente representado como um Cadeia JSON.

A disponibilidade de armazenamento local na web pode melhorar tanto a experiência do usuário final quanto desempenho de suas aplicações web, economizando idas e vindas ao servidor para recuperar ou armazenar dados temporários. Você deve considerar o armazenamento local na web como temporário. Mesmo quando você estiver usando o localStorage em vez do sessionStorage, você deve pensar no armazenamento como temporário e projete suas aplicações de modo que elas possam cair de volta nos valores e comportamento padrão se o usuário limpa o armazenamento da web. O armazenamento da web proporciona uma forma de disponibilizar dados localmente e até mesmo persistir durante as sessões do navegador. Estas técnicas funcionam com um site conectado ao vivo.

Se você quiser disponibilizar uma aplicação offline, de forma desconectada, você pode usar [AppCache API](#).

## AppCache API

A possibilidade de continuar a trabalhar com aplicações web quando desconectado da Internet se tornou particularmente importante no mundo móvel de hoje.

Esta seção fala sobre como criar uma aplicação que funcione quando desconectada, usando o Cache de Aplicação API, também comumente chamado de [AppCache API](#).

A API AppCache torna o conteúdo e as páginas web disponíveis mesmo quando uma aplicação web está em modo off-line. AppCache armazena arquivos no cache do aplicativo no navegador. Assim como com Web Storage, a quantidade de dados que o navegador pode armazenar localmente é limitada para uso offline. Dois componentes compõem o AppCache API: o arquivo manifesto e um JavaScript API para apoiá-lo.

### Usando AppCache manifest

Especificando que uma página deve estar disponível para uso offline é tão fácil quanto adicionar um atributo a um elemento HTML na página. Aqui está um exemplo:

```
<html manifest="webApp.appcache">  
...  
</html>
```

O atributo manifest no elemento html diz ao navegador que esta página web precisa estar disponível off-line. O valor do atributo manifest aponta para um arquivo manifest. O nome do arquivo é uma convenção mais do que uma exigência; você pode nomear o arquivo qualquer coisa, mas a extensão do arquivo é geralmente [.appcache](#)

## Dica para exames

Se você realmente quer mudar a extensão do arquivo, você precisa configurar o servidor web para que sua extensão de arquivo escolhida seja devolvida com um tipo MIME de texto/cache-manifest.

O arquivo manifest da aplicação deve listar cada um dos arquivos e recursos necessários para ser armazenado para uso offline. Quando o navegador analisa o atributo manifest do elemento html, ele baixa o manifest e o armazena localmente. Ele também garante que serão baixados todos os arquivos listados no manifest para que estejam disponíveis offline. **O arquivo manifest contém três seções: CACHE, NETWORK, e Fallback.** Cada seção pode aparecer apenas uma vez, várias vezes no arquivo, ou não aparecer. Cada uma serve a um propósito específico com relação a como o cache de aplicação funciona ao lidar com os recursos em cenários específicos. Um arquivo manifest típico se parece com isto:

### webApp.appcache

```
CACHE MANIFEST
# My Web Application Cache Manifest
# v.1.0.0.25
#
#Cache Section. All Cached items.
```

```
CACHE MANIFEST
html5apis.html

#Required Network resources
NETWORK:
html5apis.html

#Fallback items.
FALLBACK:html5apis.html html5apis-appcache.html
```

A primeira linha em um arquivo manifest deve ser CACHE MANIFEST. O arquivo manifest, pode ter linhas de comentário adicionadas para explicações adicionais, como indicado no código pelo símbolo #.

A seção CACHE lista todos os recursos que devem ser armazenados em cache offline. Isto deve incluir todos os arquivos CSS, arquivos JPG, arquivos de vídeo e áudio, e qualquer outro recurso necessário para a página funcionar corretamente. Se você omitir um item do arquivo do manifest, ele não será colocado em cache, o que pode resultar em comportamento inesperado quando a aplicação é executada offline.

A seção NETWORK declara todos os recursos que devem estar disponíveis na Internet. Estes itens não podem ser armazenados em cache. Qualquer coisa que a página exija da Internet, como por exemplo elementos incorporados de terceiros, devem ser listados aqui. Se tal recurso não estiver listado aqui, o navegador não saberá verificar na Internet quando estiver no modo off-line. Quando o navegador está em modo offline, não tenta acessar à Internet por nada a menos que esteja listado na seção NETWORK.

A seção Fallback permite que você forneça instruções de retorno para o navegador no caso de um item não estar disponível no cache e o navegador estiver em modo offline. No arquivo de exemplo, se o login.html não estiver disponível no cache, será renderizada o arquivo fallback-login.html.

Você pode usar atalhos na seção Fallback para fornecer redirecionamentos mais gerais, tais como os seguintes:

```
/resources /resource.jpg
```

Isto diz ao navegador que se o navegador estiver offline e não puder acessar nada na pasta resources, ela deve substituir qualquer referência a itens da pasta de recursos por resource.jpg. Note que resource.jpg está em cache porque está especificado na seção Fallback. Você não precisa especificar também resource.jpg na seção Cache.

## Usando AppCache API

Assim como no Web Storage, o cache do aplicativo está disponível em JavaScript como um objeto global. O código seguinte obtém uma referência ao objeto global AppCache:

```
var appCache = window.applicationCache;
```

Quando você está usando o cache da aplicação para disponibilizar páginas offline, um das mais coisas úteis que você pode fazer quando a página é carregada é verificar seu status. Você consegue isto avaliando o status de propriedade do objeto AppCache. O status de propriedade pode ser um dos os valores listados na Tabela 1-14.

**TABLE 1-14** The application cache *status* property

Status	Description
<i>Uncached</i>	The web application isn't associated with an application manifest.
<i>Idle</i>	The caching activity is idle, and the most up-to-date copy of the cache is being used.
<i>Checking</i>	The application manifest is being checked for updates.
<i>Downloading</i>	The resources in the application manifest are being downloaded.
<i>UpdateReady</i>	The resources listed in the manifest have been successfully downloaded.
<i>Obsolete</i>	The manifest can no longer be downloaded, so the application cache is being deleted.

Após conhecer o status do cache, dois métodos no objeto AppCache podem ser úteis. A tabela 1-15 lista estes métodos.

**TABLE 1-15** Methods available with the *applicationCache* object

Method	Description
<i>swapCache</i>	Indicates that the cache be replaced with a newer version.
<i>update</i>	Tells the browser to update the cache if an update is available.

Quando o método de atualização é chamado, uma atualização do cache é preparada. Quando isso estiver pronto para download, o status do cache do aplicativo muda para *UpdateReady*. Quando isto é definido, uma chamada para o método *swapCache* diz ao aplicativo para mudar para o cache mais recente.

## Dica para exames

A chamada para o método de atualização é assíncrona. Portanto, você deve lidar com o evento **onupdateready** para determinar quando a atualização completou o processo de download.

O objeto **AppCache** possui uma série de eventos com os quais você pode lidar. O cache de aplicações normalmente opera em segundo plano, e você não vai precisar desses eventos. No entanto, em alguns casos, lidar com alguns dos eventos e forçar uma atualização pode ser útil. A tabela 1-16 lista os eventos disponíveis.

**TABLE 1-16** Events available from the *applicationCache* object

Event	Description
<i>onchecking</i>	The browser is checking for an update to the application manifest, or the application is being cached for the first time.
<i>onnoupdate</i>	The application manifest has no update available.
<i>ondownloading</i>	The browser is downloading what it has been told to do per the manifest file.
<i>onprogress</i>	Files are being downloaded to the offline cache. This event fires periodically to report progress.
<i>oncached</i>	The download of the cache has completed.
<i>onupdateready</i>	The resources listed in the manifest have been newly redownloaded, and the <i>swapCache</i> method might be called.
<i>onobsolete</i>	A manifest file is no longer available.
<i>onerror</i>	An error has occurred. This could result from many things. Appropriate logging is necessary to get the information and resolve.

A maioria desses eventos pode não ser usada com freqüência, ou até mesmo nem ser usada. O cenário mais comum é lidar com o método **onupdateready** e depois fazer uma chamada para o método **swapCache**, como neste exemplo:

```
window.onload = function () {
    var appCache = window.applicationCache;
    appCache.oncached = function (e) { alert("cache successfully downloaded."); };
    appCache.onupdateready = function (e) { appCache.swapCache(); };
}
```

A utilização do cache de aplicação é mais sobre configuração do que sobre codificação. No entanto, é importante que você esteja ciente de que a API está disponível para cenários avançados onde você precisa de mais controle sobre o processo, ou quando você precisar receber informações oportunas sobre o processo, como, por exemplo, ao lidar com o evento **onprogress**.

## GeoLocation API

Os serviços de localização tornaram-se comum na vida da maioria das pessoas. Cada vez mais pessoas estão usando alguma forma de serviços de localização. Os serviços de localização dependem do Sistema de Posicionamento Global (GPS), endereços IP e outras características. Você pode tirar proveito da geolocalização em aplicações web alavancando navegadores que suportam o API de Geolocalização.

Você pode obter uma referência ao API de Geolocalização a partir da propriedade `window.navigator`, como segue:

```
var geoLocator = window.navigator.geolocation;
```

Este código guarda uma referência à API de Geolocalização em uma variável para fornecer acesso a API durante o uso futuro. Uma boa prática é garantir que o navegador do cliente apóie a API de Geolocalização, certificando-se de que a referência esteja realmente presente.

O API de Geolocalização suporta três métodos-chave que você utiliza para interagir com ele:

1. `getCurrentPosition`
2. `watchPosition`
3. `clearWatch`.

### Método `getCurrentPosition`

Aqui está um exemplo de uso do método `getCurrentPosition`:

```
getCurrentPosition(positionCallback, [positionErrorCallback], [positionOptions])
```

Você usa `getCurrentPosition` para obter a posição atual do usuário ou do dispositivo em que a aplicação está sendo executada. Este método requer um parâmetro e dois parâmetros opcionais. O primeiro parâmetro é um método de chamada de retorno que o API chama depois que a posição atual for determinada. O segundo parâmetro é opcional, mas também é um função de chamada retornada quando ocorre um erro. O método de chamada de retorno que você especifica aqui deve lidar com quaisquer erros que possam ocorrer ao tentar obter a posição atual. A última opção é um objeto especial chamado `PositionOptions`, que lhe permite definir algumas opções especiais que controlam como o método `getCurrentPosition` se comporta. A tabela 1-17 lista os valores possíveis.

**TABLE 1-17** Properties available on the `PositionOptions` object

Property	Description
<code>enableHighAccuracy</code>	This causes the method to be more resource intensive if set to true. The default is false. If true, the <code>getCurrentPosition</code> method tries to get as close as it can to the actual location.
<code>timeout</code>	This specifies a timeout period for how long the <code>getCurrentPosition</code> method can take to complete. This number is measured in milliseconds and defaults to zero. A value of zero represents infinite.

Property	Description
<i>maximumAge</i>	If this is set, the API is being told to use a cached result if available, rather than make a new call to get the current position. The default is zero, so a new call is always made. If <i>maximumAge</i> is set to a value and the cache isn't older than the allowable age, the cached copy is used. This value is measured in milliseconds.

### aula\_4\geolocation.html

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>
    <meta charset="utf-8" />
    <title></title>
    <script>
        window.onload = function () {
            var geoLocator = window.navigator.geolocation;
            var posOptions = { enableHighAccuracy: true, timeout: 45000 };
            geoLocator.getCurrentPosition(successPosition, errorPosition, posOptions);
        }

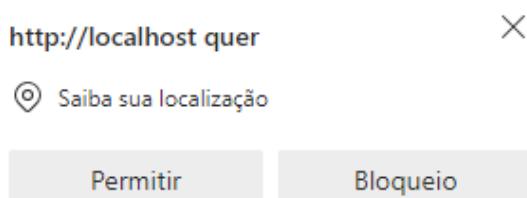
        function successPosition(pos) {
            console.log(pos);
        }

        function errorPosition(err) {
            console.log(err);
        }
    </script>
</head>

<body>
    <div id="geoResults">
        <p>Current Location is:</p>
    </div>
</body>

</html>
```

Quando o código é executado no navegador, algumas coisas interessantes podem acontecer. Primeiro, como segurança; os usuários são questionados se querem permitir que esta aplicação determine sua localização. No Internet Explorer, a mensagem é parecida com a imagem da figura a seguir.



Se o usuário optar por permitir que a aplicação prossiga, tudo ocorre normalmente. Caso contrário, é lançada uma exceção.

Para fins de demonstração do código, selecione Permitir para este site a partir da lista suspensa para que a página possa prosseguir. Pode levar alguns segundos, mas a chamada retorna e mostra um caixa de mensagem de que um objeto de posição existe como passado para o método de chamada de retorno de sucesso.

Tanto o sucesso como o erro dos métodos de chamada de retorno recebem um parâmetro da API Geolocalização. O método de sucesso recebe um objeto de posição, enquanto o método de erro recebe um objeto de erro. O objeto de posição expõe duas propriedades: coords e timestamp. A propriedade timestamp indica a hora em que as coordenadas foram recebidas. A propriedade coords é em si um objeto de coordenadas que contém a latitude, longitude, altitude, direção e velocidade da posição atual do dispositivo e/ou em relação à última posição adquirida. A posiçãoError object contém duas propriedades: uma para o código e outra para a mensagem.

#### **aula\_4\geolocation2.html**

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

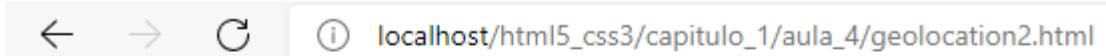
<head>
    <meta charset="utf-8" />
    <title></title>
    <script>
        window.onload = function () {
            var geoLocator = window.navigator.geolocation;
            var posOptions = { enableHighAccuracy: true, timeout: 45000 };
            geoLocator.getCurrentPosition(successPosition, errorPosition, posOptions);
        }

        function successPosition(pos) {
            var sp = document.createElement("p");
            sp.innerText = "Latitude: " + pos.coords.latitude +
                " Longitude: " + pos.coords.longitude;
            document.getElementById("geoResults").appendChild(sp);
        }

        function errorPosition(err) {
            var sp = document.createElement("p");
            sp.innerText = "error: " + err.message + " code: " + err.code;
            document.getElementById("geoResults").appendChild(sp);
        }
    </script>
</head>

<body>
    <div id="geoResults">
        <p>Current Location is:</p>
    </div>
</body>

</html>
```



Current Location is:

Latitude: -23.643108 Longitude: -46.601285

## Método watchPosition

O segundo método disponível no objeto de geolocalização é o método `watchPosition`, que fornece um mecanismo embutido que faz pesquisas contínuas para a posição atual. Aqui está um exemplo de utilização do método:

```
geoLocator.watchPosition(successCallBack,errorCallback,positionOptions)
```

O método `watchPosition` toma o mesmo conjunto de parâmetros que o método `getCurrentPosition` mas retorna um objeto `watchPosition`:

```
var watcher = geoLocator.watchPosition...
```

Depois de executar este código, a variável `watcher` conterá uma referência à posição `watchPosition` da instância invocada, que pode ser útil mais tarde. O método chama a função de sucesso sempre que o API de Geolocalização detecta um novo local. A pesquisa continua para sempre a não ser que você impeça. Você pode cancelar as pesquisas chamando o método `clearWatch`. Você pode chamar este método tanto no sucesso quanto no erro. Por exemplo, para cancelar a pesquisa quando você tiver capturado informações suficientes sobre a posição ou quando quiser interromper a pesquisa por um período de tempo:

```
geoLocator.clearWatch(watcher);
```

## aula\_4\geolocation3.html

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

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

    var watcher;
    var geoLocator;

    window.onload = function () {
      geoLocator = window.navigator.geolocation;
      var posOptions = {enableHighAccuracy: true,timeout: 45000};
      watcher = geoLocator.watchPosition(successPosition, errorPosition, posOptions);
    }

    function successPosition(pos) {
      var sp = document.createElement("p");
      sp.innerText = "Latitude: " + pos.coords.latitude + " Longitude: " + pos.coords.longitude;
      document.getElementById("geoResults").appendChild(sp);
      geoLocator.clearWatch(watcher);
    }
  </script>
</head>
<body>
  <div id="geoResults"></div>
</body>
</html>
```

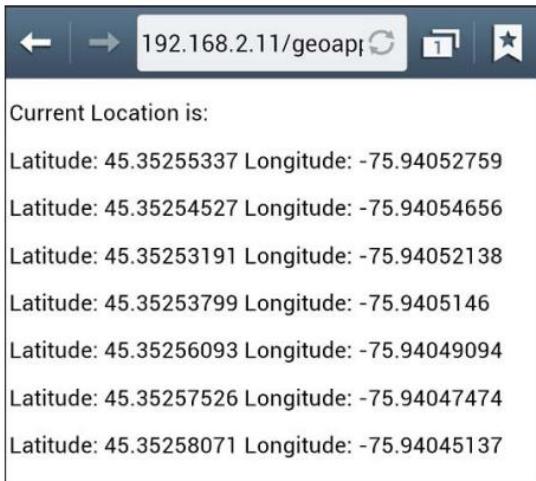
```

function errorPosition(err) {
    var sp = document.createElement("p");
    sp.innerText = "error: " + err.message + " code: " + err.code;
    document.getElementById("geoResults").appendChild(sp);
}
</script>
</head>

<body>
<div id="geoResults">
    <p>Current Location is:</p>
</div>
</body>
</html>

```

Figure 1-55 shows the output of this code on a mobile device.



**FIGURE 1-55** Multiple positions being recorded by the *watchPosition* method

## Resumo

- A nova API Web Storage permite armazenar dados localmente no computador do cliente.
- Web Storage suporta tanto o localStorage quanto o sessionStorage.
- Os dados em Web storage são armazenados como pares de nomes e valores.
- A API AppCache fornece uma forma de disponibilizar páginas web quando os usuários estão offline.
- O manifest AppCache define o que está disponível offline.
- A API de Geolocalização fornece uma forma de integrar serviços de localização em uma página da web.
- A API de Geolocalização fornece dois métodos: getPosition e watchPosition.

## Objective Review

1. When using the Web Storage API, where should you store data to ensure that it's cleared when the user closes the browser?

- A. localStorage
- B. cookieStorage
- C. sessionStorage
- D. A hidden input element

A. Incorreto: localStorage é persistente mesmo após o encerramento da sessão.  
B. Incorreto: o cookieStorage não existe na API de armazenamento da Web.  
C. Correto: sessionStorage se desmarca quando a sessão é encerrada.  
D. Incorreta: Um elemento de entrada escondido não é uma solução válida para atender ao requisito.

2. What do you need to do to designate a page as available offline?

- A. Specify in JavaScript as document.offLine=true.
- B. Specify the manifest attribute on the form element.
- C. Specify the manifest attribute on the HTML element.
- D. Tell users to switch to offline mode using their browser. No code is required.

A. Incorreto: A propriedade offLine não é uma opção JavaScript válida.  
B. Incorreto: Você não deve especificar o manifest no elemento <form>.  
C. Correto: Especificar o atributo manifest no elemento HTML é a ação correta.  
D. Incorreto: A opção offline do navegador não invoca a API AppCache.

3. Which of the following aren't valid sections of the AppCache manifest?

- A. Cache manifest
- B. Session manifest
- C. Network manifest
- D. Fallback manifest

A. Incorreto: Cache manifest lista todos os recursos que devem ser armazenados em cache offline.  
B. Correto: Session manifest não é uma opção válida.  
C. Incorreto: Network manifest especifica todos os recursos que devem estar disponíveis na Internet.  
D. Incorreto: Fallback manifest permite que você diga ao navegador o que fazer quando recursos não estão disponíveis offline.

4. Which event is fired by the AppCache object when the cache download is complete?

- A. oncached
- B. onupdateready
- C. ondownloading
- D. onchecking

A. Correto: oncached é disparado quando o download é concluído.  
B. Incorreto: onupdateready é disparado quando os itens do manifest são recém baixados e o método swapCache pode ser chamado.  
C. Incorreto: ondownloading é disparado quando o navegador está fazendo o download.  
D. Incorreto: onchecking é disparado quando o navegador está verificando por atualizações.

5. When using the Geolocation API, how do you configure the ability to use cached data?

- A. Set the enableCache property to true on the PositionOptions object.
  - B. Set the maximumAge property to a non-zero value on the PositionOptions object.**
  - C. Set the timeout property of the PositionOptions object.
  - D. Using the cache is always on to save bandwidth, so no configuration is required.
- 
- A. Incorreto: A propriedade enableCache não existe.
  - B. Correto: Defina a propriedade MaximumAge para um valor diferente de zero no objeto PositionOptions.
  - C. Incorreto: A propriedade timeout especifica quanto tempo esperar por uma resposta antes de disparar o evento de timeout.
  - D. Incorreto: O cache maximumAge não tem o valor padrão de 0, portanto o cache está desligado por padrão.

## Aula 5 - Estabelecer o escopo de objetos e variáveis

Um componente chave de qualquer linguagem de programação é como ela usa variáveis, e o JavaScript não é uma exceção. Para utilizar as variáveis de forma eficaz, é preciso entender seu escopo e sua vida útil.

A declaração de variáveis e objetos instanciadores consome recursos. O principal recurso do sistema utilizado para as variáveis é a memória.

Nesta aula veremos como:

- Estabelecer o ciclo de vida de variáveis e escopos de variáveis
- Evitar usar namespaces globais
- Utilizar "this"

### Estabelecendo o ciclo de vida de variáveis e escopos de variáveis

Variáveis iniciam seu ciclo de vida ao serem declaradas. Para declarar uma variável em Javascript utiliza-se a palavra **var**.

```
var myVariable;
```

Diversas variáveis podem ser simultaneamente declaradas em uma única linha:

```
var x, y, z;
```

Variáveis podem ser declaradas e inicializadas com um valor:

```
var x = 0.3, y = 1.2, z = 0.7
```

Até que uma variável seja inicializada, ela tem um valor indefinido. Depois que uma variável está disponível para uso, ela é considerada como "em escopo". A duração durante a qual a variável permanece no escopo, depende de onde a variável é declarada. Uma variável que tem escopo global está disponível em toda a página web. Uma variável com escopo local está disponível apenas dentro de um contexto especificado (um bloco de código).

## aula\_5\exemplo\_01.html

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta charset="utf-8"/>
    <style>
      div {
        width: 100px;
        height: 100px;
        border: 1px solid black;
      }
    </style>
    <script>
      var globalVar = "global";

      window.onload = function () {

        var localVar = "local";

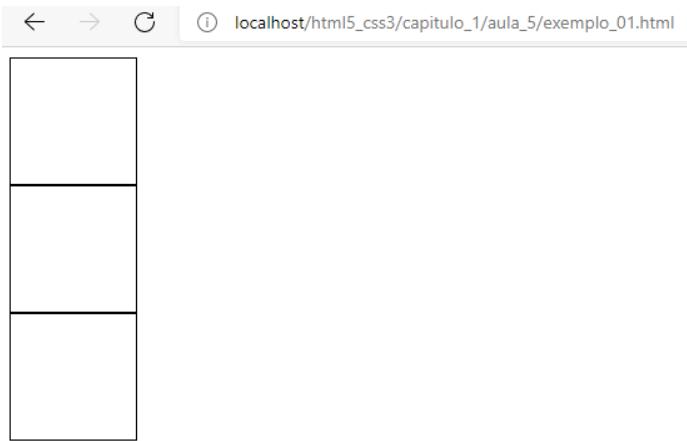
        document.getElementById("Div1").onclick = function () {
          var insideDiv1Click = "insideDiv1";
          alert(globalVar);
          alert(localVar);
          alert(insideDiv1Click);
        };

        document.getElementById("Div2").onclick = function () {
          alert(globalVar);
          alert(localVar);
          alert(insideDiv1Click);
        };

        document.getElementById("Div3").onclick = function () {
          var insideDiv3 = "Div3";
          AFunction();
          BFunctionWithParam(insideDiv3);
        };

        function AFunction() {
          var x;
          alert(insideDiv3);
        }

        function BFunctionWithParam(p) {
          alert(p);
          alert(localVar);
        }
      }
    </script>
  </head>
  <body>
    <div id="Div1"></div>
    <div id="Div2"></div>
    <div id="Div3"></div>
  </body>
</html>
```



Observe que a primeira linha na seção de scripts é a variável globalVar, que é considerada global para a página toda. Um código JavaScript em qualquer lugar desta página pode acessar esta variável. Portanto dentro do manipulador de eventos onload, o código tem acesso à variável globalVar.

No próximo nível, o código implementa um gerenciador de eventos com `window.onload`. Dentro deste manipulador de eventos, a primeira linha declara uma variável chamada localVar que pode ser vista dentro dos manipuladores de eventos dentro de `window.onload`, mas não fora de `window.onload` (em um outro manipulador de eventos).

Assim clicando na Div1:



Clicando na Div2:



Clicando na Div3, nenhum alerta é exibido.

## Evitando usar namespaces globais

O Javascript já tem algumas palavras reservadas que podem ser utilizadas de forma global. O Javascript possui namespaces globais (variáveis nativas) como: Math, WebSocket e JSON. Portanto, para não criar conflitos, evite criar variáveis globais com esses nomes.

Uma estratégia para evitar colisões de nomes é criar seus próprios namespaces para suas bibliotecas JavaScript. Um padrão a ser considerado para criar nomes únicos no namespace é usar o nome do seu domínio ao contrário, tal como [com.microsoft](#). Como os nomes de domínio são únicos, este padrão ajuda a reduzir a possibilidade de colisões de nomes. O seguinte trecho de código demonstra esta estratégia para criar um namespace para uma biblioteca desenvolvida para um bookstore:

```
var com = {};
com.Bookstore = {};

com.Bookstore.Book = {
    title: 'my book',
    genre: 'fiction'
};

com.Bookstore.Author = {
    firstName: 'R',
    lastName: 'D'
}
```

Ao criar os objetos desta forma, você pode ter certeza razoável de que se outro desenvolvedor criar uma biblioteca útil para administrar os livros que você deseja incluir em seu site, você não terá que preocupar-se com uma colisão de nomes entre seus objetos Livro e Autor e os objetos fornecidos pela outra biblioteca. Ao desenvolver bibliotecas JavaScript reutilizáveis, nunca implemente seu objetos no namespace global.

## Uso da palavra-chave "this"

A palavra-chave **this** permite aos desenvolvedores de JavaScript referenciar diretamente o objeto que o contém. O seguinte trecho de código demonstra o contexto desta palavra-chave:

```
<script>
    //Here, "this" references the global namespace
    this.navigator.geolocation

    window.onload = function () {
        //Here, "this" references the window object
        this...
        document.getElementById("aDiv").onclick = function()
        {
            //Here, "this" references the DIV element
            this...
        }
    }
</script>
```

Quando você usa **this**, você deve entender o contexto dele.

## Resumo

- As variáveis são indefinidas até serem inicializadas.
- As variáveis são escopadas e acessíveis dependendo de onde são declaradas. Se elas forem dentro de uma função, por exemplo, eles são locais para a função.
- Passar parâmetros é a única maneira de disponibilizar uma variável local em outra função.
- O namespace global não deve ser usado porque é compartilhado por todos.
- Deve-se aplicar um namespace a objetos personalizados para evitar conflitos no namespace.
- A palavra-chave **this** fornece acesso direto ao objeto que criou o evento

## Objective Review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the “Answers” section at the end of this chapter.

1. In JavaScript, how do you determine the scope of a variable?

- A. The scope of a variable is global within the context of the page.
- B. The scope of a variable depends on where inside the script it's declared.**
- C. The scope of a variable changes depending on the type it represents.

A. Incorreto: As variáveis são globais somente se declaradas no espaço global.

B. Correto: O escopo de uma variável depende de onde ela é declarada dentro do script.

C. Incorreta: O tipo de uma variável não afeta seu escopo.

2. Why is it important to avoid creating custom JavaScript objects in the global namespace?

A. The global namespace is reserved for the browser.

**B. The global namespace is available to all applications in the session, and using it could result in a naming conflict.**

C. The global namespace creates a security risk to users' systems.

A. Incorreto: O namespace global não é reservado para o navegador.

B. Correto: Porque o namespace global está disponível para todas as aplicações na sessão, sua utilização poderia resultar em um conflito de nomes.

C. Incorreto: O namespace global não cria um risco de segurança para o sistema do usuário.

3. What JavaScript keyword in an event handler can be easily used to reference the object that raised the event?

A. The `it` keyword provides a reference to the object.

B. The `document.current` property provides a reference to the object.

**C. The `this` keyword provides a reference to the object.**

D. No way is available other than to use a selector query to retrieve the object from the DOM.

A. Incorreto: A palavra-chave `it` não existe.

B. Incorreto: A palavra-chave `document.current` keyword não fornece a referência.

C. Correto: A palavra-chave `this` fornece uma referência para o objeto.

D. Incorreto: A palavra-chave `this` fornece um atalho direto para o elemento que gerou o evento.

## Aula 6 - Criar e implementar objetos e métodos

JavaScript é uma linguagem de programação orientada a objetos, o que significa que para desenvolver aplicações em JavaScript de forma eficaz, você deve entender como trabalhar com objetos.

Essencialmente, existem dois tipos de objetos em JavaScript:

1. Objetos JavaScript nativos, que são fornecidos com o próprio JavaScript
2. Objetos personalizados, que os desenvolvedores criam para representar suas próprias construções de dados e comportamentos

Em alguns casos, a criação de um objeto totalmente novo não é necessária. Você pode basear os objetos em outros objetos (se forem um subtipo desse objeto), utilizando herança de objetos, nos quais um objeto herda todos os atributos e comportamentos de outro objeto, mas também pode implementar aspectos adicionais que lhe são exclusivos.

Os objetos encapsulam funcionalidades e informações de estado que são relevantes para eles. A funcionalidade é fornecida na forma de métodos, enquanto as informações de estado são fornecidas na forma de propriedades.

Nesta aula veremos como:

- Implementar objetos nativos
- Criar objetos e propriedades customizadas usando protótipos
- Implementar herança
- Implementar métodos nativos e criar métodos customizados

### Implementando objetos nativos

Os objetos nativos estão disponíveis para os desenvolvedores diretamente através do JavaScript. O JavaScript fornece um grande número de objetos que proporcionam funcionalidade para facilitar a vida dos desenvolvedores.

Alguns objetos nativos estão disponíveis estaticamente, outros exigem que você crie uma instância. Você pode encontrar os dois tipos de objetos no namespace global. Um exemplo de um objeto estático é Math, que está disponível no namespace global e oferece uma grande funcionalidade sem que você tenha que criar uma instância:

```
var squareValue = Math.sqrt(144);
```

Outros objetos, como o Array exigem que você crie uma instância para trabalhar com eles:

```
var listofPrimeNumbers = new Array(1, 2, 3, 5, 7, 11, 13, 17, 19, 23);
```

Este código introduz a palavra-chave `new`, que você usa para instanciar um objeto. Isto diz ao runtime para alocar um novo objeto do tipo especificado. Neste caso, um novo objeto do tipo Array está sendo solicitado. A lista após o tipo de objeto Array é chamada de construtor do objeto. Esta informação pode ser passada ao objeto como parâmetro para construir o estado inicial do objeto.

Alguns objetos têm muitos construtores para escolher, com diferentes conjuntos de parâmetros. A adição de múltiplos construtores é chamada de construtor sobreescrito.

O JavaScript também fornece objetos de encapsulamento que envolvem, por exemplo, tipos nativos. Tipos nativos são definidos como inteiro, string, char, e assim por diante.

Você poderia criar as variáveis txt e num da forma indicada a seguir, mas isso não é muito comum.

```
var txt = new String("my long string");
var num = new Number(5);
```

A sintaxe vista até agora se aplica tanto a objetos nativos quanto a objetos personalizados. Objetos personalizados são criados pelo desenvolvedor, enquanto os objetos nativos são fornecidos pelo núcleo JavaScript.

### **aula\_6\exemplo\_01.html**

```
<html>
  <head>
    <meta charset="utf-8" />
    <title></title>
  </head>

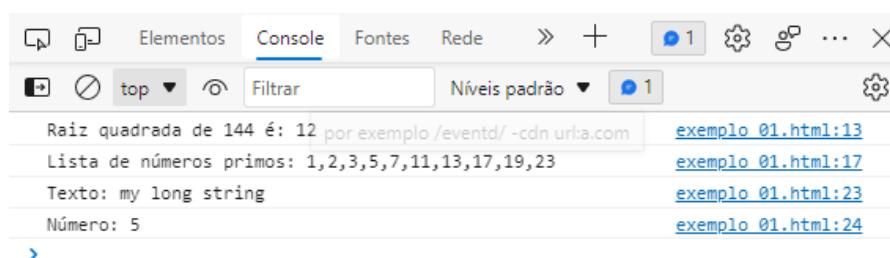
  <body>
    <script>
      //Objetos nativos do JS. Ex: Math, é estático
      var squareValue = Math.sqrt(144);
      console.log("Raiz quadrada de 144 é: " + squareValue);

      //Array, necessita instanciar.
      var listofPrimeNumbers = new Array(1, 2, 3, 5, 7, 11, 13, 17, 19, 23);
      console.log("Lista de números primos: " + listofPrimeNumbers);

      //Objetos tipos nativos/primitivos
      var txt = "my long string"; //string
      var num = 5 //int

      console.log("Texto: " + txt);
      console.log("Número: " + num);

    </script>
  </body>
</html>
```



## Criando objetos personalizados

A criação de objetos personalizados é uma prática padrão quando se trabalha com informações em aplicações personalizadas. Como o JavaScript é uma linguagem orientada a objetos, você deve aplicar práticas apropriadas orientadas a objetos ao desenvolver aplicações JavaScript. Em quase todos os casos, isto envolve a criação de objetos personalizados para encapsular a funcionalidade dentro de entidades lógicas. Por exemplo, o seguinte código cria um objeto de livro. Este é um objeto dinâmico.

```
var book = {  
    ISBN: "55555555",  
    Length: 560,  
    genre: "programming",  
    covering: "soft",  
    author: "John Doe",  
    currentPage: 5  
}
```

O objeto criado representa um livro. Ele fornece uma maneira de encapsular em um único objeto as propriedades que se aplicam a um livro - neste caso, uma entidade do livro. O código especifica cinco propriedades. Ao utilizar a variável livro, você pode acessar todas elas da mesma forma.

As propriedades de um objeto representam seu estado, enquanto que os métodos de um objeto fornecem seu comportamento. Neste ponto, o objeto livro tem apenas propriedades. Para fornecer ao objeto livro alguns comportamento, você pode adicionar o seguinte código:

## aula\_6\exemplo\_02.html

```
<html>

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

    <body>

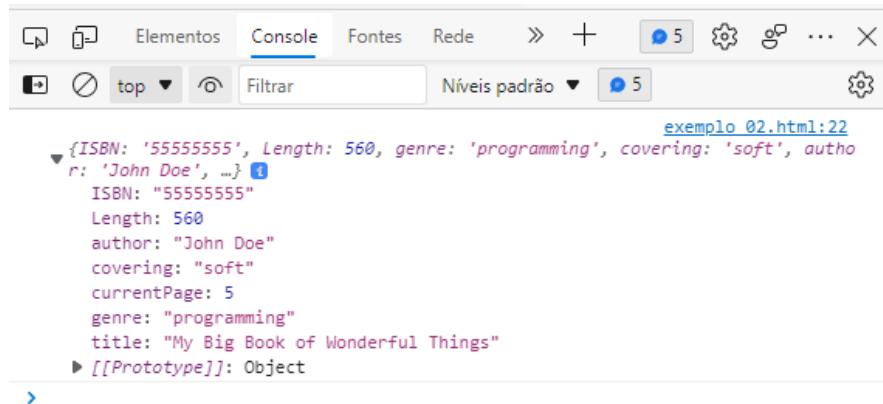
        <script>

            //criando objetos customizados
            var book = {
                ISBN: "55555555",
                Length: 560,
                genre: "programming",
                covering: "soft",
                author: "John Doe",
                currentPage: 5,
                title: "My Big Book of Wonderful Things",
            }
            console.log(book);

        </script>

    </body>

</html>
```



## aula\_6\exemplo\_03.html

```
<html>

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

    <body>

        <script>

            //criando objetos customizados
            var book = {
                ISBN: "55555555",
                Length: 560,
                genre: "programming",
                covering: "soft",
                author: "John Doe",
                currentPage: 5,
                title: "My Big Book of Wonderful Things",
                flipTo: function flipToAPage(pNum) {
                    this.currentPage = pNum;
                    console.log("Fui pra pagina: " + this.currentPage);
                },
                turnPageForward: function turnForward() {
                    this.flipTo(this.currentPage+1);
                },
                turnPageBackward: function turnBackward() {
                    this.flipTo(this.currentPage-1);
                }
            }

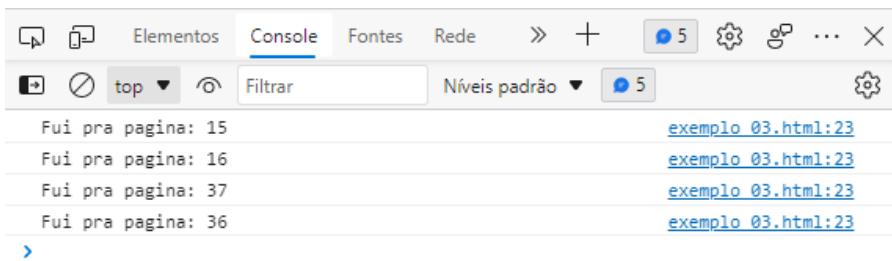
            book.flipTo(15); // Vai para a página 15
            book.turnPageForward(); // Avança para a página 16
            book.flipTo(37); // Vai para a página 37
            book.turnPageBackward(); // Retrocede para a página 36
        </script>

    </body>

</html>
```

No objeto livro, três métodos foram adicionados: turnPageForward, turnPageBackward e flipTo. Cada método fornece alguma funcionalidade ao objeto livro, permitindo que um leitor se mova através das páginas. As partes interessantes deste código são as próprias declarações de função.

Por exemplo, quando você olha o código para a função flipTo, você pode pensar que a função é chamada FlipToAPage porque foi isso que foi declarado. Entretanto, os métodos são chamados usando o pseudônimo de propriedade que atribuiu a função. Ao usar o o código, o tempo de execução sabe que é um método, não uma propriedade, e espera que o método seja chamado com parênteses



Criar objetos em linha como no exemplo do código anterior é útil apenas quando é usado na página onde é definido, e talvez apenas algumas vezes. Entretanto, se você planeja usar um objeto com freqüência, considere a criação de um protótipo para ele. Um protótipo fornece uma definição do objeto para que você possa construir o objeto usando a palavra-chave `new`. Quando um objeto pode ser construído, por exemplo, com a palavra-chave `new`, o construtor pode tomar parâmetros para inicializar o estado do objeto, e o próprio objeto pode, internamente, tomar medidas extras conforme necessário para inicializar-se.

## Dica para exames

O JavaScript consiste em objetos. Tudo em JavaScript é um objeto. Cada objeto é baseado em um protótipo. Sempre que você cria uma nova instância de um objeto, essa instância é baseada no protótipo do objeto.

## aula\_6\exemplo\_04.html

```
<html>

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

    <body>

        <script>

            // construtores

            function Book(){
                //just creates an empty book.
            }

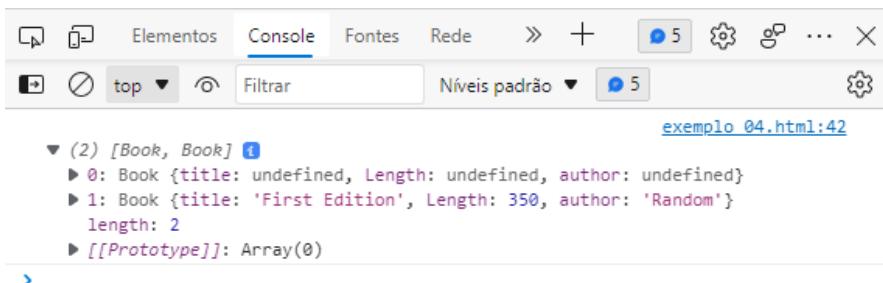
            function Book(title, length, author) {
                this.title = title;
                this.length = length;
                this.author = author;
            }

            // protótipo

            Book.prototype = {
                ISBN: "",
                Length: -1,
                genre: "",
                covering: "",
                author: "",
                currentPage: 0,
                title: "",
                flipTo: function FlipToAPage(pNum) {
                    this.currentPage = pNum;
                },
                turnPageForward: function turnForward() {
                    this.flipTo(this.currentPage++);
                },
                turnPageBackward: function turnBackward() {
                    this.flipTo(this.currentPage--);
                }
            };

            var books = new Array(new Book(), new Book("First Edition",350,"Random"));
            console.log(books);

        </script>
    </body>
</html>
```



Com este novo código, você pode criar um objeto Livro vazio usando o construtor sem parâmetros, ou você pode criar um objeto Livro usando parâmetros específicos para inicializar alguns campos.

Os objetos podem conter outros objetos, conforme a necessidade. Neste exemplo, a propriedade Autor poderia ser facilmente incorporada em um novo protótipo, tornando-o mais extensível e encapsulando informações relacionadas a um autor.

### aula\_6\exemplo\_05.html

```
<html>
  <head>
    <meta charset="utf-8" />
    <title></title>
  </head>

  <body>

    <script>

      function Book(){
        //just creates an empty book.
      }

      function Book(title, length, author) {
        this.title = title;
        this.Length = length;
        this.author = author;
      }

      Book.prototype = {
        ISBN: "",
        Length: -1,
        genre: "",
        covering: "",
        author: new Author(),
        currentPage: 0,
        title: ""
      }

      function Author(){
      }

      function Author(firstName, lastName, gender){
        this.firstName = firstName;
        this.lastName = lastName;
        this.gender = gender;
      }
    </script>
  </body>
</html>
```

```

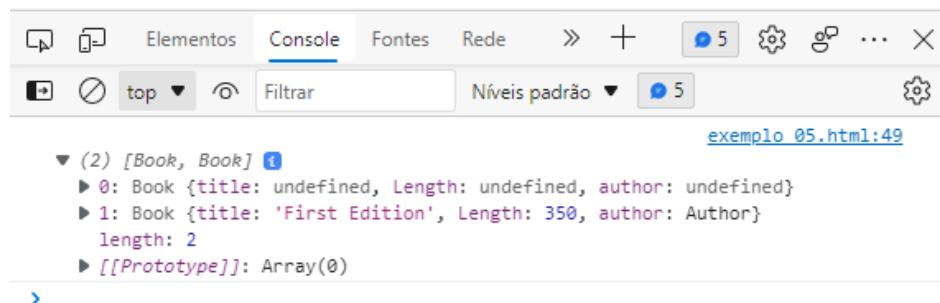
Author.prototype = {
    firstName:"",
    lastName:"",
    gender:"",
    BookCount: 0
}

var books = new Array(new Book(), new Book("First Edition",350, new Author("Roberto","Pinheiro","M")));
console.log(books);

</script>

</body>
</html>

```



Agora, o autor do livro é um objeto personalizado. Isto proporciona mais extensibilidade no projeto. Se você decidir mais tarde que precisa adicionar informações sobre o autor, você pode simplesmente adicionar a propriedade ou propriedades ao protótipo do Autor.

## Dica para exames

Você pode adicionar propriedades a um protótipo dinamicamente em vez de usar o método anterior. O seguinte código produz o mesmo resultado. A utilização de tal código é apenas uma questão de preferência.

```

Book.prototype.ISBN = "";
Book.prototype.Length = 350;
Book.prototype.genre = "";
Book.prototype.cover = "";
Livro.prototype.autor = novo Autor();
Book.prototype.currentPage = 0;
Book.prototype.title = "";

```

## Implementando herança

Em programação orientada a objetos, a herança é um conceito fundamental. Na programação orientada a objetos padrão, as classes são criadas em uma hierarquia relacional, de modo que os atributos e funcionalidades de uma entidade podem ser reutilizadas dentro de outra entidade sem ter que recriar todo o código. Veremos como funciona herança de objetos em JavaScript.

No exemplo de código anterior, você criou um objeto chamado Livro. Mas muitos tipos de livros existem. Para ampliar a definição de Livro, é preciso separar as diferenças de funcionalidade entre, por exemplo, livros pop-up e outros livros. Os livros pop-up têm alguma funcionalidade extra, como a exibição do pop-up na página atual e talvez tocando um som. Em outras palavras, enquanto um livro pop-up "é-um" livro, estas funcionalidades extras não se aplicam a todos os livros. Neste caso, seria útil herdar todos os atributos e comportamentos básicos de um livro sem ter que recriá-los.

Extender o protótipo do Livro é muito parecido com criar um novo protótipo. Você precisa de apenas uma linha de código para dizer ao JavaScript para herdar a funcionalidade e os atributos de outro objeto. Você faz isto inicializando o protótipo para o objeto pai:

```
function PopUpBook() {  
    Book.call(this);  
}  
  
PopUpBook.prototype = Book.prototype;  
PopUpBook.prototype.hasSound = false;  
PopUpBook.prototype.showPopUp = function ShowPop() {};
```

Desta forma, PopUpBook agora estende a implementação do objeto Livro e acrescenta sua funcionalidade própria para reutilização. A função PopUpBook faz uma chamada de método para Book.call(...). Esta é uma chamada ao construtor da super-classe (a classe que está sendo herdada). Se a superclasse tem um construtor que toma parâmetros, este método lhe permitiria passar como parâmetro valores para os construtores de super-classe para inicialização de objetos.

## Resumo

- Tudo em JavaScript é um objeto - até mesmo funções.
- O JavaScript suporta objetos nativos e objetos personalizados (personalizados).
- Os objetos são criados com a palavra-chave new.
- O acesso a métodos e propriedades em objetos é feito com a notação de pontos: object.method ou objeto.propriedade.
- Você pode criar objetos personalizados de forma dinâmica ou usando protótipos.
- Os protótipos permitem a reutilização da definição de objetos, enquanto os objetos dinâmicos requerem atributos e métodos definidos para cada uso.
- A herança é obtida em JavaScript através da extensão de protótipos.

## Objective Review

Answer the following questions to test your knowledge of the information in this objective.

1. In JavaScript, which of the following isn't a native object?

- A. Function
- B. Array
- C. Integer
- D. Person

- A. Incorreto: As funções são objetos em JavaScript.
- B. Incorretos: Os arrays são objetos em JavaScript.
- C. Incorreto: Inteiros são objetos em JavaScript.
- D. Correto: As pessoas não são objetos nativos em JavaScript.

2. Which of the following snippets shows the correct way to create a custom book object?

- A. var book = "Title: 'My book about things'" + "Author: 'Jane Doe'" + " Pages: 400";
- B. var book = {Title: "My book about things", Author: "Jane Doe", Pages: 400};
- C. var book = (Title= "My book about things", Author= "Jane Doe"= Pages: 400);
- D. var book = new {Title: "My book about things", Author: "Jane Doe", Pages: 400};

- A. Incorreto: Este segmento de código produz uma cadeia de caracteres (string).
- B. Correto: Este segmento de código cria um objeto dinâmico.
- C. Incorreto: Este segmento é incorreto porque (...) foi usado em vez de {}, juntamente com = em vez de :.
- D. Incorreto: A palavra-chave new não é usada quando se cria um objeto dinâmico.

3. Inheritance is accomplished in JavaScript through the use of which construct?

- A. inherits keyword
- B. implements keyword
- C. this keyword
- D. Prototypes

- A. Incorreto: A palavra-chave inherits não é uma construção JavaScript.
- B. Incorreto: A palavra-chave implements não é uma construção em JavaScript.
- C. Incorreta: A palavra-chave this não fornece herança.
- D. Correto: Os protótipos são usados para criar árvores sucessórias em JavaScript.

## CAPÍTULO 2 - ESTRUTURAS DE DOCUMENTOS HTML5

Ser capaz de manipular o Document Object Model (DOM), criar animações e usar as várias interfaces de programação de aplicativos (APIs) fornecidas pela biblioteca JavaScript é uma grande habilidade para se ter. Para aproveitar totalmente o poder da experiência do usuário, no entanto, você precisa fornecer aos usuários certas funções do site apenas sob certas condições, um conceito conhecido como fluxo de programa.

Sem o fluxo do programa, os programas JavaScript processariam de cima para baixo na ordem em que o código foi escrito. Isso é útil em alguns casos, mas na maioria das situações em que uma experiência dinâmica do usuário é necessária, a lógica precisa ser processada condicionalmente. O fluxo do programa pode ser condicional, iterativo ou comportamental:

- O fluxo condicional do programa é baseado na avaliação do estado para tomar uma decisão na qual o código deve ser executado.
- Fluxo iterativo é a capacidade de processar listas ou coleções de informações sistematicamente e de forma consistente.
- O fluxo comportamental pode ser definido como um evento ou retorno de chamada em que uma lógica específica deve ser aplicada com base no envolvimento do usuário com o aplicativo da web ou na conclusão de outra tarefa.

O fluxo pode - e quase sempre incluirá - uma combinação dos três.

Outro tipo especial de fluxo do programa envolve o tratamento de exceções. Tratamento de exceções fornecem a capacidade de executar uma lógica específica no caso de um erro no programa.

Objetivos neste capítulo:

- Aula 1: Implementar o fluxo do programa
- Aula 2: Levantar e tratar um evento
- Aula 3: Implementar o tratamento de exceções
- Aula 4: Implementar um callback
- Aula 5: Criar um processo web worker

# Aula 1 - Implementando fluxo de programa

## Objetivos

- Analisar expressões
- Trabalhar com Arrays
- Implementação de tipos especiais
- Usar métodos avançados com Array
- Implementação de controle de fluxo iterativo

## Analisando expressões

### Operadores condicionais

TABLE 2-1 Conditional and logical operators

Operator	Type	Description
>	Conditional	Evaluates whether the value on the left is greater than the value on the right
<	Conditional	Evaluates whether the value on the right is greater than the value on the left
>=,<=	Conditional	Evaluates the same as > or < but with the additional logic that the values can also be equal
!=	Conditional	Evaluates whether the values aren't equal
==	Conditional	Evaluates whether the values are equal independent of the underlying data type
== =	Conditional	Evaluates whether the values are equal both in value and underlying data type
&&	Logical	The AND logical operator, in which the expressions on both sides must evaluate to true
	Logical	The OR logical operator, in which at least one expression on either side must evaluate to true

## aula\_1\exemplo\_01.html

```
<html>

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

<body>

    <div id="elemento1" style="background-color:red">
        Div
    </div>

    <script>

        //igualdade

        var n = 2000, s = '2000';
        console.log(n == s);
        console.log(n === s);

    </script>

</body>

</html>
```



## Dica para exames

Dois operadores condicionais estão disponíveis para verificar a igualdade: `==` (operador de igualdade) e `===` (operador de identidade). A verificação da igualdade com o operador `==` ignorará o tipo de dados subjacente, enquanto o operador de identidade `===` considerará o tipo de dados. Veja o seguinte exemplo:

```
var n = 2000, s = '2000';
alerta(n == s);
alerta(n === s);
```

A primeira expressão, que usa o operador de igualdade, retorna verdadeiro porque a cadeia é lançada a um número para o propósito da avaliação. A segunda expressão, que usa o operador de identidade, avalia como falso porque a "2000" não é igual ao número inteiro 2000.

## **aula\_1\exemplo\_02.html**

```
<html>

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

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

            //condicional

            var userAge = 10, gender = 'M';
            var minimumAge = 11;

            if (userAge > minimumAge) {
                if (gender == 'M') {
                    alert("Gênero masculino\nIdade inferior a 11 anos");
                } else {
                    alert("Gênero feminino\nIdade inferior a 11 anos");
                }
            } else if (gender == 'M') {
                alert("Gênero masculino\nIdade superior a 11 anos");
            } else {
                alert("Idade superior a 11 anos");
            }

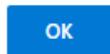
        </script>

    </body>

</html>
```

**localhost diz**

Gênero masculino  
Idade superior a 11 anos



## **aula\_1\exemplo\_03.html**

```
<html>

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

    <body>
        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

            //condicional com elementos

            var canvas = document.getElementById("elemento1");
            if (canvas.style.backgroundColor == 'green' || canvas.style.backgroundColor == 'yellow') {
                console.log('A cor da DIV é verde ou amarelo');
            }
            else if (canvas.style.backgroundColor == 'red') {
                console.log('A cor da DIV é vermelha');
            }

            // condicional com switch

            switch (canvas.style.backgroundColor) {
                case 'yellow':
                    alert('slow down');
                    break;
                case 'green':
                    alert('proceed');
                    break;
                case 'red':
                    alert('stop');
                    break;
                default:
                    alert('unknown condition');
                    break;
            }

            // operador ternário
            canvas.style.backgroundColor == 'green' ? document.write('proceed') : document.write('stop');

        </script>
    </body>
</html>
```

**localhost diz**

stop

OK



## Trabalhando com arrays

aula\_1\exemplo\_04.html

```
<html>
  <head>
    <meta charset="utf-8" />
    <title></title>
  </head>

  <body>

    <div id="elemento1" style="background-color:red">
      Div
    </div>

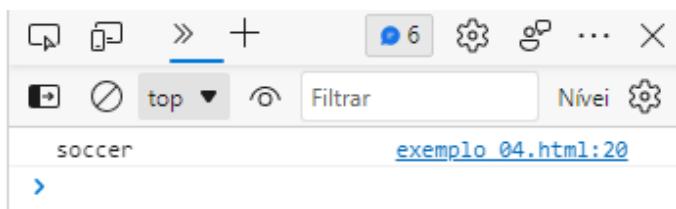
    <script>
      //array

      var anArray = new Array(5);
      anArray[1] = 'soccer';
      console.log(anArray[1]);

    </script>

  </body>

</html>
```



## aula\_1\exemplo\_05.html

```
<html>

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

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

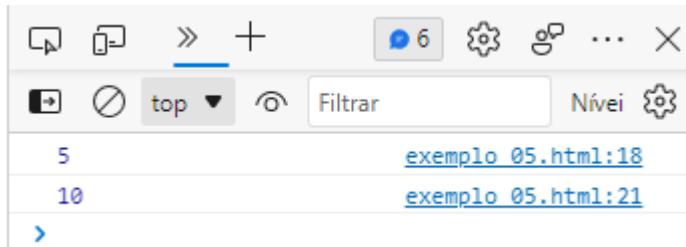
            //array - se você inicialmente alocar apenas 5 mas depois procurar em outro index maior, a array é
            dinamicamente aumentada para acomodar
            var anArray = new Array(5);
            console.log(anArray.length);

            anArray[9] = "soccer";
            console.log(anArray.length);

        </script>

    </body>

</html>
```



## aula\_1\exemplo\_06.html

```
<html>

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

<body>

    <div id="elemento1" style="background-color:red">
        Div
    </div>

    <script>

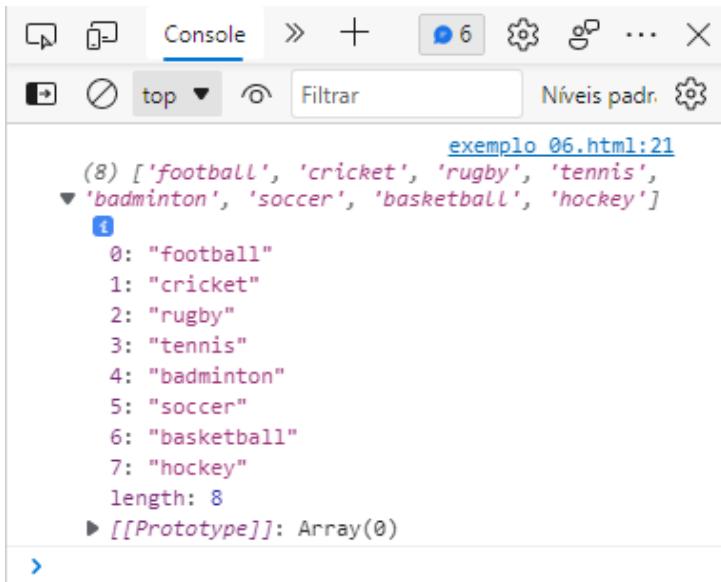
        //array concat

        var sports = new Array('football', 'cricket', 'rugby', 'tennis', 'badminton');
        var moreSports = new Array('soccer', 'basketball', 'hockey');
        var combinedSports = sports.concat(moreSports);
        console.log(combinedSports);

    </script>

</body>

</html>
```



## aula\_1\exemplo\_07.html

```
<html>

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

<body>

    <div id="elemento1" style="background-color:red">
        Div
    </div>

    <script>

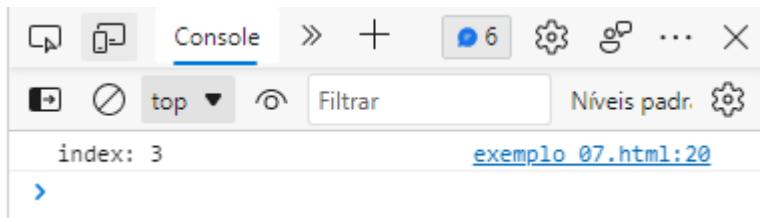
        //indexof

        var sports2 = new Array('soccer', 'basketball', 'hockey', 'football', 'cricket', 'rugby', 'tennis', 'badminton');
        var index = sports2.indexOf('football', 0);
        console.log("index: " + index);

    </script>

</body>

</html>
```



## **aula\_1\exemplo\_08.html**

```
<html>

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

<body>

    <div id="elemento1" style="background-color:red">
        Div
    </div>

    <script>

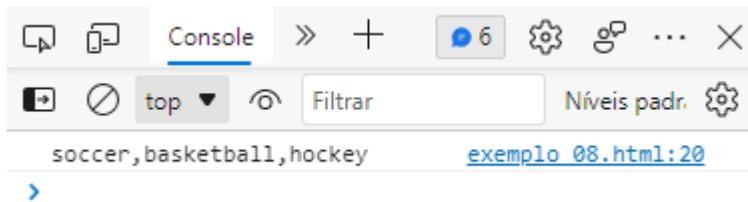
        //Join - O método join coloca todos os elementos em apenas uma string separada pelo separador especificado

        var sports3 = new Array('soccer', 'basketball', 'hockey');
        var joined = sports3.join(',');
        console.log(joined);

    </script>

</body>

</html>
```



## aula\_1\exemplo\_09.html

```
<html>

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

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

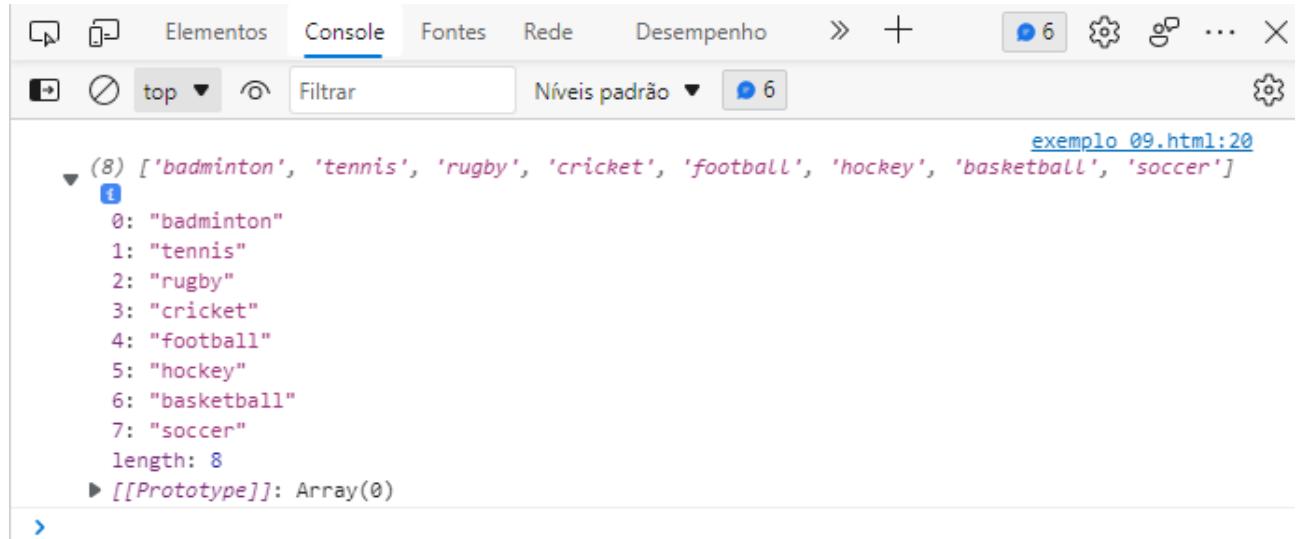
            //Reverse - O método reverse pega todos os elementos e os colocam em ordem reversa

            var sports4 = new Array('soccer', 'basketball', 'hockey', 'football', 'cricket', 'rugby', 'tennis', 'badminton');
            sports4.reverse();
            console.log(sports4);

        </script>

    </body>

</html>
```



## Método Sort

O método sort seqüencia os itens da matriz em ordem ascendente. Na matriz sports, seria por ordem alfabética, como mostra o exemplo a seguir:

**aula\_1\exemplo\_10.html**

```
<html>

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

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

            //Sort - O método sort seqüencia os itens da matriz em ordem alfabética ascendente

            var sports = new Array('soccer', 'basketball', 'hockey', 'football', 'cricket', 'rugby', 'tennis', 'badminton');
            alert(sports.indexOf('soccer'));
            sports.sort();
            console.log(sports);
            alert(sports.indexOf('soccer'));

        </script>

    </body>

</html>
```

**localhost diz**

0

OK

**localhost diz**

6

OK



## Método slice

O método slice retira um ou mais itens de uma matriz e os move para uma nova matriz.

### aula\_1\exemplo\_11.html

```
<html>

    <head>
        <meta charset="utf-8" />
        <title>Método slice</title>
    </head>

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

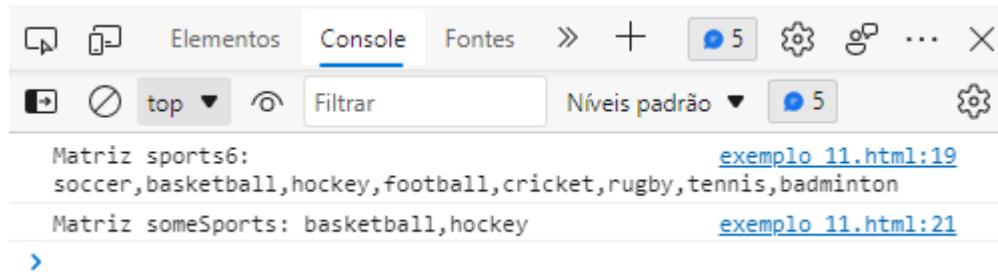
            //slice - O método slice tira um ou mais valores da string e os coloca em outra array.

            var sports6 = new Array('soccer', 'basketball', 'hockey', 'football', 'cricket', 'rugby', 'tennis', 'badminton');
            console.log('Matriz sports6: ' + sports6);
            var someSports = sports6.slice(1, 3);
            console.log('Matriz someSports: ' + someSports);

        </script>

    </body>

</html>
```



## Método splice

O método splice fornece uma maneira de substituir itens em uma matriz por novos itens. O seguinte código demonstra isto:

**aula\_1\exemplo\_12.html**

```
<html>

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

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

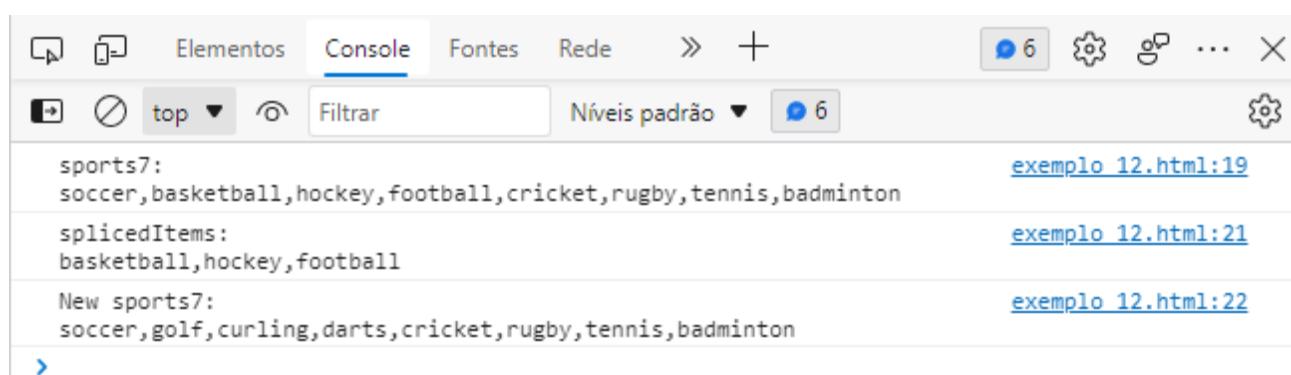
            //splice - método para trocar itens em uma array por novos items

            var sports7 = new Array('soccer', 'basketball', 'hockey', 'football', 'cricket', 'rugby', 'tennis', 'badminton');
            console.log("sports7:\n" + sports7);
            var splicedItems = sports7.splice(1, 3, 'golf', 'curling', 'darts');
            console.log("splicedItems:\n" + splicedItems);
            console.log("New sports7:\n" + sports7);

        </script>

    </body>

</html>
```



## Método push

O método push adiciona um item específico no final de um array.

**aula\_1\exemplo\_13.html**

```
<html>

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

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

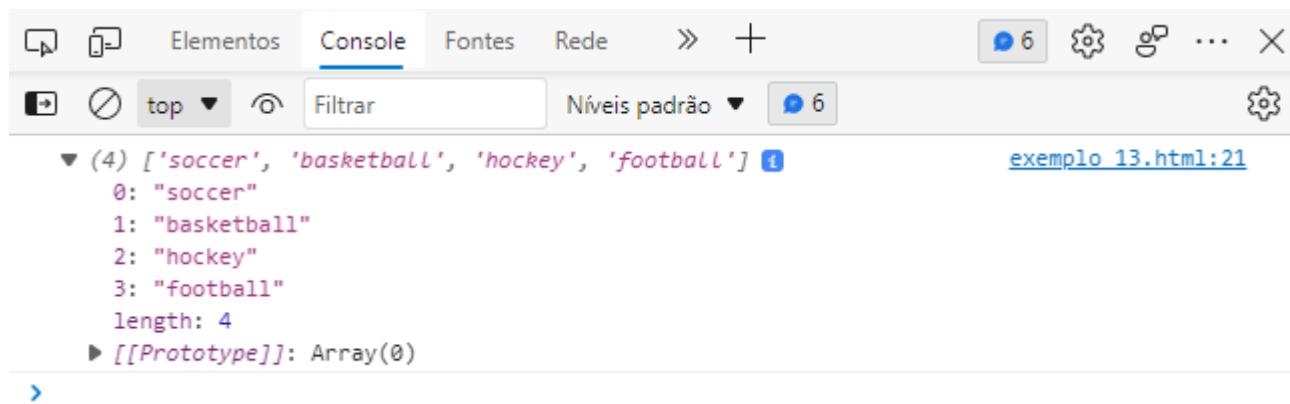
            //push: insere um elemento no final da lista

            var sports8 = new Array();
            sports8.push('soccer', 'basketball', 'hockey');
            sports8.push('football');
            console.log(sports8);

        </script>

    </body>

</html>
```



## Método pop

O método pop remove o último item de um array, retornando o item removido.

### aula\_1\exemplo\_14.html

```
<html>
  <head>
    <meta charset="utf-8" />
    <title></title>
  </head>

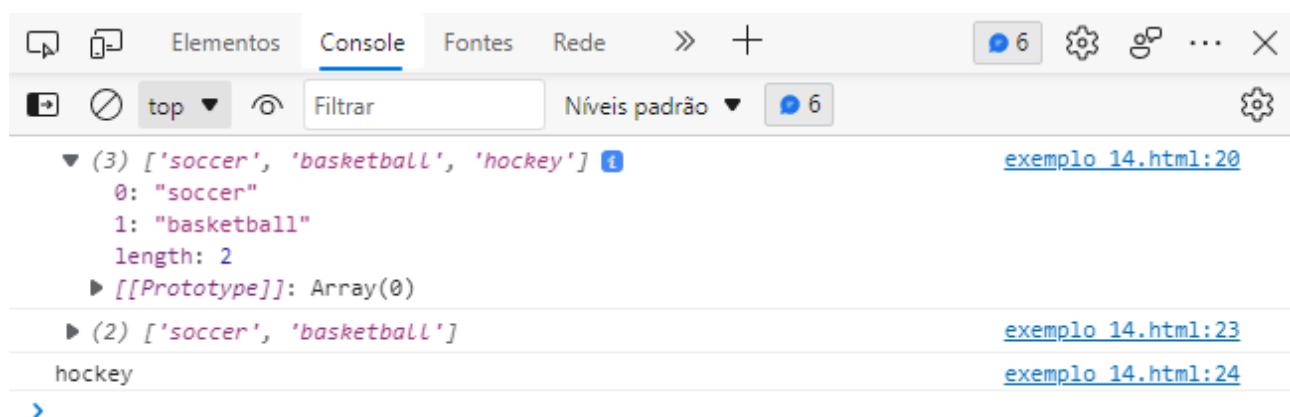
  <body>
    <div id="elemento1" style="background-color:red">
      Div
    </div>

    <script>
      //pop: remove o último item de um array, retornando o item removido

      var sports8 = new Array();
      sports8.push('soccer', 'basketball', 'hockey');
      console.log(sports8);

      var nextSport = sports8.pop();
      console.log(sports8);
      console.log(nextSport);

    </script>
  </body>
</html>
```



The screenshot shows the Chrome DevTools Console tab. The interface includes tabs for Elementos, Console (which is selected), Fontes, Rede, and others. There are also buttons for back, forward, and search, along with a 'Filtrar' (Filter) input field and a 'Níveis padrão' (Default levels) dropdown. The main area displays the execution of JavaScript code and its output:

```
▼ (3) ['soccer', 'basketball', 'hockey'] ⓘ exemplo 14.html:20
  0: "soccer"
  1: "basketball"
  length: 2
  ▶ [[Prototype]]: Array(0)

▶ (2) ['soccer', 'basketball'] exemplo 14.html:23
hockey exemplo 14.html:24
```

The output shows the initial array with three elements: 'soccer', 'basketball', and 'hockey'. After calling `pop()`, the array has two elements: 'soccer' and 'basketball'. The removed element 'hockey' is then printed to the console.

## Método unshift

O método unshift insere um elemento no inicio do array.

### aula\_1\exemplo\_15.html

```
<html>

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

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

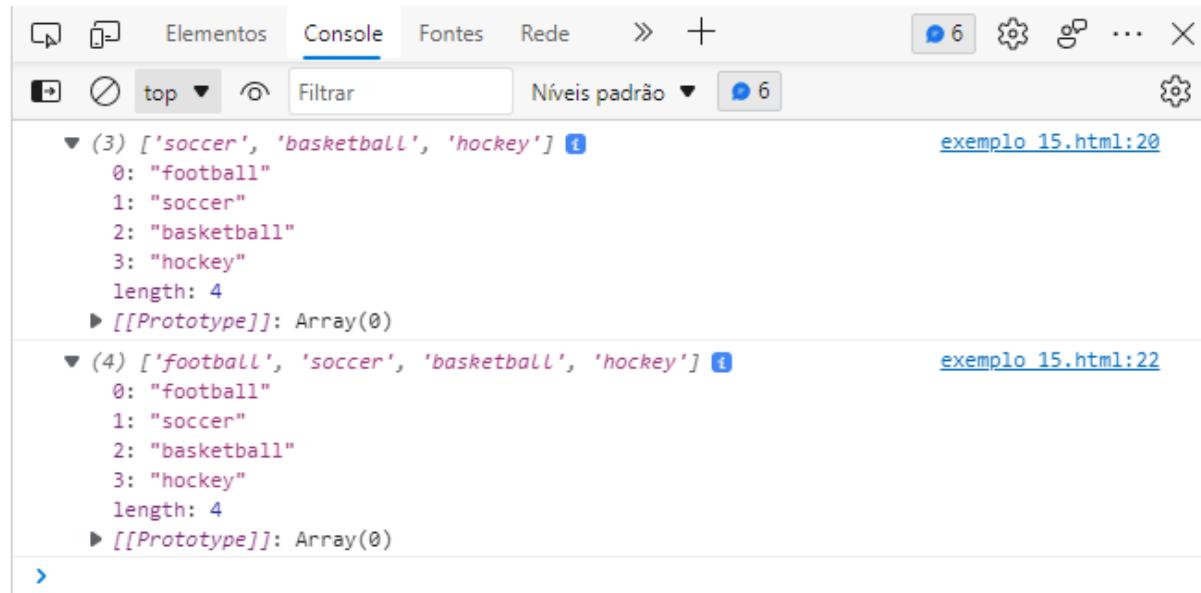
            //unshift : O unshift insere um elemento no inicio do array

            var sports9 = new Array();
            sports9.unshift('soccer', 'basketball', 'hockey');
            console.log(sports9);
            sports9.unshift('football');
            console.log(sports9);

        </script>

    </body>

</html>
```



## Método shift

O método unshift remove o elemento no inicio do array, retornando esse elemento removido.

**aula\_1\exemplo\_16.html**

```
<html>

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

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

            //shift : shift remove e retorna a primeira evidencia em um array.

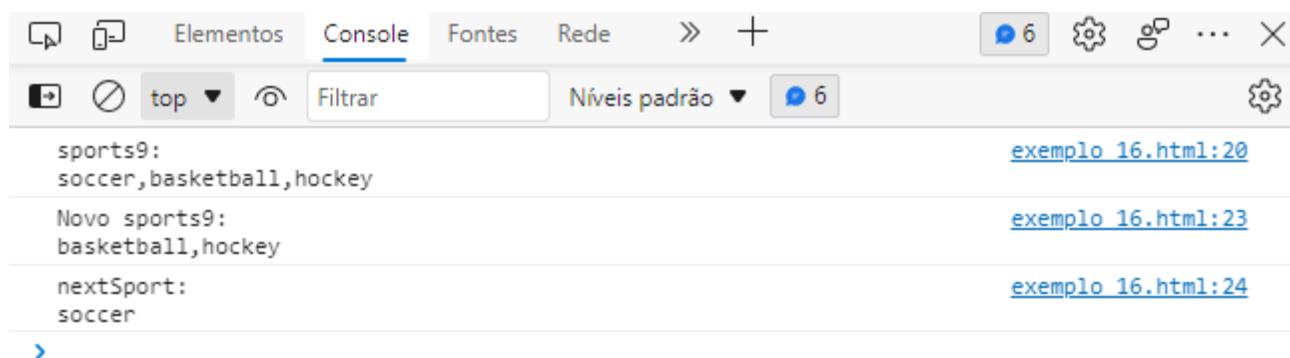
            var sports9 = new Array();
            sports9.unshift('soccer', 'basketball', 'hockey');
            console.log("sports9:\n" + sports9);

            var nextSport = sports9.shift();
            console.log("Novo sports9:\n" + sports9);
            console.log("nextSport:\n" + nextSport);

        </script>

    </body>

</html>
```



## Método every

O método every permite processar uma lógica específica para cada elemento da matriz que atende a alguma condição.

**aula\_1\exemplo\_17.html**

```
<html>

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

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

            //every : Possibilita processar lógica em cada elemento do array através de callbacks de funcoes/metodos

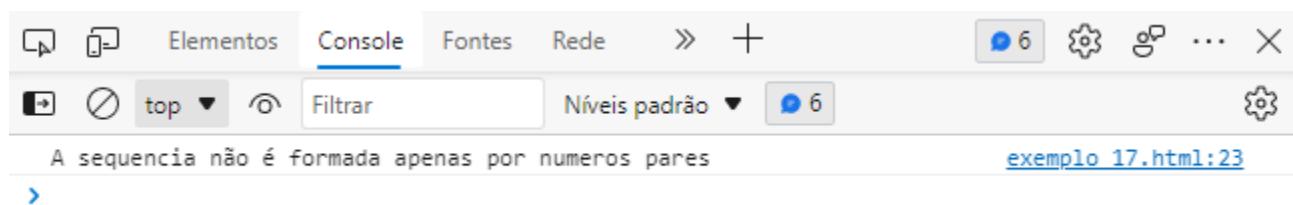
            var evenNumbers = new Array(0, 2, 4, 6, 8, 9, 10, 12);
            var allEven = evenNumbers.every(evenNumberCheck, this);
            if (allEven) {
                console.log("A sequencia é formada apenas por numeros pares");
            } else {
                console.log("A sequencia não é formada apenas por numeros pares");
            }

            function evenNumberCheck(value, index, array) {
                return (value % 2) == 0;
            }

        </script>

    </body>

</html>
```



## Método some

O método verifica se pelo menos um elemento do array satisfaz a condição fornecida.

**aula\_1\exemplo\_18.html**

```
<html>

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

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

            //some: verifica se pelo menos um elemento do array satisfaz a condição fornecida.

            var evenNumbers = new Array(0, 2, 4, 6, 8, 9, 10, 12);
            var leastOne = evenNumbers.some(evenNumberCheck, evenNumbers);

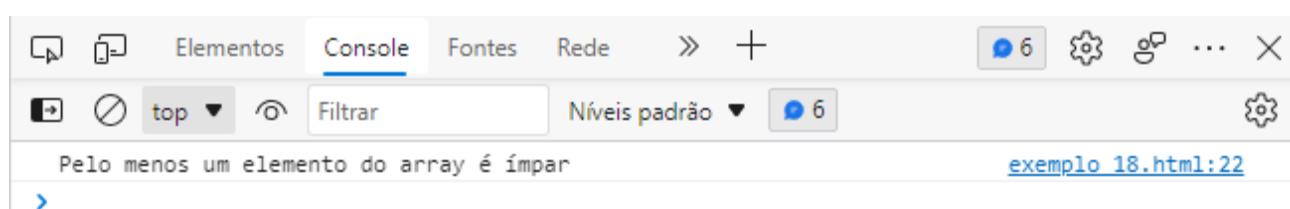
            if (leastOne) {
                console.log("Pelo menos um elemento do array é ímpar");
            } else {
                console.log("Todos os elementos do array são pares");
            }

            function evenNumberCheck(value, index, array) {
                return (value % 2) == 1;
            }

        </script>

    </body>

</html>
```



## Método forEach

O método forEach processa lógica em cada elemento do array/coleção.

### aula\_1\exemplo\_19.html

```
<html>

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

    <body>

        <ul id="sportsList"></ul>

        <script>

            //foreach: processa lógica em cada elemento do array/coleção

            var sportsArray = ['soccer', 'basketball', 'hockey', 'football', 'cricket', 'rugby'];
            sportsArray.forEach(offerSport);

            function offerSport(value, index, array) {
                var sportsList = document.getElementById("sportsList");
                var bullet = document.createElement("li");
                bullet.innerText = value;
                sportsList.appendChild(bullet);
            }

        </script>

    </body>

</html>
```

localhost/html5\_css3/capitulo\_2/aula\_1/exemplo\_19.html

- soccer
- basketball
- hockey
- football
- cricket
- rugby

## Método filter

O método filter filtra itens de um array se o retorno do callback for VERDADEIRO e cria um novo array com o resultado.

**aula\_1\exemplo\_20.html**

```
<html>

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

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

            //filter: filtra itens de uma array se o retorno do callback for VERDADEIRO e cria um novo array com o resultado

            var numbers = new Array(0, 2, 4, 6, 8, 9, 10, 12);
            console.log("numbers: " + numbers);

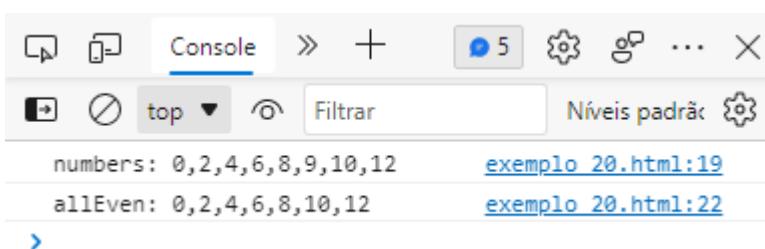
            var allEven = numbers.filter(evenNumberCheck, numbers);
            console.log("allEven: " + allEven);

            //work with the even numbers....
            function evenNumberCheck(value, index, array) {
                return (value % 2) == 0;
            }

        </script>

    </body>

</html>
```



## Método map

O método map possibilita alterar valores em um array. Cada elemento é passado para o método callback e o valor retornado do callback irá substituir o elemento do array.

### aula\_1\exemplo\_21.html

```
<html>

    <head>
        <meta charset="utf-8" />
        <title>Método map</title>
    </head>

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

            //map: cada elemento é passado para o método callback e o valor retornado do callback irá substituir o elemento do array.

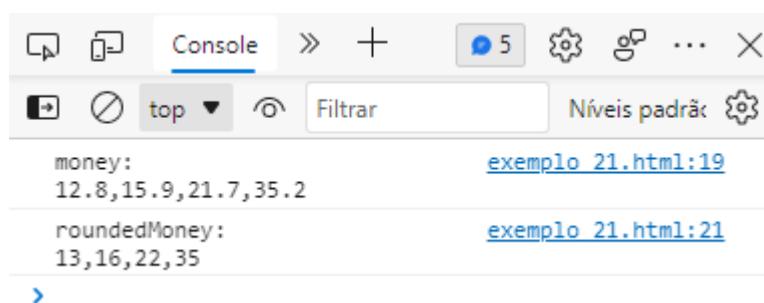
            var money = [12.8, 15.9, 21.7, 35.2];
            console.log("money:\n" + money);
            var roundedMoney = money.map(roundOff, money);
            console.log("roundedMoney:\n" + roundedMoney);

            function roundOff(value, position, array) {
                return Math.round(value);
            }

        </script>

    </body>

</html>
```



## Método reduce e reduceRight

Reduce e ReduceRight são métodos recursivos. O valor resultante do callback é passado de volta pro método que o chamou com o valor novo e antigo

aula\_1\exemplo\_22.html

```
<html>

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

    <body>

        <div id="elemento1" style="background-color:red">
            Div
        </div>

        <script>

            //Reduce e ReduceRight: Métodos recursivos.
            //O valor resultante do callback é passado de volta pro método que o chamou com o valor novo e antigo

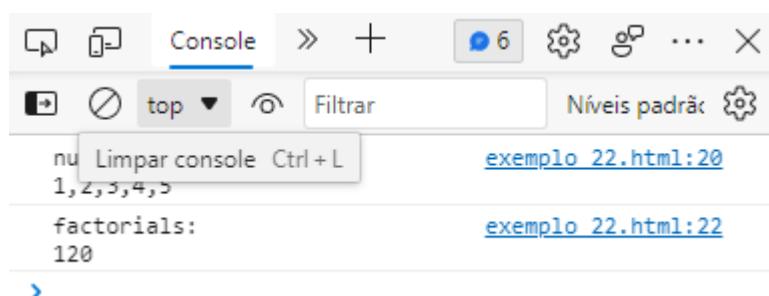
            var numbers = [1, 2, 3, 4, 5];
            console.log("numbers:\n" + numbers);
            var factorials = numbers.reduce(factorial);
            console.log("factorials:\n" + factorials);

            function factorial(previous, current) {
                return previous * current;
            }

        </script>

    </body>

</html>
```



## Implementando controle de fluxo iterativo

### loop for

aula\_1\exemplo\_23.html

```
<html>
```

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

```
  <body>
```

```
    <script>
```

```
      for(var i= 1; i<100;i*=2){
        document.write(i);
        document.write("<br />");
      }
```

```
    </script>
```

```
  </body>
```

```
</html>
```



A screenshot of a web browser window. The address bar shows the URL: `localhost/html5_css3/capitulo_2/aula_1/exemplo_23.html`. The page content displays the following numbers: 1, 2, 4, 8, 16, 32, 64, each on a new line.

```
1  
2  
4  
8  
16  
32  
64
```

## loop for...in

**aula\_1\exemplo\_24.html**

```
<html>

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

    <body>

        <script>

            var person = { firstName: "Jane", lastName: "Doe", birthDate: "Jan 5, 1925", gender: "female"};
            for (var prop in person) {
                document.write(prop + ": " + person[prop]);
                document.write("<br />");
            }

        </script>

    </body>

</html>
```

localhost/html5\_css3/capitulo\_2/aula\_1/exemplo\_24.html

```
firstName: Jane
lastName: Doe
birthDate: Jan 5, 1925
gender: female
```

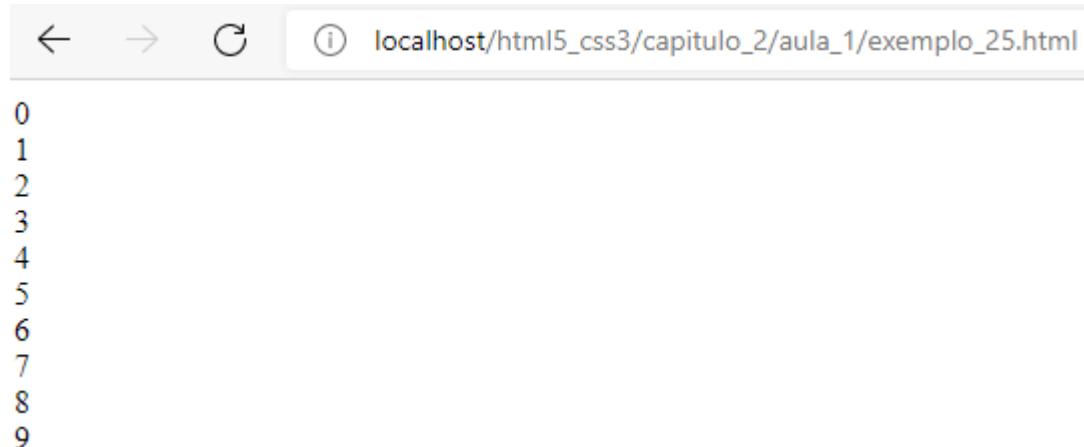
## loop while

**aula\_1\exemplo\_25.html**

```
<html>
  <head>
    <meta charset="utf-8" />
    <title></title>
  </head>

  <body>
    <script>
      var i = 0;

      while (i < 10) {
        document.write(i + "<br />");
        i++;
      }
    </script>
  </body>
</html>
```



## loop do...while

**aula\_1\exemplo\_26.html**

```
<html>

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

    <body>

        <script>

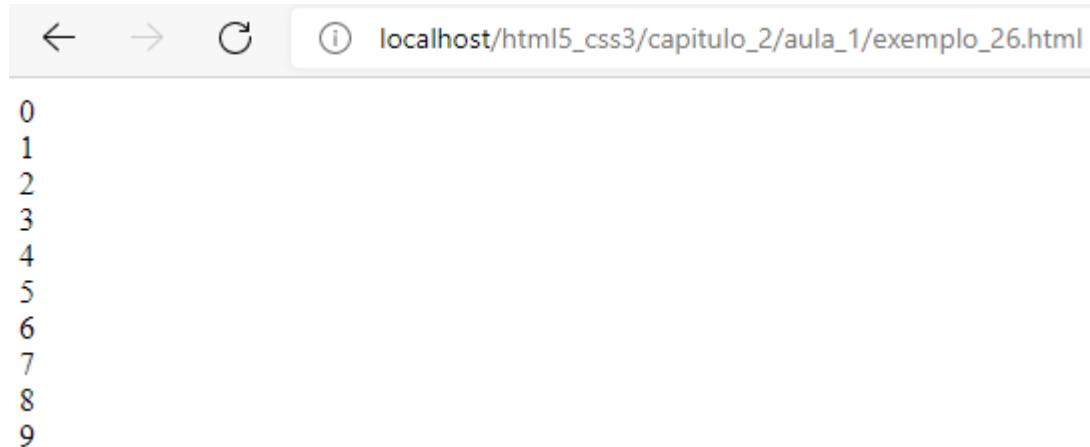
            var i = 0;

            do{
                document.write(i + "<br />");
                i++;
            }while(i < 10)

        </script>

    </body>

</html>
```



## Dica para exames

A palavra-chave break é utilizada apenas no loop que está em execução. Se o laço contendo break é aninhado dentro de outro laço, o laço externo continua a iterar controlado por sua própria expressão.

## Objective Review

1. Which of the following keywords provide iterative control flow?

- A. if statement
- B. switch statement
- C. **for**
- D. break

2. Which of the following array methods combines two arrays?

- A. join
- B. combine
- C. split
- D. **concat**

3. Which iterative control syntax can guarantee that the loop is processed at least once?

- A. for...in loop
- B. while loop
- C. **do...while loop**
- D. for loop

4. Which keyword is used to exit a loop?

- A. continue
- B. **break**
- C. stop
- D. next

## Aula 2 - Criar e gerir eventos

O browser proporciona um comportamento dinâmico através de eventos.

Os eventos normalmente seguem uma convenção de nomenclatura. Alguns eventos comuns são `onkeypress` ou `onblur`. Para que os eventos funcionem, você precisa designar um manipulador de eventos. O manipulador de eventos é uma função JavaScript que é chamado quando uma ação desencadeia o evento.

### Objetivos:

- Usar eventos, incluindo gerenciar o evento por uso de funções anônimas
- Gerenciar eventos DOM (Documento Object Model), ex: `onBlur`, `onFocus`, `onClick` e sua convenção de nomenclatura "on\_" + Nome do evento
- Criar eventos customizados

## Usando eventos

A idéia é dizer ao navegador que quando um determinado evento ocorre, deve ser chamada uma função específica. É necessário atribuir uma função a um evento para quando ele ocorrer.

Você pode realizar um evento de três maneiras:

1. Declarando diretamente na tag HTML.
2. Atribuindo manipuladores de eventos através do JavaScript.
3. Usando os métodos `addEventListener` e `removeEventListener`

Ao designar os manipuladores de eventos através do JavaScript, você tem duas opções: fornecer um função nomeada e atribuir uma função anônima. A diferença entre estes dois será examinado.

### Event object

Em geral, `event object` é um objeto comum disponível nos manipuladores de eventos que fornece metadados sobre o evento. Por exemplo, se os eventos de teclado estão sendo tratados, você pode querer saber qual tecla foi pressionada. Se os eventos do mouse estão sendo tratados, talvez você queira saber qual botão do mouse foi pressionado. O `event object` contém todas estas propriedades.

O `event object` é acessado dentro de uma função de manipulador de eventos, usando o objeto `window` como mostrado a seguir:

```
var evnt = window.event;
```

## Declarando diretamente na tag HTML

aula\_2\exemplo\_01.html

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Declaração do evento na tag HTML</title>

    <script>

      function onloadHandler() {
        console.log("olá enviado pelo evento associado a propriedade do elemento html.");
        alert("olá enviado pelo evento associado a propriedade do elemento html.");
      }

    </script>
  </head>

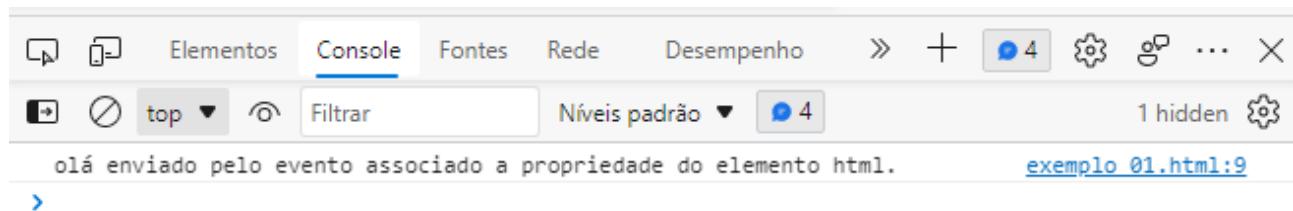
  <body onload='onloadHandler()'>
  </body>

</html>
```

**localhost** diz

olá enviado pelo evento associado a propriedade do elemento html.

OK



## Atribuindo manipuladores de eventos através do JavaScript.

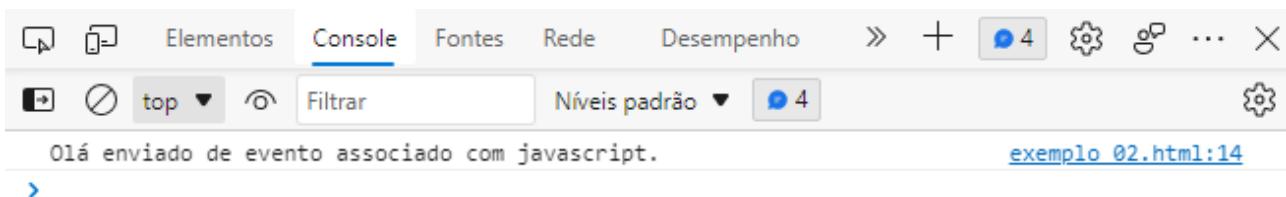
aula\_2\exemplo\_02.html

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Chamando função direto por Javascript</title>

    <script>
      //chamando função direto por Javascript, sem associar ao atributo do elemento
      window.onload = onloadHandler();

      function onloadHandler() {
        console.log("Olá enviado de evento associado com javascript.");
      }
    </script>
  </head>

  <body>
  </body>
</html>
```



## Chamando função anônima direto por Javascript

aula\_2\exemplo\_03.html

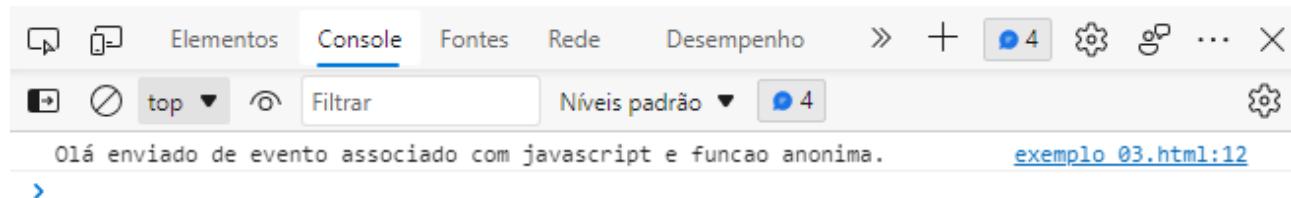
```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Chamando função anônima direto por Javascript</title>

    <script>
      //chamando função direto por Javascript, sem associar ao atributo do elemento

      window.onload = function () {
        console.log("Olá enviado de evento associado com javascript e funcao anonima.");
      }

    </script>
  </head>

  <body>
  </body>
</html>
```



## Usando os métodos addEventListener e removeEventListener

addEventListener e removeEventListener são os dois métodos preferidos para ligar uma função a um evento e depois removê-la mais tarde, conforme necessário.

O método addEventListener aceita dois parâmetros necessários e um opcional

addEventListener(<event name>,<event function>,<optional cascade rule>)

### aula\_2\exemplo\_04.html

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>

    <script>
      //chamando funcoes com addEventListener e removeEventListener

      window.addEventListener("load", onloadHandlerListener, false);
      window.addEventListener("load", onloadHandlerListener2, false);

      function onloadHandlerListener() {
        alert("hello event 1.");
      }

      function onloadHandlerListener2() {
        alert("hello event 2.");
      }

    </script>
  </head>

  <body>
  </body>
</html>
```

**localhost** diz

hello event 1.

OK

**localhost** diz

hello event 2.

OK

## **aula\_2\exemplo\_05.html**

```
<html>

<head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>

    <script>

        //chamando funções com addEventListener e removeEventListener

        window.addEventListener("load", onloadHandlerListener, false);
        window.addEventListener("load", onloadHandlerListener2, false);

        function onloadHandlerListener() {
            alert("hello event 1.");
        }

        function onloadHandlerListener2() {
            alert("hello event 2.");
        }

        window.removeEventListener("load", onloadHandlerListener2, false);

    </script>
</head>

<body>
</body>

</html>
```

**localhost diz**

hello event 1.

OK

## Cancelando um evento

A capacidade de cancelar o processamento de eventos pode ser útil quando se deseja anular completamente a implementação da funcionalidade nativa de um elemento DOM. Um exemplo perfeito é se fosse necessário anular a funcionalidade inerente de um link. Um evento ouvinte seria configurado para o evento de clique. Então, no evento de clique, através do event object, a propriedade `returnValue` seria definida como false ou a própria função pode retornar false.

### aula\_2\exemplo\_06.html

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>

    <script>

      window.onload = function () {
        var a = document.getElementById("aLink");
        a.onclick = OverrideAnchorClick;
      }

      function OverrideAnchorClick() {
        alert("Redirecionamento para o Google cancelado!");
        window.event.returnValue = false;
        //or
        //return false;
      }

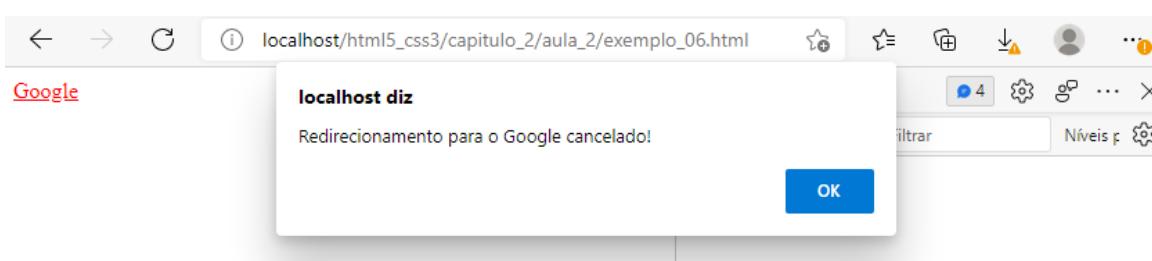
    </script>
  </head>

  <body>

    <a id="aLink" href="http://google.com.br">Google</a>

  </body>

</html>
```



Neste caso, quando o link é clicado, o manipulador de eventos personalizado é executado, mas a navegação normalmente fornecida pelo elemento `<a>` é impedida de ser realizada.

## Declaração e tratamento de eventos bubbled

Event bubbling é o conceito que se aplica quando o documento HTML tem elementos aninhados.

### **aula\_2\exemplo\_08.html**

```
<html>
    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>
        <style>
            #outer {
                width: 200px;
                height: 200px;
                background-color: red;
            }
            #middle {
                width: 50%;
                height: 50%;
                position: relative;
                top: 25%;
                left: 25%;
                background-color: green;
            }
            #inner {
                width: 50%;
                height: 50%;
                position: relative;
                top: 25%;
                left: 25%;
                background-color: blue;
            }
        </style>
    <script>
        window.onload = function () {
            document.getElementById("outer").addEventListener("click", outerDivClick, false);
            document.getElementById("middle").addEventListener("click", middleDivClick, false);
            document.getElementById("inner").addEventListener("click", innerDivClick, false);
            document.getElementById("clearButton").addEventListener("click", clearList);
        }

        function outerDivClick() {
            appendText("outer Div Clicked");
        }

        function middleDivClick() {
            appendText("middle Div Clicked");
        }

        function innerDivClick() {
            appendText("inner Div Clicked");
        }

        function clearList() {
            document.getElementById("list").innerHTML = "";
        }

        function appendText(text) {
            var listElement = document.getElementById("list");
            var listText = document.createTextNode(text);
            listElement.appendChild(listText);
        }
    </script>

```

```

function appendText(s) {
    var li = document.createElement("li");
    li.innerText = s;
    document.getElementById("eventOrder").appendChild(li);
}

function clearList() {
    var ol = document.createElement("ol");
    ol.id = "eventOrder";
    document.getElementById("bod").replaceChild(ol,document.
    getElementById("eventOrder"));
}
</script>

</head>
<body id="bod">

<div id="outer">
    <div id="middle" >
        <div id="inner">
        </div>
    </div>
</div>
<ol id="eventOrder"> </ol>
<button type="button" id="clearButton">Clear</button>
</body>

</html>

```



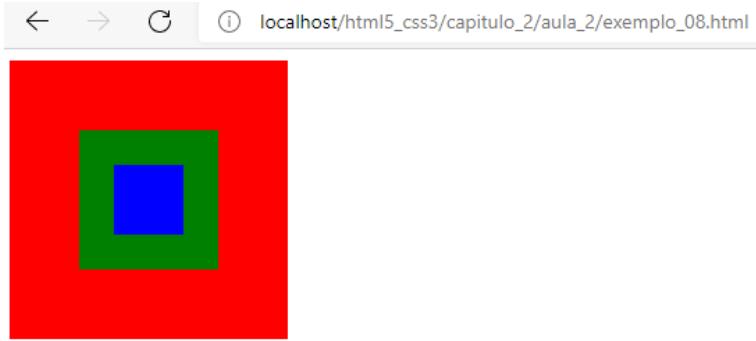
Clique no box vermelho:



1. outer Div Clicked



- Clique no botão "Clear" e em seguida no box verde:



1. middle Div Clicked
2. outer Div Clicked

- Clique no botão "Clear" e em seguida no box azul:



1. inner Div Clicked
2. middle Div Clicked
3. outer Div Clicked

- Faça o processo inverso clicando nos boxes de dentro para fora.

## Manuseio de eventos DOM

O DOM oferece um grande número de eventos integrados. O DOM fornece estes eventos através da API JavaScript. As funções podem ser especificadas como ouvintes de eventos, e o comportamento personalizado pode ser implementado em páginas da web com base no evento que está ocorrendo. Estes eventos se aplicam à maioria dos elementos DOMs.

### Evento change

Um evento change ocorre quando o valor associado a um elemento muda. Isto mais geralmente ocorre em elementos de entrada.

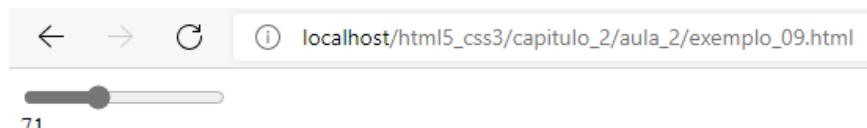
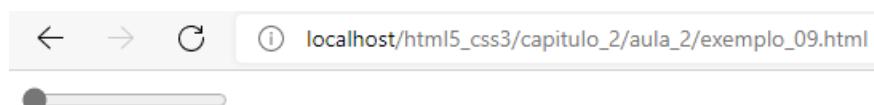
#### aula\_2\exemplo\_09.html

```
<html>
    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>

        <script>
            window.onload = function () {
                document.getElementById("aRange").addEventListener("change", rangeChangeEvent);
            }

            function rangeChangeEvent() {
                document.getElementById("rangeValue").innerText = this.value;
            }
        </script>

    </head>
    <body>
        <input id="aRange" type="range" max="200" min="0" value="0"/>
        <div id="rangeValue"></div>
    </body>
</html>
```



Neste exemplo, ao mover com o mouse o controle da barra deslizante, a div exibe o valor em que o mesmo se encontra (no caso, de 1 a 200).

## Evento focus

TABLE 2-2 The DOM focus events

Event	Description
<i>focus</i>	Raised when the element receives the focus
<i>blur</i>	Raised when the element loses the focus
<i>focusin</i>	Raised just before an element receives the focus
<i>focusout</i>	Raised just before an element loses the focus

### aula\_2\exemplo\_10.html

```
<html>

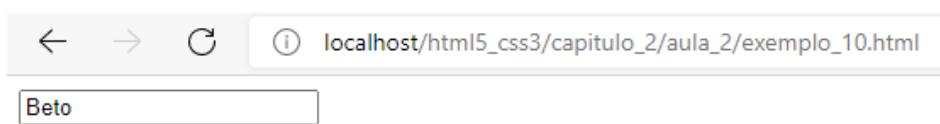
    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>

        <script>
            window.onload = function () {
                document.getElementById("firstNameText").focus();
                document.getElementById("firstNameText").addEventListener("blur", function () {
                    if (this.value.length < 5) {
                        document.getElementById("ruleViolation").innerText = 'First Name is required with least 5 letters';
                        document.getElementById("ruleViolation").style.color = 'red';
                        this.focus();
                    }
                });
            }
        </script>

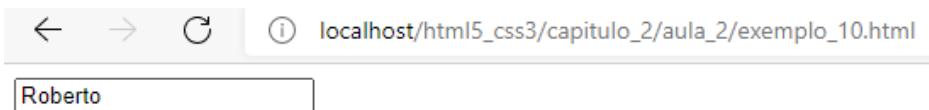
    </head>
    <body>
        <input type="text" id="firstNameText" value=""><br /><br />
        <div id="ruleViolation"></div><br />
    </body>

</html>
```

A página ao ser carregada, fica com o foco na caixa de texto, aguardando a digitação. Se, ao perder o foco, clicando em qualquer outra parte do browser, a caixa de texto tiver menos de 4 caracteres digitados, será exibida uma mensagem de erro. Caso contrário, não.



First Name is required with least 5 letters



## Eventos de teclado

TABLE 2-3 Available keyboard events

Event	Description
<i>keydown</i>	Raised when a key is pushed down
<i>keyup</i>	Raised when a key is released
<i>keypress</i>	Raised when a key is completely pressed

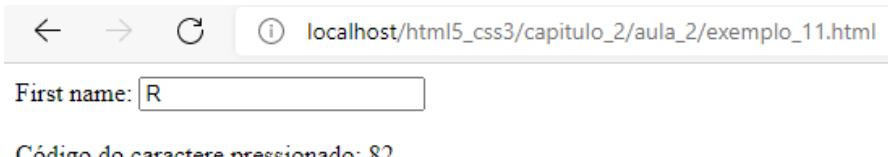
### aula\_2\exemplo\_11.html

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>

    <script>

      window.onload = function () {
        document.getElementById("firstNameText").addEventListener("keydown", function () {
          document.getElementById("message").innerText = "Código do caractere pressionado: " +
window.event.keyCode;
        });
      }

    </script>
  </head>
  <body>
    <label for="firstNameText">First name:</label>
    <input type="text" id="firstNameText" ><br /><br />
    <div id="message"></div><br />
  </body>
</html>
```



Um código como este pode ser usado para filtrar caracteres inválidos de serem inseridos em uma caixa de texto. Propriedades extras de eventos de teclado estão disponíveis no event object para ajudar. Por exemplo, talvez você precise saber se a tecla Shift ou a tecla Control também estava sendo pressionadas. A tabela 2-4 lista as propriedades do event object para eventos de teclado.

**TABLE 2-4** Event object properties for keyboard events

Property	Description
<i>altKey</i>	A Boolean value to indicate whether the Alt key was pressed
<i>keyCode</i>	The numeric code for the key that was pressed
<i>ctrlKey</i>	A Boolean value as to whether the Control key was pressed
<i>shiftKey</i>	A Boolean value as to whether the Shift key was pressed

## Eventos de mouse

**TABLE 2-5** Available mouse events

Event	Description
<i>click</i>	Raised when the mouse performs a click
<i>dblclick</i>	Raised when the mouse performs a double-click
<i>mousedown</i>	Raised when the mouse button is pressed down
<i>mouseup</i>	Raised when the mouse button is released
<i>mouseenter</i> or <i>mouseover</i>	Raised when the mouse cursor enters the space of an HTML element
<i>mouseleave</i>	Raised when the mouse cursor leaves the space of an HTML element
<i>mousemove</i>	Raised when the mouse cursor moves over an HTML element

**TABLE 2-6** Properties of the mouse event

Property	Description
<i>clientX</i>	The x or horizontal position of the mouse cursor relative to the viewport boundaries
<i>clientY</i>	The y or vertical position of the mouse cursor relative to the viewport boundaries
<i>offsetX</i>	The x or horizontal position of the mouse cursor relative to the target element
<i>offsetY</i>	The y or vertical position of the mouse cursor relative to the target element
<i>screenX</i>	The x or horizontal position of the mouse cursor relative to the upper-left corner of the screen
<i>screenY</i>	The y or vertical position of the mouse cursor relative to the upper-left corner of the screen

## **aula\_2\exemplo\_12.html**

```
<html>

<head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>

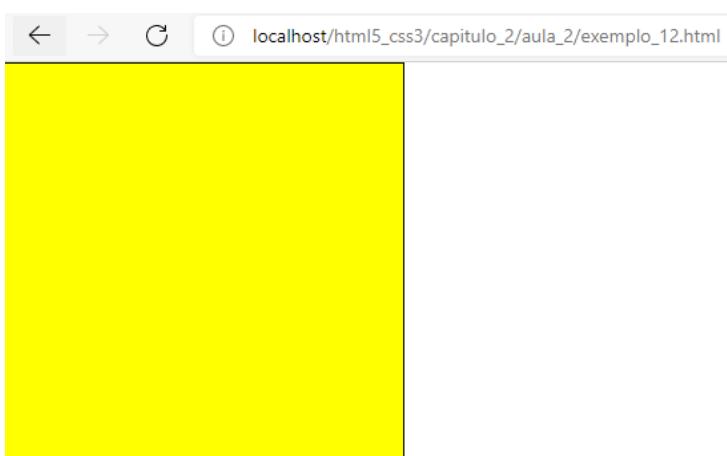
    <style>
        body{
            margin: 0px;
        }
        #yellowBox{
            width: 300px;
            height: 300px;
            border: solid 1px black;
            padding: 0px;
            background-color: yellow;
        }
    </style>
    <script>

        window.onload = function () {
            document.getElementById("yellowBox").addEventListener("click", yellowBoxClick);
        }

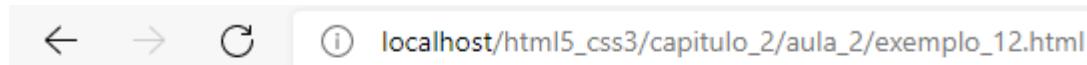
        function yellowBoxClick() {
            document.write("Client X: " + window.event.clientX + " ClientY: " + window.event.clientY);
            document.write("<BR />");
            document.write("offsetX: " + window.event.offsetX + " offsetY: " + window.event.offsetY);
            document.write("<BR />");
            document.write("screen X: " + window.event.screenX + " screenY: " + window.event.screenY);
        }

    </script>
</head>
<body>
    <div id="yellowBox"></div>
</body>

</html>
```



- Ao clicar em alguma parte do box, serão exibidas as coordenadas:



## Eventos mouseenter e mouseleave

Os eventos `mouseenter` e `mouseleave` indicam, respectivamente, quando a posição do cursor do mouse entrou ou saiu da área coberta por um elemento em particular.

### aula\_2\exemplo\_13.html

```
<html>
    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>

        <style>
            .scale {
                transform:scale(1.5);
            }
        </style>
        <script>
            window.onload = function () {
                document.getElementById("yellowBox").addEventListener("mouseenter", yellowBoxEnter);
                document.getElementById("yellowBox").addEventListener("mouseleave", yellowBoxLeave);
            }

            function yellowBoxEnter() {
                this.classList.add("scale");
            }

            function yellowBoxLeave() {
                this.classList.remove("scale");
            }
        </script>
    </head>
    <body>
        <div id="yellowBox" style="width: 50%;height:50%;margin: 0 auto; background-color:yellow;"></div>
    </body>
</html>
```



## Funcionalidade drag-and-drop

TABLE 2-7 Events available to drag and drop

Event	Description
<i>drag</i>	Raised continuously while the element is being dragged
<i>dragend</i>	Raised on the element being dragged when the mouse is released to end the drop operation
<i>dragenter</i>	Raised on a target element when a dragged element is dragged into its space
<i>dragleave</i>	Raised on a target element when a dragged element leaves its space
<i>dragover</i>	Raised continuously on the target element while the dragged element is being dragged over it
<i>dragstart</i>	Raised on the element being dragged when the drag operation is beginning
<i>drop</i>	Raised on the target element when the dragged element is released

## Criando eventos personalizados

Em alguns casos, você pode querer criar um evento personalizado para usar de forma mais genérica. Para criar um evento personalizado, você usa o objeto `CustomEvent`. Para usar eventos personalizados, primeiro é necessário criar uma instância do objeto `CustomEvent`.

```
myEvent = new CustomEvent("anAction",
{
    detail: { description: "a description of the event",
              timeofevent: new Date(),
              eventcode: 2 },
    bubbles: true,
    cancelable: true
});
;
```

O construtor de objetos `CustomEvent` aceita dois parâmetros:

1. O primeiro parâmetro é o nome do evento. Isto é qualquer coisa que faça sentido para o que o evento é suposto representar. Neste exemplo, o evento é chamado de `anAction`.
2. O segundo parâmetro é um objeto dinâmico que contém uma propriedade de detalhe que pode ter propriedades a ele atribuídas contendo informações que devem ser passadas para o manipulador de eventos. Além disso, o parâmetro fornece a capacidade de especificar se o evento deve ser bubble e se o evento pode ser cancelado.

### aula\_2\exemplo\_14.html

```
<html>

    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>

        <script>

            //eventos personalizados

            myEvent = new CustomEvent("anAction",
            {
                detail: {
                    description: "Descrição do evento personalizado",
                    timeofevent: new Date(),
                    eventcode: 2
                },
                bubbles: true,
                cancelable: true
            });
;

            document.addEventListener("anAction", customEventHandler);
            document.dispatchEvent(myEvent);
        </script>
    </head>
    <body>
        <h1>Certificação HTML5</h1>
        <p>Este é o resultado da certificação HTML5.</p>
    </body>
</html>
```

```

function customEventHandler() {
    alert(window.event.detail.timeofevent);
    console.log(window.event.detail.description);
}

</script>
</head>

<body>
</body>

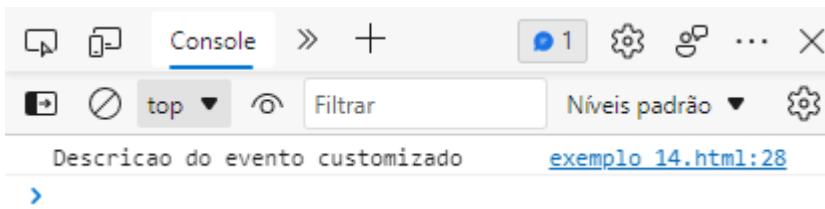
</html>

```

**localhost** diz

Fri Sep 17 2021 13:16:46 GMT-0300 (Horário Padrão de Brasília)

OK



## Resumo

- Os eventos fornecem uma maneira de interagir com os usuários quando eles executam ações sobre a página web.
- Eventos de foco ocorrem quando um objeto recebe ou perde o foco.
- Eventos do teclado ocorrem quando as teclas são pressionadas sobre um objeto focalizado.
- Eventos do mouse ocorrem quando o mouse clica em um objeto ou o ponteiro é movido para cima ou para fora de um objeto.
- A funcionalidade de arrastar e soltar oferece uma maneira de mover elementos de um recipiente para outro

## Objective Review

1. Which of the following isn't a supported way to add an event handler to a DOM element?
  - A. Declaring within the HTML element by assigning the event attribute to a JavaScript function
  - B. Setting the attribute in CSS to a valid JavaScript function
  - C. Dynamically through JavaScript by assigning a JavaScript function to the object's event property
  - D. Dynamically through JavaScript via the assign/remove event listener methods

A. Incorreto: Este é um método válido.  
B. Correto: O CSS não fornece uma maneira de designar manipuladores de eventos.  
C. Incorreto: Este é um método válido.  
D. Incorreto: Este é um método válido.
2. Which of the following isn't an attribute of an anonymous function?
  - A. Anonymous functions can't be called by any other code.
  - B. Anonymous functions have a clearly defined name.
  - C. Anonymous functions can be passed as parameters.
  - D. Anonymous functions can't be assigned to a DOM element declaratively.

A. Incorreto: Funções anônimas não podem ser chamadas. (V)  
B. Incorretas: As funções anônimas não têm nome. (V)  
C. Incorretas: As funções anônimas podem ser passadas como parâmetros. (V)  
D. Correto: Funções anônimas não podem ser atribuídas declarativamente a um elemento DOM (F)
3. Which code line would successfully cancel an event?
  - A. `window.event.returnValue = false;`
  - B. `return false;`
  - C. `window.event.Return();`
  - D. `window.Stop();`

A. Correto: `window.event.returnValue = false;` cancela o evento.  
B. Incorreto: `return false;` não cancela o evento.  
C. Incorreto: `window.event.return();` não é válido.  
D. Incorreto: `window.stop();` não é válido.
4. Which event occurs when a DOM element receives the cursor?
  - A. `focus`
  - B. `change`
  - C. `keydown`
  - D. `mouseleave`

A. Correto: O evento `focus` dispara quando um elemento recebe o foco.  
B. Incorreto: O evento `change` dispara quando o valor de um elemento é alterado.  
C. Incorreto: O evento `keydown` é disparado quando uma tecla de teclado é pressionada.  
D. Incorreto: O evento `mouseleave` dispara quando o ponteiro do mouse deixa a área de um elemento.

5. Which option provides the correct sequence of events in a drag-and-drop operation?

- A. **dragstart, drag, dragenter, drop**
- B. dragstart, drag, dragenter, dragstop
- C. drag, dragstart, drop, dragenter
- D. drag, dragstart, dragenter, dragstop

## Aula 3: Implementar o tratamento de exceções

O JavaScript fornece construções estruturadas de manipulação de erros. O tratamento de erros estruturados em JavaScript é conseguido com a construção try...catch...finally. Uma boa programação defensiva também inclui a verificação de valores nulos para evitar erros. Além de erros de manuseio, o código pode levantar erros personalizados conforme necessário para enviar informações de erro de volta a um programa em execução a partir de objetos personalizados ou bibliotecas.

Objetivos:

- Implementar try-catch-finally, incluindo setar e responder a códigos e lançar exceções
- Checar valores nulos

### Implementando try-catch-finally

**TABLE 2-8** Properties available on the *exception* object

Property	Description
<i>message</i>	A textual description of the error that occurred
<i>number</i>	A numeric error code
<i>name</i>	The name of the <i>exception</i> object

## **aula\_3\exemplo\_01.html**

```
<html>

<head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>

    <style>
        #myCanvas{
            width: 400px;
            height: 400px;
            border: 1px solid black;
        }
    </style>

    <script>

        window.onload = function () {

            try{
                var canvas = document.getElementById("myCanvas");
                var context = canvas.getContext("2d");

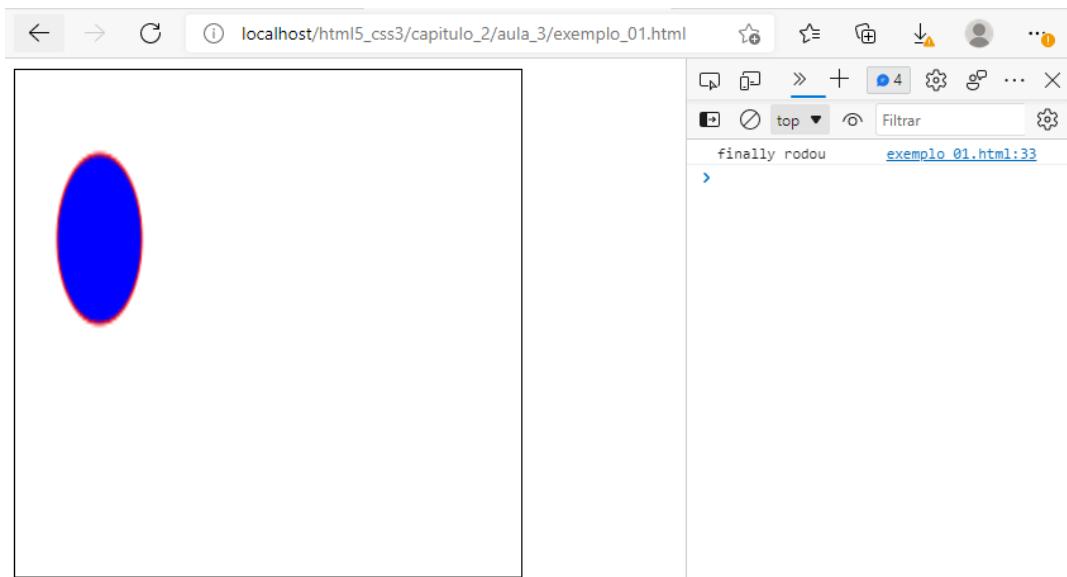
                context.fillStyle = "blue";
                context.arc(50, 50, 25, 0, 360);
                context.fill();
                context.strokeStyle = "red";
                context.stroke();
            }
            catch (e) {
                console.log(e.message);
            }
            finally {
                console.log('finally rodou');
            }
        }

    </script>

</head>

<body>
    <canvas id="myCanvas"></canvas>
</body>

</html>
```



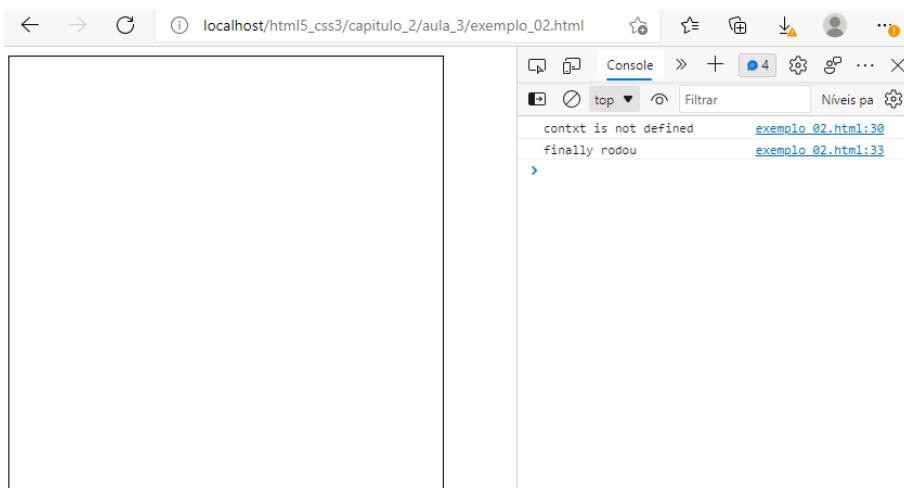
## aula\_3\exemplo\_02.html

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>
    <style>
      #myCanvas{
        width: 400px;
        height: 400px;
        border: 1px solid black;
      }
    </style>
    <script>
      window.onload = function () {

        try{
          var canvas = document.getElementById("myCanvas");
          var context = canvas.getContext("2d");

          context.fillStyle = "blue";
          contxt.arc(50, 50, 25, 0, 360);
          context.fill();
          context.strokeStyle = "red";
          context.stroke();
        }
        catch (e) {
          console.log(e.message);
        }
        finally {
          console.log('finally rodou');
        }

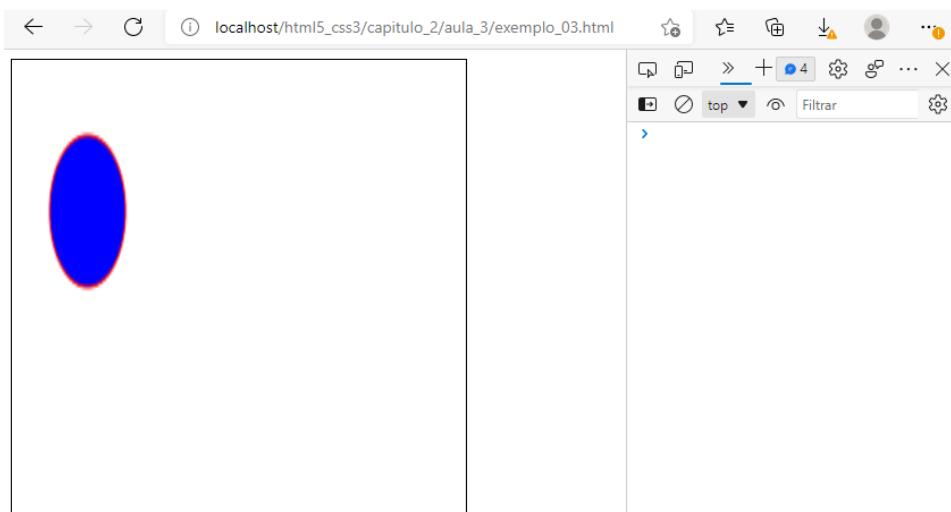
      }
    </script>
  </head>
  <body>
    <canvas id="myCanvas"></canvas>
  </body>
</html>
```



## aula\_3\exemplo\_03.html

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>
    <style>
      #myCanvas{
        width: 400px;
        height: 400px;
        border: 1px solid black;
      }
    </style>
    <script>
      window.onload = function () {
        try {
          WorkWithCanvas();
        } catch (e) {
          console.log(e.message);
        }
      }

      function WorkWithCanvas() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.arc(50, 50, 25, 0, 360);
        context.fillStyle = "blue";
        context.fill();
        context.strokeStyle = "red";
        context.stroke();
      }
    </script>
  </head>
  <body>
    <canvas id="myCanvas"></canvas>
  </body>
</html>
```



## Lançando exceção

**aula\_3\exemplo\_04.html**

```
<html>

<head>
    <meta charset="utf-8" />
    <title>Certificação HTML5</title>

    <style>
        #myCanvas{
            width: 400px;
            height: 400px;
            border: 1px solid black;
        }
    </style>

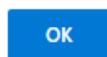
    <script>
        var ball = {
            x: -1,
            y: -1,
            draw: function DrawBall(c) {
                if (this.x < 0)
                    throw new Error("Invalid X coordinate");
            }
        }

        window.onload = function () {
            try {
                var canvas = document.getElementById("myCanvas");
                var context = canvas.getContext("2d");
                ball.draw(context);
            } catch (e) {
                alert("Error: " + e.message);
            }
        }
    </script>
</head>
<body>
    <canvas id="myCanvas"></canvas>
</body>

</html>
```

**localhost diz**

Error: Invalid X coordinate



## Checando valores nulos

Uma maneira de evitar muitos erros é verificar a existência de valores nulos antes de usar algo. Uma variável antes de ser inicializada em um programa JavaScript é um valor nulo. O JavaScript sabe sobre a existência da variável, mas ainda não tem um valor.

Um lugar comum para garantir que as variáveis tenham valores está em funções que aceitam parâmetros.

### aula\_3\exemplo\_05.html

```
<html>

    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>

        <style>
            #myCanvas{
                width: 400px;
                height: 400px;
                border: 1px solid black;
            }
        </style>

        <script>

            window.onload = function () {
                try {
                    var a, b, c;
                    a = 5;
                    b = 10;
                    var result = multiplyNumbers(a, b, c);
                    alert(result);
                } catch (e) {
                    alert(e.message);
                }
            }

            function multiplyNumbers(first, second, third) {
                if (first == null || second == null || third == null)
                {
                    throw new Error("Error: Forgot to initialize a number.");
                }
                return first * second * third;
            }

        </script>
    </head>
    <body>
        <canvas id="myCanvas"></canvas>
    </body>

</html>
```

**localhost** diz

Error: Forgot to initialize a number.

OK

## Resumo

- O tratamento de erros estruturados é fornecido pela linguagem JavaScript na forma de try...catch...finally.
- try...catch...finally fornece uma maneira de tentar alguma lógica, capturar um eventual erro e tratá-lo apropriadamente e, finally, fazer algumas limpezas.
- O bloco finally sempre funciona quer seja ou não lançada uma exceção.
- Verificar um valor nulo antes de acessar quaisquer objetos para garantir que eles sejam inicializados é uma boa prática

## Objective Review

1. Which statement correctly describes the proper error handling using try...catch...finally blocks?

A. Proper error handling allows code processing to continue and to provide appropriate user feedback.

B. Proper error handling allows users to fix problems with the webpage.

C. Proper error handling allows you to debug the application at run time.

D. Proper error handling allows you to suppress all the bugs in your scripts.

A. Correto: Ao utilizar o tratamento estruturado de erros, você pode fornecer feedback aos usuários e lidar adequadamente com situações desconhecidas. (V)

B. Incorreto: O manuseio correto de erros permite aos usuários corrigir problemas com a página web. (F)

C. Incorreto: O tratamento correto de erros permite depurar a aplicação em tempo de execução. (F)

D. Incorreto: O tratamento correto de erros permite que você suprima todos os bugs em seus scripts. (F)

2. Which of the following isn't a property of the exception object?

A. message

B. description

C. number

D. name

A. Incorreto: a mensagem é uma propriedade válida que dá a descrição textual do erro. (V)

B. Correto: a descrição não é uma propriedade válida. (F)

C. Incorreto: A propriedade do número fornece o número associado ao erro. (V)

D. Incorreto: A propriedade nome fornece o nome do objeto de exceção. (V)

3. Why is checking for null a good practice?

A. Checking for null prevents the use of an object before it's initialized.

B. Checking for null prevents errors resulting in NaN.

C. Checking for null prevents the need to throw custom errors

A. Correto: A verificação da nulidade impede o uso de um objeto antes que ele seja inicializado e previne resultados inesperados.

B. Incorreto: O NaN é uma construção diferente da nula.

C. Incorreto: Os erros personalizados não estão relacionados à verificação de nulos.

## Aula 4: Implementar um callback

Os callbacks são um padrão de projeto a ser implementado quando você está trabalhando com múltiplos threads ou apenas precisando ter algo que funcione de forma assíncrona. O conceito de callback é bastante simples e é muito utilizado. A idéia de um callback é chamar uma função para ser executada, mas quando isso for feito, chamar de volta uma função específica com normalmente alguns tipos de resultado ou status da operação.

### Objetivos

- Implementar comunicação bi-direcional com WebSocket API
- Tornar páginas dinâmicas com jQuery e Ajax
- "Ligar" um evento com jQuery
- Implementar call-back com funções anônimas
- Usar o ponteiro this

#### aula\_4\exemplo\_01.html

```
<html>
<head>
<meta charset="utf-8" />
<title>Certificação HTML5</title>

<script>
window.onload = function () {
    WillCallBackWhenDone(MyCallBack, 3, 3);
}

function WillCallBackWhenDone(f, a, b) {
    var r = a * b;
    f(r);
}

function MyCallBack(result) {
    alert("Resultado: " + result);
}
</script>

</head>
<body>

</body>
</html>
```

**localhost** diz

Resultado: 9

OK

## Implementando comunicação bi-direcional com WebSocket API

WebSocket API fornece suporte de comunicação bidirecional para suas aplicações web.

O WebSocket simplificou muito a forma como os dados podem ser enviados e recebidos. Métodos tradicionais, como as longas pesquisas de opinião pública, existem há muito tempo e são hoje amplamente utilizadas em toda a web. Entretanto, as técnicas tradicionais utilizam os mecanismos HTTP mais pesados, o que torna a aplicação inherentemente menos eficiente. O uso do WebSocket API permite a conexão diretamente a um servidor sobre um socket. Esta é uma conexão muito mais leve e é totalmente bidirecional; dados binários e baseados em texto podem ser enviados e recebidos.

### Nota

A implementação completa de um WebSocket API requer que um servidor web tenha um implementação do lado do servidor que aceite conexões de socket. Tecnologias tais como o Node.js funcionam bem para este fim.

O uso do WebSocket API é ideal para aplicações em tempo real, tais como messenger/chat, jogos baseados em servidores e cenários mais avançados, tais como WebRTC (Web Comunicação em Tempo Real) videoconferência. Os dados transmitidos através de WebSockets podem ser binários ou baseados em texto.

**TABLE 2-9** Possible values of the WebSocket *readyState*

Value	Description
OPEN	The connection is open.
CONNECTING	The connection is in the process of connecting and not ready for use yet. This is the default value.
CLOSING	The connection is in the process of being closed.
CLOSED	The connection is closed.

### aula\_4\exemplo\_02.html

```
<html>

    <head>
        <meta charset="utf-8" />
        <title>Certificação HTML5</title>

        <script type="text/javascript">

            window.onload = function (){
                var wsConnection;
                var chatBox = document.getElementById("chatWindow");
                var disconnectButton = document.getElementById("Disconnect");
                var connectButton = document.getElementById("Connect");
                var sendButton = document.getElementById("Send");
                var msgToSend = document.getElementById("msgSendText");
            }
        </script>
    </head>
    <body>
        <div id="chatWindow"></div>
        <button id="Disconnect">Disconnect</button>
        <button id="Connect">Connect</button>
        <input type="button" value="Send" id="Send" />
        <input type="text" value="" id="msgSendText" />
    </body>
</html>
```

```

disconnectButton.onclick = function () {
    wsConnection.close();
}

connectButton.onclick = function () {
    //Or the use of wss for secure WebSockets. IE: wss://studygroup.70480.com
    //Opens the WebSocket
    wsConnection= new WebSocket('ws://studygroup.70480.com', ['soap', 'xmpp']);
}

sendButton.onclick = function () {
    //check the state of the connection
    if (wsConnection.readyState == WebSocket.OPEN) {
        //send message to server.
        wsConnection.send(msgToSend.value);
    }
    else
        return;

    //show message in chat window.
    NewLine();
    chatBox.value = chatBox.value + "You: " + msgToSend.value;
    //clear message text box
    msgToSend.value = "";
}

// event handler for when the WebSocket connection is established
wsConnection.onopen = function () {
    chatBox.textContent = chatBox.textContent +
    "System: Connection has been established";
}

//event handler for when the WebSocket encounters an error
wsConnection.onerror = function (err) {
    //write an error to the screen
    NewLine();
    chatBox.value = chatBox.value + "System: Error Occurred. ";
}

wsConnection.onclose = function () {
    //write the connection has been closed.
    NewLine();
    chatBox.value = chatBox.value + "System: Connection has been closed.";
}

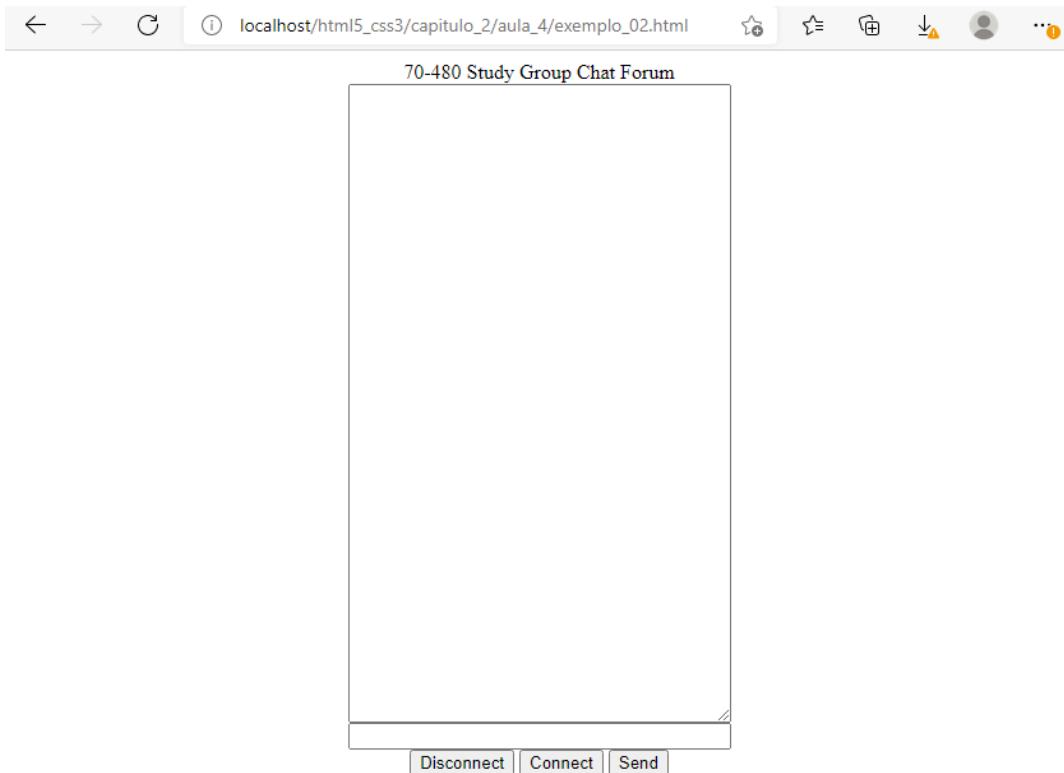
wsConnection.onmessage = function (msg) {
    //write message
    NewLine();
    chatBox.value = chatBox.value + "Them: " + msg.data;
}

//helper functions.
function NewLine()
{
    chatBox.textContent = chatBox.textContent + '\r\n';
}
}

</script>

```

```
</head>
<body>
<div align="center">
<div>
    70-480 Study Group Chat Forum
</div>
<div>
    <textarea id="chatWindow" style="height: 500px; width: 300px">
    </textarea>
</div>
<div>
    <input type="text" id="msgSendText" style="width: 300px"/>
</div>
<div>
    <button id="Disconnect">Disconnect</button>
    <button id="Connect">Connect</button>
    <button id="Send">Send</button>
</div>
</div>
</body>
</html>
```



[https://www.tutorialspoint.com/html5/html5\\_websocket.htm](https://www.tutorialspoint.com/html5/html5_websocket.htm)

**aula\_4\exemplo\_03.html**

```
<!DOCTYPE HTML>

<html>
  <head>

    <script type = "text/javascript">
      function WebSocketTest() {

        if ("WebSocket" in window) {
          alert("WebSocket is supported by your Browser!");

          // Let us open a web socket
          var ws = new WebSocket("ws://localhost:9998/echo");

          ws.onopen = function() {

            // Web Socket is connected, send data using send()
            ws.send("Message to send");
            alert("Message is sent...");

          };

          ws.onmessage = function (evt) {
            var received_msg = evt.data;
            alert("Message is received...");
          };

          ws.onclose = function() {

            // websocket is closed.
            alert("Connection is closed...");
          };
        } else {
          // The browser doesn't support WebSocket
          alert("WebSocket NOT supported by your Browser!");
        }
      }
    </script>

  </head>

  <body>
    <div id = "sse">
      <a href = "javascript:WebSocketTest()">Run WebSocket</a>
    </div>
  </body>
</html>
```

[Run WebSocket](#)

**localhost diz**

WebSocket is supported by your Browser!

OK

**localhost diz**

Connection is closed...

OK

## Páginas dinâmicas com jQuery e Ajax

Até agora, você tem visto algumas grandes maneiras de tornar as páginas web dinâmicas usando JavaScript. JavaScript é a linguagem que o navegador da página web entende. Em alguns casos, usando JavaScript simples ou a biblioteca JavaScript padrão disponível no navegador pode ser incômodo. Aqui é onde jQuery pode ser útil. jQuery é uma biblioteca JavaScript especializada em trabalhar com o DOM para tornar as páginas web dinâmicas.

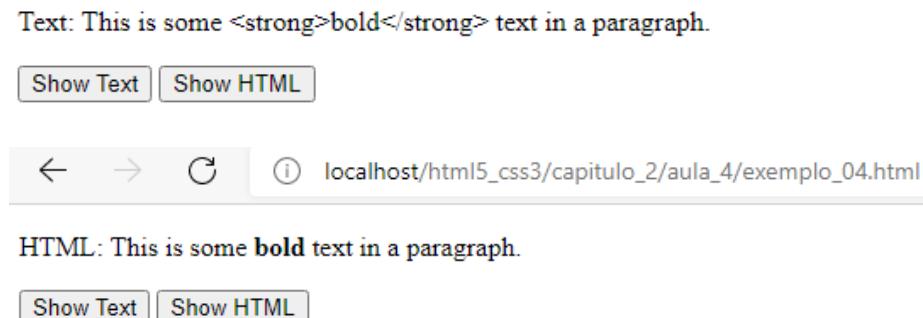
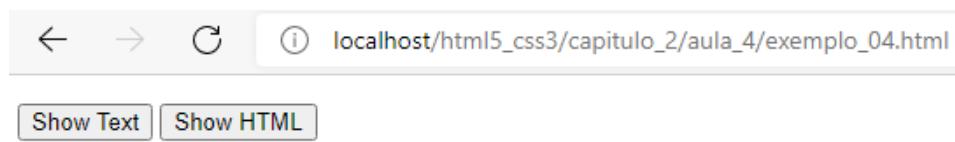
### aula\_4\exemplo\_04.html

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script>
      $(document).ready(function(){
        $("#btn1").click(function(){
          $("#test").text("Text: This is some <strong>bold</strong> text in a paragraph.");
        });
        $("#btn2").click(function(){
          $("#test").html("HTML: This is some <strong>bold</strong> text in a paragraph.");
        });
      });
    </script>
  </head>
  <body>

    <p id="test"></p>

    <button id="btn1">Show Text</button>
    <button id="btn2">Show HTML</button>

  </body>
</html>
```



Veremos como usar jQuery e AJAX para fazer solicitações de servidor para recuperar conteúdo atualizado para suas páginas. No desenvolvimento tradicional da web, quando o conteúdo precisa ser atualizado em uma página, é feita uma solicitação para o servidor da própria página, onde o código do lado do servidor pode ser executado para obter o novo conteúdo, talvez a partir de um banco de dados, e reenviar a página com informações atualizadas. A página inteira precisa ser atualizada. O uso do AJAX solucionou este problema permitindo que você faça solicitações do lado do servidor via JavaScript sem ter que solicitar um atualização de página completa. Você pode implementar o AJAX sem jQuery; entretanto, devido à popularidade e facilidade de uso que jQuery proporciona, usando jQuery para implementar este tipo de funcionalidade é muito mais desejável.

Ao solicitar dados de um servidor com JavaScript via jQuery e AJAX, você pode recuperar dados nos bastidores e depois usar as várias técnicas de manipulação DOM que você tem aprendeu a atualizar áreas específicas da página que precisam ser atualizadas. Isto evita a necessidade de enviar um pedido para a página inteira.

## aula\_4\exemplo\_05.html

```
<html>
  <head>

    <meta charset="utf-8" />
    <title>jQuery com Ajax</title>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script>
        $(document).ready(function(){

            $("#searchButton").click(function(){

                var searchPath;

                switch ($("#searchFruit").val()) {
                    case "long":
                        searchPath = "Fruit/long.xml";
                        break;
                    case "round":
                        searchPath = "Fruit/round.xml";
                        break;
                    case "orange":
                        searchPath = "Fruit/orange.xml";
                        break;
                    default:
                        InvalidSearchTerm();
                }

                $.ajax({
                    url: searchPath,
                    cache: false,
                    dataType: "xml",
                    success: function (data) {
                        $(data).find("fruit").each(
                            function () {
                                $("#searchResults").append($(this).text());
                                $("#searchResults").append("<br />");
                            }
                        );
                    },
                    error: function (xhr, textStatus, errorThrown) {
                        $("#searchResults").append(errorThrown);
                    }
                });

            });
        });

        function InvalidSearchTerm() {
            $("#searchResults").html("");
            $("#searchResults").text("Invalid Search Term. Please try again.");
        }
    });

</script>
</head>
```

```

<body>
  <div>
    Enter search term: <input type="text" id="searchFruit" />
    <button id="searchButton">Search</button>
  </div>

  <h1>Results:</h1>

  <div id="searchResults"></div>

</body>
</html>

```

### **aula\_4\Fruit\long.xml**

```

<?xml version="1.0" encoding="UTF-8"?>
<lista>
  <fruit>
    <nome>orange</nome>
  </fruit>
  <fruit>
    <nome>apple</nome>
  </fruit>
  <fruit>
    <nome>pineapple</nome>
  </fruit>
  <fruit>
    <nome>banana</nome>
  </fruit>
</lista>

```

← → ⌂ ⓘ localhost/html5\_css3/capitulo\_2/aula\_4/exemplo\_05.html

Enter search term:  Search

### **Results:**

← → ⌂ ⓘ localhost/html5\_css3/capitulo\_2/aula\_4/exemplo\_05.html

Enter search term:  Search

### **Results:**

Invalid Search Term. Please try again.

← → ⌂ ⓘ localhost/html5\_css3/capitulo\_2/aula\_4/exemplo\_05.html

Enter search term:  Search

### **Results:**

```

orange
apple
pineapple
banana

```

Nesta listagem, os usuários são apresentados com uma interface de usuário muito simples, na qual eles podem entrar um termo de busca e recuperar um conjunto de resultados baseado nesse termo de busca. Neste caso, os usuários podem digitar um dos termos de busca suportados e obter de volta os dados do servidor. A solicitação de dados é feita usando AJAX e como tal a página inteira não precisa ser atualizada, apenas a área que exibe os resultados. A parte da página onde os dados são necessários é a única parte da página que é afetada pelos novos dados que estão sendo recebidos.

O termo de busca é avaliado para garantir que ele corresponde a um dos termos de busca suportados. Se não corresponder, o usuário é apresentado com uma mensagem indicando isto. Se o fizer, o código procede para fazer uma chamada AJAX para o servidor para o correto conjunto de dados que corresponda ao termo de busca. Neste caso, trata-se de um arquivo XML codificado em código rígido. Entretanto, a fonte de dados é irrelevante desde que o XML retornado corresponda ao esquema que a página web espera para que possa ser analisado e exibido.

## Implementando callback com uma função anônima

As funções de callback são utilizadas em todos os lugares. O conceito de callback é a base de como os eventos trabalham. É o mecanismo pelo qual as operações assíncronas retornam ao chamador.

No JavaScript, as funções são consideradas objetos e são freqüentemente notadas como cidadãos de primeira classe. Isto significa que uma variável pode ser atribuída a uma função, ou uma função pode ser passada para outra função como um parâmetro. Ver as funções utilizadas desta forma é uma convenção comum em JavaScript. As funções utilizadas desta forma são chamadas funções anônimas.

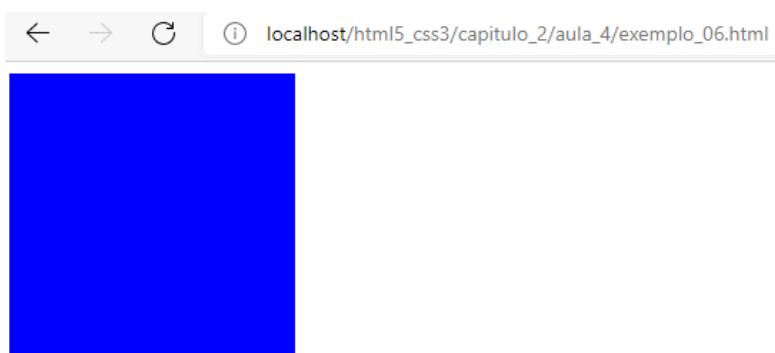
Uma função é considerada anônima quando não tem um nome.

## Usando o ponteiro this

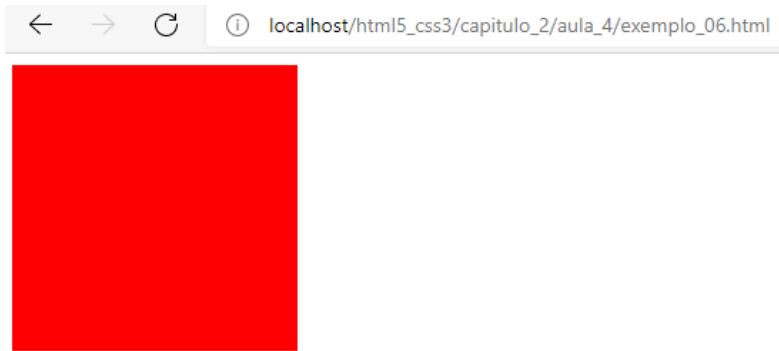
O ponteiro this é um objeto especial fornecido pela estrutura jQuery. Ele fornece um atalho para acessar o item dentro do contexto atual do filtro jQuery.

### aula\_4\exemplo\_06.html

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      #floorDiv{
        width: 200px;
        height: 200px;
        background-color: blue;
      }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script>
      $("document").ready(
        function () {
          $('#floorDiv').click(function () {
            $(this).css("background-color", "red");
          })
        }
      );
    </script>
  </head>
  <body>
    <div id="floorDiv">
    </div>
  </body>
</html>
```



- Clicando no box:



### aula\_4\exemplo\_07.html

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            div{
                width: 200px;
                height: 200px;
                background-color: blue;
                display: inline-block;
                margin-right: 5px;
            }
        </style>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
        <script>
            $("document").ready(
                function () {
                    $('#floorDiv').click(function () {
                        $("div").each(function () {
                            $(this).css("background-color", "red");
                        });
                    })
                }
            );
        </script>
    </head>
    <body>
        <div id="floorDiv"></div>
        <div id="div2"></div>
        <div id="div3"></div>
        <div id="div4"></div>
    </body>
</html>
```



- Clicando no box da esquerda:



## Resumo

- WebSockets fornecem comunicação bidirecional com um servidor.
- WebSockets suportam tanto conexões não seguras (ws) quanto seguras (wss) com o servidor.
- A estrutura jQuery AJAX fornece um mecanismo para fazer requisições web assíncronas.
- Você pode ligar os eventos usando a sintaxe de seletores jQuery.

## Objective Review

1. Which of the following is a valid WebSocket instantiation?

- A. wsConnection = new WebSocket('http://studygroup.70480.com');
- B. wsConnection = new WebSocket('tcp://studygroup.70480.com',['soap','xmpp']);
- C. **wsConnection = new WebSocket('wss://studygroup.70480.com',['soap','xmpp']);**
- D. wsConnection = new WebSocket('ftp://studygroup.70480.com',['soap','xmpp']);

- A. Incorreto: o http não é um protocolo WebSocket válido.
- B. Incorreto: o tcp não é um protocolo WebSocket válido.
- C. Correto: wss ou ws é um protocolo válido para criar um WebSocket.
- D. Incorreto: ftp não é um protocolo válido para WebSocket.

2. Which of the following statements properly handles the reception of data from a WebSocket?

- A. wsConnection.onpost = function(msg){..};
- B. wsConneciton.onreceive = function(msg){...};
- C. **wsConnection.onmessage = function(msg){...};**
- D. wsConnection.ongetdata = function(msg){...};

- A. Incorreto: wsConnection.onpost não é um método.
- B. Incorreto: o wsConneciton.onreceive não é um método.
- C. Correto: o wsConnection.onmessage recebe os dados resultantes.
- D. Incorreto: o wsConnection.ongetdata não é um método.

3. Which list identifies the properties that need to be set up to make an AJAX call?

- A. cache, datatype, success
- B. url, cache, datatype, success
- C. **url, datatype, onsuccess**
- D. url, datatype, oncomplete

- A. Incorreto: o cache não é necessário.
- B. Incorreto: o cache não é necessário.
- C. Correto: url, datatype, e onsuccess são necessários.
- D. Incorreto: oncomplete não é uma propriedade.

4. Why is wiring up events with jQuery easier?

- A. **It allows you to assign the event listener to many elements at once via the selector syntax.**
- B. There is no difference wiring up events with jQuery versus addEventListener method.
- C. jQuery works more efficiently in a loop.
- D. jQuery allows both named and anonymous functions to be used as event listeners.

- A. Correto: A ligação de eventos com jQuery permite que você atribua o ouvinte do evento a muitos elementos ao mesmo tempo, utilizando a sintaxe seletora.
- B. Incorreto: jQuery proporciona muito mais flexibilidade.
- C. Incorreto: jQuery não funciona de forma diferente dentro de um loop.
- D. Incorreto: Isto não é exclusivo de jQuery.

## Aula 5: Criar um processo web worker

Os Web Workers apresentam uma forma de desenvolver aplicações multi-tarefas JavaScript.

Veremos como usar o Web Worker API para tirar proveito da flexibilidade que isso traz às aplicações web. Veremos também sobre as desvantagens e precauções que vêm com o uso de web workers.

### Objetivos

- Iniciar com WebWorker
- Criar um processo WebWorker com o WebWorker API
- Usar um WebWorker
- Entender as limitações do WebWorker
- Configurar timeouts e intervalos

### Iniciando com WebWorker

O código abaixo ao ser executado exibe uma pequena bola saltando dentro de um box. Os usuários podem usar as teclas de seta para mudar a direção da bola.

#### **aula\_5\exemplo\_01.html**

```
<html>

    <head>

        <meta charset="utf-8" />
        <title></title>

        <script>

            window.requestAnimFrame = (function (callback) {
                return window.requestAnimationFrame || window.webkitRequestAnimationFrame ||
window.mozRequestAnimationFrame || window.oRequestAnimationFrame || window.msRequestAnimationFrame ||
function (callback) { window.setTimeout(callback, 1000 / 30); };
            })();

            window.setTimeout(getDirection, 30000);
            var x = 176, y = 176, w = 600, h = 600, r = 26;
            var d, c, s; var rColor, gColor, bColor;
            var hd = "r";
            var horizontal = true;

            window.onload = function () {
                try {
                    c = document.getElementById("c"); w = c.width; h = c.height; s =
parseInt(document.getElementById("speedy").value);
                    getDirection();
                    drawBall();
                }

                document.getElementById("intensiveWork").onclick = function () {
```

```

        DoIntensiveWork();
    };

document.onkeydown = function () {
    switch (window.event.keyCode) {
        case 40: horizontal = false;
            hd = "d";
            break;
        case 37: horizontal = true;
            hd = "l";
            break; case 38: horizontal = false; hd = "u";
            break;
        case 39: horizontal = true; hd = "r";
            break;
    }
}
} catch (e) {
    alert(e.message);
}
}

function increaseSpeed() {
    s++;
    document.getElementById("speedy").value = s;
}

function decreaseSpeed() {
    s--;
    document.getElementById("speedy").value = s;
}

function changeDirection() {
    var cx = window.event.offsetX; var cy = window.event.offsetY; x = cx; y = cy;

    document.getElementById("speedy").value = s;
}

function setNewPoint(d) {

    try {
        switch (horizontal) {
            case true: if (x < (w - r) && hd == "r") x += s;
            else if (x > r && hd == "l") x -= s;
                break;
            case false: if (y < (h - r) && hd == "d") y += s;
            else if (y > r && hd == "u") y -= s;
                break;
        }

        if (x >= (w - r))
            hd = "l";
        if (x <= r)
            hd = "r";
        if (y >= (h - r))
            hd = "u";
    }
}

```

```

        if (y <= r)
            hd = "d";

    } catch (e) {
        alert(e.message);
    }

}

function getDirection() {
    horizontal = !horizontal;
    var d = Math.ceil(Math.random() * 2);

    if (horizontal) {

        if (d == 1) { hd = "r"; }
        else { hd = "l"; }
    } else {
        if (d == 1) {
            hd = "u";
        }
        else {
            hd = "d";
        }
    }
}

function drawBall() {
    try {
        var rgbFill = "rgb(0,0,0)"; var rgbStroke = "rgb(128,128,128)"; setNewPoint(d); var ctxt =
c.getContext("2d");

        ctxt.clearRect(0, 0, c.width, c.height);
        ctxt.beginPath();
        ctxt.lineWidth = "5";
        ctxt.strokeStyle = rgbStroke;
        ctxt.arc(x, y, r, 0, 360);
        ctxt.fillStyle = rgbFill;
        ctxt.fill();
        ctxt.stroke();
        s = parseInt(document.getElementById("speedy").value);

        requestAnimFrame(function () {
            drawBall();
        });
    } catch (e) {
        alert(e.message);
    }
}

function DoIntensiveWork() {
    var result = document.getElementById("workResult");
    result.innerText = "";
    var work = 10000000; var i = 0; var a = new Array(work);
    var sum = 0;

    for (i = 0; i < work; i++) {
        a[i] = i * i
    }
}

```

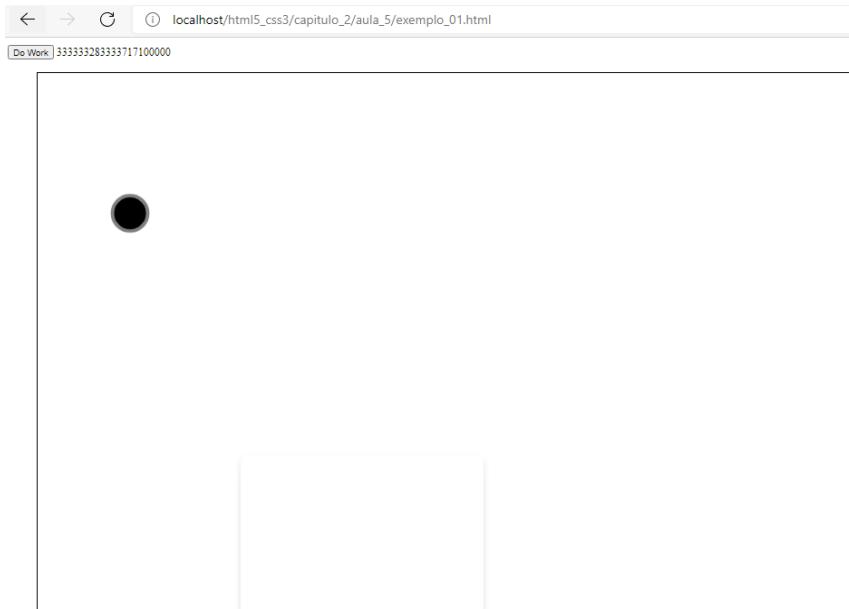
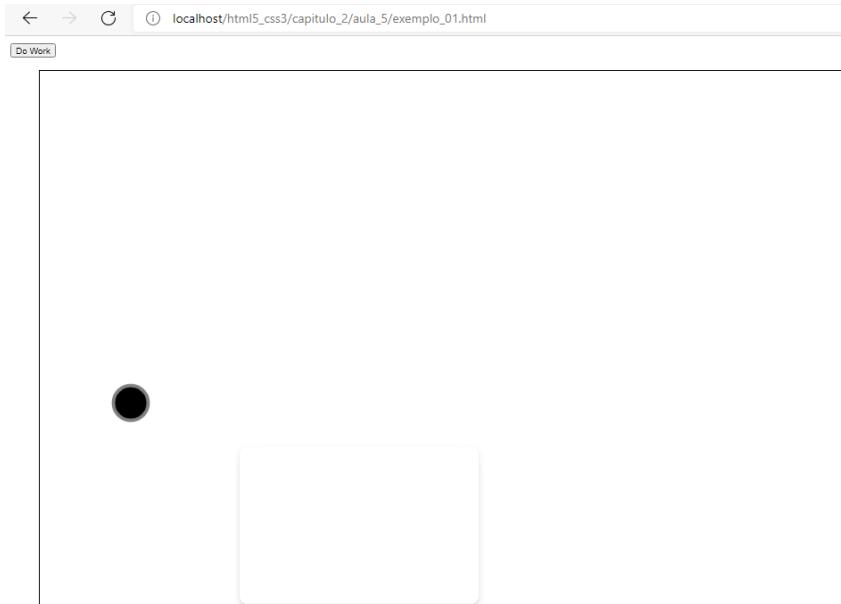
```

        sum += i * i;
    }
    result.innerText = sum;
}
</script>
</head>

<body>
<canvas id="c" width="1200" height="800" style="border: 2px solid black; position: absolute; top: 50px; left: 50px;"></canvas>
<input id="intensiveWork" type="button" value="Do Work" />
<span id="workResult"></span>
<input id="speedy" type="range" min="0" max="10" value="10" style="position: relative; visibility: hidden;" step="1" />
</body>

</html>

```



## Criando um processo WebWorker com o WebWorker API

O Web Worker API é baseado na estrutura de mensagens JavaScript. Esta estrutura permite que seu código envie parâmetros a um worker e que o worker envie resultados. Um web worker básico é estabelecido criando um arquivo separado para conter o roteiro que serão processados na linha separada.

O objeto worker é criado dessa forma:

```
var webWorker = new Worker("workercode.js");
```

**TABLE 2-10** Worker object operations

Method	Description
<i>postMessage</i>	Starts the worker process. This method expects a single parameter containing the data to pass to the worker thread. If nothing is required in the worker thread, an empty string can be supplied.
<i>terminate</i>	Stops the worker process from continuing.
<i>onmessage</i>	Specifies the function for the worker thread to call back to when complete. This function accepts a single parameter in the form of <i>EventData</i> with a property named <i>data</i> containing the values.

Method	Description
<i>onerror</i>	Specifies a function to call when an error occurs in the worker thread. The <i>onerror</i> method receives event data, including the following: <i>message</i> : textual message of the error <i>filename</i> : the filename the error occurred in <i>lineno</i> : the line number in the file that created the error

Assim que o objeto Worker é instanciado, ele fica disponível para ser usado a qualquer momento. Assim já será possível iniciar o processo e chamar o método postMessage:

```
webWorker.postMessage("");
```

Assim que o webWorker estiver funcionando, a aplicação principal continua como de costume. Para cancelar o worker process, deve ser feita uma chamada para o método terminate:

```
webWorker.terminate();
```

Depois que o worker process for concluído e os resultados precisarem serem processados, a função onmessage é chamada do worker.

```
webWorker.onmessage = function(evt) {...}
```

Em seguida, você precisa criar o próprio código do worker. Crie o arquivo workercode.js que foi usado no construtor. A primeira linha do arquivo será o arquivo de uma mensagem a ser atribuída uma função a ser processada:

```
onmessage = função(e){...}
```

Isto diz ao tempo de execução o ponto de entrada para o trabalho a ser executado dentro do worker process. Em algum lugar do worker process, onde um resultado deve ser enviado de volta para a aplicação de chamada, o método `postMessage` é chamado:

```
onmessage = função(e){  
...  
    self.postMessage(resultado);  
}  
}
```

Isso é o que está envolvido na criação de um worker process.

aula\_5\exemplo\_02.html

```
<html>

<head>

<meta charset="utf-8" />

<script>

    window.requestAnimFrame = (function (callback) {
        return window.requestAnimationFrame || window.webkitRequestAnimationFrame ||
window.mozRequestAnimationFrame || window.oRequestAnimationFrame || window.msRequestAnimationFrame ||
function (callback) { window.setTimeout(callback, 1000 / 30); };
    })();

    window.setTimeout(getDirection, 30000);
    var x = 176, y = 176, w = 600, h = 600, r = 26;
    var d, c, s; var rColor, gColor, bColor;
    var hd = "r";
    var horizontal = true;

    window.onload = function () {
        try {
            c = document.getElementById("c"); w = c.width; h = c.height; s =
parseInt(document.getElementById("speedy").value);
            getDirection();
            drawBall();
        }

        // document.getElementById("intensiveWork").onclick = function () { DoIntensiveWork(); };

        document.getElementById("intensiveWork").onclick = function () {
            var result = document.getElementById("workResult");
            result.innerText = "";
            var worker = new Worker("CalculateWorker.js");
            worker.onmessage = function (evt) {
                try {
                    result.innerText = evt.data;
                } catch (e) {
                    alert(e.message);
                }
            }
            worker.onerror = function (err) {
                alert(err.message + err.filename + err.lineno);
            }
            worker.postMessage("");
        }
    }
}
```

```

        document.getElementById("stopWorker").onclick = function () { worker.terminate();}

};

document.onkeydown = function () {

    switch (window.event.keyCode) {
        case 40: horizontal = false;
            hd = "d";
            break;
        case 37: horizontal = true;
            hd = "l";
            break; case 38: horizontal = false; hd = "u";
            break;
        case 39: horizontal = true; hd = "r";
            break;
    }
}

} catch (e) {
    alert(e.message);
}
}

function increaseSpeed() {
    s++;
    document.getElementById("speedy").value = s;
}

function decreaseSpeed() {

    s--;
    document.getElementById("speedy").value = s;
}

function changeDirection() {
    var cx = window.event.offsetX; var cy = window.event.offsetY; x = cx; y = cy;

    document.getElementById("speedy").value = s;
}

function setNewPoint(d) {

try {
    switch (horizontal) {
        case true: if (x < (w - r) && hd == "r") x += s;
        else if (x > r && hd == "l") x -= s;
            break;
        case false: if (y < (h - r) && hd == "d") y += s;
        else if (y > r && hd == "u") y -= s;
            break;
    }

    if (x >= (w - r))
        hd = "l";
    if (x <= r)
        hd = "r";
    if (y >= (h - r))
        hd = "u";
    if (y <= r)
        hd = "d";
}
}

```

```

} catch (e) {
    alert(e.message);
}

}

function getDirection() {
    horizontal = !horizontal;
    var d = Math.ceil(Math.random() * 2);

    if (horizontal) {

        if (d == 1) { hd = "r"; }
        else { hd = "l"; }
    } else {
        if (d == 1) {
            hd = "u";
        }
        else {
            hd = "d";
        }
    }
}

function drawBall() {
    try {
        var rgbFill = "rgb(0,0,0)"; var rgbStroke = "rgb(128,128,128)"; setNewPoint(d); var ctxt = c.getContext("2d");

        ctxt.clearRect(0, 0, c.width, c.height);
        ctxt.beginPath();
        ctxt.lineWidth = "5";
        ctxt.strokeStyle = rgbStroke;
        ctxt.arc(x, y, r, 0, 360);
        ctxt.fillStyle = rgbFill;
        ctxt.fill();
        ctxt.stroke();
        s = parseInt(document.getElementById("speedy").value);

        requestAnimationFrame(function () {
            drawBall();
        });
    } catch (e) {
        alert(e.message);
    }
}

function DoIntensiveWork() {
    var result = document.getElementById("workResult");
    result.innerText = "";
    var work = 10000000; var i = 0; var a = new Array(work);
    var sum = 0;

    for (i = 0; i < work; i++) {
        a[i] = i * i
        sum += i * i;
    }
    result.innerText = sum;
}

```

```

    }

```

```

</script>
</head>

<body>
  <canvas id="c" width="1200" height="800" style="border: 2px solid black; position: absolute; top: 50px; left: 50px;"></canvas>
  <input id="intensiveWork" type="button" value="Do Work" />
  <span id="workResult"></span>
  <input id="speedy" type="range" min="0" max="10" value="10" style="position: relative; visibility: hidden;" step="1"
/></body>
  <input id="stopWorker" type="button" value="Stop Work" />

</html>

```

## Usando web workers

Agora que já viu a criação de web workers, você pode voltar ao exemplo da bola saltadora e passar as operações matemáticas intensivas para um worker process, para que ele não interfira com a atividade da bola saltadora. Para fazer isso, crie um novo arquivo JavaScript chamado [CalculateWorker.js](#) com o seguinte código:

### aula\_5\CalculateWorker.js

```

onmessage = function (evt) {
  var work = 10000000;
  var i = 0;
  var a = new Array(work);
  var sum = 0;

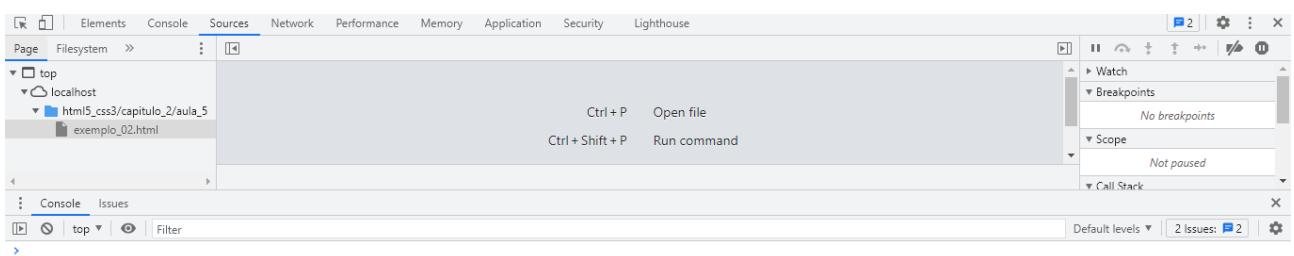
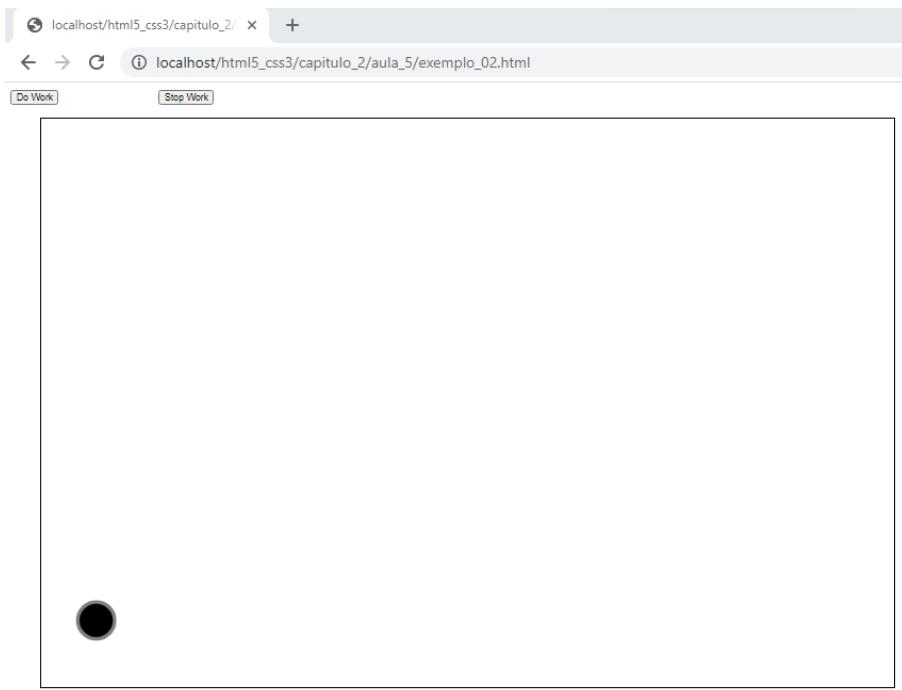
  for (i = 0; i < work; i++) {
    a[i] = i * i;
    sum += i * i;
  }

  self.postMessage(sum);
}

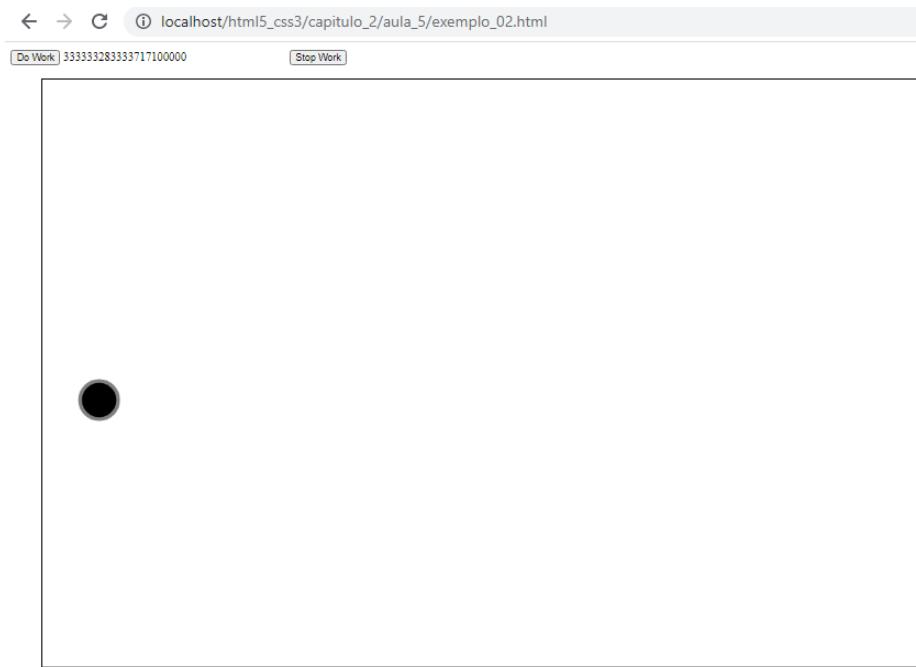
```

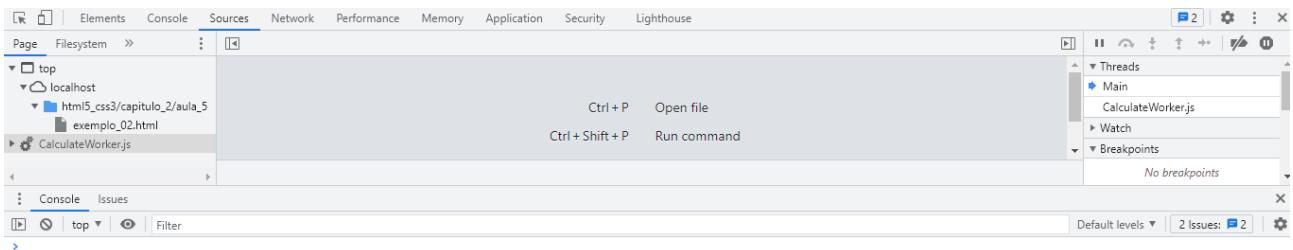
Este código começa com a atribuição ao manipulador de mensagens de uma função a ser executada quando estiver dentro do contexto de um worker. No final da mensagem, [postMessage](#) retorna um resultado de volta à pessoa que o chamou.

Execute este código, clicando no botão [Do Work](#). A bola continua a se mover na tela e é sensível às setas do teclado. Para que o worker process demore mais tempo, basta aumentar o tamanho do número com o qual ele precisa trabalhar.

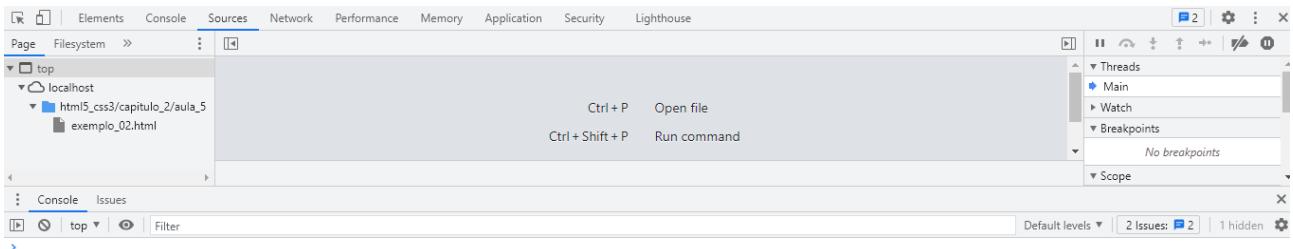


- Clicando no botão "Do work"





- Clicando no botão "Stop worker"



## Entendendo as limitações dos web workers

Os Web Workers são muito convenientes. Eles podem resolver muitos problemas de aplicativos com processamento intensivo na web. Entretanto, esteja ciente também das limitações impostas aos workers.

### Passagem de parâmetros

O método `postMessage` aceita um parâmetro que lhe permite passar ao worker dados. O parâmetro `postMessage` é uma string que pode transmitir qualquer objeto serializável, como tipos de dados nativos, objetos JSON, ou XML. O parâmetro não pode ser um função.

### Número de workers

Embora nenhum limite seja imposto ao número de workers que podem ser processados ou criados ao mesmo tempo, o número de workers utilizados é algo a que você precisa prestar atenção. Criar workers é uma operação pesada.

### Acesso DOM

Os workers operam em seu próprio contexto global, o que significa que eles não têm acesso ao DOM da página que os invocou. O DOM não deve ser manipulado a partir de um worker process. O contexto do worker não tem acesso ao objeto `window`, ao objeto `document` ou a qualquer objeto.

### Configurando timeouts e intervalos

Você pode construir um web worker para ser executado em um intervalo especificado em segundo plano. Isto é feito usando qualquer método `setTimeout` ou `setInterval` existente. O método `setTimeout` chama um método ou função após o atraso especificado. O `setInterval` chama a função especificada repetidamente após cada intervalo de tempo especificado. Por exemplo, o seguinte código executa o worker após 3 segundos:

```
var work = novo Operário ("workerFile.js");
setTimeout(function(){
    work.postMessage("");
},3000);
```

### Resumo

- Os Web Workers permitem que o Javascript forneça multithreading, em tempo de execução.
- Os Web Workers podem ter subworkers.
- O número de workers que você pode usar é ilimitado, mas muitos workers podem dificultar o desempenho.
- Os Web Workers podem receber um único parâmetro contendo quaisquer dados necessários para o worker.
- Os Web Workers não têm acesso ao DOM da página de chamada.
- Use `setTimeout` para atrasar antes de executar uma função de script. Use `setInterval` para repetir um função de script após cada intervalo especificado.

## Objective Review

1. Which of the following isn't a valid web worker operation?

- A. postMessage
- B. onmessage
- C. close
- D. terminate

- A. Incorreto: o postMessage inicia o roteiro a ser executado no worker.
- B. Incorreto: a onmessage é o manipulador de eventos usado para receber as mensagens dos workers.
- C. Correto: close não é um método para o web worker.
- D. Incorreto: terminate é usado para cancelar um worker web.

2. Which method cancels a web worker?

- A. close
- B. terminate
- C. suspend
- D. sleep

- A. Incorreto: close não cancela um web worker.
- B. Correto: terminate cancela um web worker.
- C. Incorreto: suspend não é um método válido.
- D. Incorreto: sleep não é um método válido.

3. Where must you place the JavaScript code to run in the context of a web worker?

- A. Between the <head></head> elements
- B. In any <script> block in the page
- C. In its own JavaScript file
- D. As a dynamic function assigned to the self.worker

- A. Incorreto: O código deve estar em seu próprio arquivo.
- B. Incorreto: O código não pode estar dentro de um bloco <script>.
- C. Correto: O código deve estar em seu próprio arquivo JavaScript, e o nome do arquivo é passado ao web worker como um parâmetro.
- D. Incorreto: Não existe tal propriedade como self.worker

4. How many web workers/subworkers can run concurrently?

- A. A multiple of four web workers including subworkers, per processor
- B. 16 workers by default, but you can change that number via self.configuration
- C. A limitless number of workers
- D. A limit of eight workers, each with a maximum of eight subworkers

- A. Incorreto: Há um limite associado aos processadores.
- B. Incorreto: Não existe tal propriedade como a self.configuration.
- C. Correto: Não há limite para o número de workers que podem ser criados. No entanto, workers em demasia resultarão em problemas de desempenho.
- D. Incorreto: Não existe tal limitação.

5. To have a script run continually every 30 seconds, which line of code should be used?

- A. wsConnection.repeatWork("workerFile.js",30000);
- B. setTimeout(function(){ worker.postMessage("");}, 30000);
- C. setTimeout( worker.postMessage(""), 30000);
- D. **setInterval(function(){ worker.postMessage("")}, 30000)**

A. Incorreto: wsConnection.repeatWork("workerFile.js",30000); não é um código válido.

B. Incorreto: setTimeout(function(){ worker.postMessage("");}, 30000); atrasará 30 segundos antes de executar a função anônima.

C. Incorreto: Em setTimeout("worker.postMessage("")", 30000);, setTimeout espera pelo atraso especificado antes de executar a função de passagem. Neste caso, o parâmetro não é uma função.

D. Correto: setInterval(function(){ worker.postMessage("")}, 30000); chama a função a cada intervalo, conforme especificado pelo segundo parâmetro em milissegundos.

## CAPÍTULO 3 - Acesso e segurança de dados

A maioria das aplicações web requer dados estáticos ou dinâmicos. Os dados estáticos são renderizados e exibidos aos usuários sem nenhuma maneira de alterar os dados. Os dados dinâmicos podem mudar. Dados dinâmicos podem, por exemplo, a partir de um feed de notícias, capturar dados do usuário para realizar uma operação e fornecer resultados, ou talvez até mesmo armazenar apenas as informações de registro de um usuário em um banco de dados.

Objetivos neste capítulo:

Aula 1: Validar a entrada do usuário usando elementos HTML5

Aula 2: Validar a entrada do usuário usando JavaScript

Aula 3: Consumir dados

Aula 4: Serializar, desserializar e transmitir dados

# Aula 1 - Validando a entrada de dados do usuário com elementos HTML5

## Objetivos

- Escolher os controles HTML5 de entrada de dados de usuário
- Implementar atributos

## Escolhendo os controles HTML5 de entrada de dados de usuário

TABLE 3-1 HTML5 input elements

Element	Description
<i>color</i> *	Provides a color picker
<i>date</i> *	Provides a date picker
<i>datetime</i> *	Provides a date/time picker
<i>month</i> *	Enables users to select a numeric month and year
<i>week</i> *	Enables users to select a numeric week and year
<i>time</i> *	Enables users to select a time of day
<i>number</i> *	Forces the input to be numeric
<i>Range</i>	Allows users to select a value within a range by using a slider bar
<i>tel</i> *	Formats entered data as a phone number
<i>url</i>	Formats entered data as a properly formatted URL
<i>Radio</i> t	Enables users to select a single value for a list of choices
<i>Checkbox</i> t	Enables users to select multiple values in a list of choices
<i>Password</i> t	Captures a password and glyphs the entered characters
<i>Button</i> t	Enables users to perform an action such as run script
<i>Reset</i> t	Resets all HTML elements within a form
<i>Submit</i> t	Posts the form data to a destination for further processing

\*Not supported currently by Internet Explorer

+Not new in HTML5

## aula\_1\exemplo\_01.html

```
<!DOCTYPE HTML>
<html>

    <head>
        <meta charset="utf-8" />
        <title>Validate Form HTML5</title>
        <style>
            td {
                margin: 5px;
                padding: 5px;
            }
            #otherCommentsText{
                padding: 5px;
            }
        </style>
    </head>

    <body>

        <form method="post" action="dados.php">
            <div>
                <hgroup>
                    <h1>Customer Satisfaction is #1</h1>
                    <h2>Please take the time to fill out the following survey</h2>
                </hgroup>
            </div>
            <table>
                <tr>
                    <td>Your Secret Code: </td>
                    <td>
                        <input type="text" name="secretKey" readonly="readonly" value="00XY998BB" />
                    </td>
                </tr>
                <tr>
                    <td>Password: </td>
                    <td>
                        <input type="password" name="password" />
                    </td>
                </tr>
                <tr>
                    <td>First Name: </td>
                    <td>
                        <input type="text" name="firstName" minlength="3" maxlength="20" />
                    </td>
                </tr>
                <tr>
                    <td>Last Name: </td>
                    <td>
                        <input type="text" name="lastName" minlength="3" maxlength="20" />
                    </td>
                </tr>
                <tr>
                    <td>Your favorite website: </td>
                    <td>
                        <input type="url" name="favouriteWebsite" />
                    </td>
                </tr>
            </table>
        </form>
    </body>

```

```

<tr>
    <td> Your age in years: </td>
    <td>
        <input type="number" name="age" />
    </td>
</tr>
<tr>
    <td> What colors have you colored your hair: </td>
    <td>
        <input type="checkbox" id="chkBrown" checked="checked" /> Brown
        <input type="checkbox" id="chkBlonde" /> Blonde
        <input type="checkbox" id="chkBlack" /> Black
        <input type="checkbox" id="chkRed" /> Red
        <input type="checkbox" id="chkNone" /> None
    </td>
</tr>
<tr>
    <td>Rate your experience: </td>
    <td>
        <input type="radio" id="chkOne" value="very_poor" name="experience" /> 1 - Very Poor
        <input type="radio" id="chkTwo" value="poor" name="experience" /> 2 - Poor
        <input type="radio" id="chkThree" value="medium" name="experience" /> 3 - Medium
        <input type="radio" id="chkFour" value="good" name="experience" /> 4 - Good
        <input type="radio" id="chkFive" value="very_good" name="experience" checked="checked" /> 5
        - Very Good
    </td>
</tr>
<tr>
    <td>How likely would you recommend the product: </td>
    <td>
        <input type="range" min="1" max="25" name="recommend" value="20" />
    </td>
</tr>
<tr>
    <td> Other Comments: </td>
    <td>
        <textarea id="otherCommentsText" name="otherComments" rows="10" cols="30" spellcheck="true"></textarea>
    </td>
</tr>
<tr>
    <td> Email address: </td>
    <td>
        <input type="email" name="email" placeholder="Digite seu e-mail" required/>
    </td>
</tr>
<tr>
    <td>
        <input type="submit" value="Submit" />
        <input type="reset" value="Clear" />
        <input type="button" value="Message" onClick="ChamarFuncaoAlerta()" />
    </td>
</tr>
</table>
</form>
<script>
    function ChamarFuncaoAlerta(){
        alert('Chamei a função!');
    }
</script>

```

```
</body>
</html>
```

← → ⌂ ⓘ localhost/html5\_css3/capitulo\_3/aula\_1/exemplo\_01.html

## Customer Satisfaction is #1

Please take the time to fill out the following survey

Your Secret Code:

Password:

First Name:

Last Name:

Your favorite website:

Your age in years:

What colors have you colored your hair:

Brown  Blonde  Black  Red  None

Rate your experience:

1 - Very Poor  2 - Poor  3 - Medium  4 - Good  5 - Very Good

How likely would you recommend the product:



Other Comments:

Email address:

### aula\_1\dados.php

<?php

```
echo "<h1>Dados coletados</h1>";

if($_POST["secretKey"]){
    $secretKey = $_POST["secretKey"];
    echo "<p>Secret key: $secretKey</p>";
} else {
    echo "<p>secretKey não recebido</p>";
}

if($_POST["password"]){
    $password = $_POST["password"];
    echo "<p>Password: $password</p>";
} else {
    echo "<p>password não recebido</p>";
}

if($_POST["firstName"]){
```

```

$firstName = $_POST["firstName"];
echo "<p>First name: $firstName</p>";
} else {
    echo "<p>firstName não recebido</p>";
}

if($_POST["lastName"]){
    $lastName = $_POST["lastName"];
    echo "<p>Last name: $lastName</p>";
} else {
    echo "<p>lastName não recebido</p>";
}

if($_POST["favouriteWebsite"]){
    $favouriteWebsite = $_POST["favouriteWebsite"];
    echo "<p>Favourite website: $favouriteWebsite</p>";
} else {
    echo "<p>favouriteWebsite não recebido</p>";
}

if($_POST["age"]){
    $age = $_POST["age"];
    echo "<p>Age: $age anos</p>";
} else {
    echo "<p>age não recebido</p>";
}

if($_POST["experience"]){
    $experience = $_POST["experience"];
    echo "<p>Experience: $experience</p>";
} else {
    echo "<p>experience não recebido</p>";
}

if($_POST["recommend"]){
    $recommend = $_POST["recommend"];
    echo "<p>Recommend: $recommend</p>";
} else {
    echo "<p>recommend não recebido</p>";
}

if($_POST["otherComments"]){
    $otherComments = $_POST["otherComments"];
    echo "<p>Other comments: $otherComments</p>";
} else {
    echo "<p>otherComments não recebido</p>";
}

if($_POST["email"]){
    $email = $_POST["email"];
    echo "<p>Email: $email</p>";
} else {
    echo "<p>email não recebido</p>";
}

?>

```

## Dicas para exames

1. A validação de dados pode ser feita de outras maneiras, como por exemplo através de jQuery e do uso de expressões em JavaScript.
2. Você não pode especificar o texto padrão em uma caixa de senha. Este é um item de segurança para ajudar a garantir a segurança das senhas. Entretanto, os navegadores fornecem um mecanismo para armazenar uma senha caso um usuário opte por ter a senha lembrada pelo navegador.
3. A validação do tipo de entrada de e-mail confirma apenas que as informações inseridas correspondem ao formato esperado de um endereço de e-mail válido. Ele não verifica de forma alguma que o endereço de e-mail em si é uma caixa de correio válida que pode receber mensagens.
4. Qualquer coisa pode ser um "botão". A maioria dos elementos DOM tem um evento de clique ou pelo menos um evento de mousedown e mouseup. Ações personalizadas podem ser programadas ao se executar um click em um elemento. Isto pode intrinsecamente transformar qualquer parte do DOM em um "botão".

## Implementando atributos de conteúdo

### Fazendo controles read-only

São controles somente para leitura. Não podem ser modificados.

```
<tr>
  <td>
    Your Secret Code:
  </td>
  <td>
    <input type="text" readonly value="00XY998BB"/>
  </td>
</tr>
```

### Fornecendo um corretor ortográfico

A verificação ortográfica é outro método disponível para validar a entrada do usuário. O atributo de verificação ortográfica ajuda a fornecer feedback aos usuários de que uma palavra que eles digitaram está mal escrita.

```
<textarea id="otherCommentsText" rows="5" cols="20" spellcheck="true"></textarea>
```

Neste HTML, a opção de verificação ortográfica foi ativada para a caixa de texto otherComments porque os usuários podem digitar o que quiserem e assim cometer erros ortográficos. A saída de uma caixa de texto com verificação ortográfica não é diferente até que um usuário comece a digitar e comete um erro

ortográfico. Um sublinhado vermelho é exibido para as palavras que são detectadas e escritas de forma incorreta.

## Especificando um padrão

Os tipos de entrada de e-mail e url possuem uma validação integrada bastante completa para garantir que as informações inseridas são precisas e como esperado. No entanto, em alguns casos, você pode exigir uma validação menos rigorosa. Suponha que você não queira que os usuários tenham que especificar o protocolo HTTP em um tipo url, mas você quer permitir apenas sites .com ou .ca. Isto pode ser obtido usando o atributo padrão, que permite o uso de uma expressão regular para definir o padrão que deve ser aceito.

## Dica para exames

O atributo padrão se aplica apenas a caixas de texto. Ele não pode ser usado para anular a validação embutida nos tipos de e-mail ou url.

O código a seguir mostra o atributo padrão utilizado para obter a validação desejada:

```
<input type="text" title="Somente .com e .ca são permitidos" pattern="^@[a-zA-Z0-9\-\\.]+\.(com|ca)$"/>
```

## Usando o atributo placeholder

O atributo placeholder permite que você informe aos usuários o que é esperado em uma determinada caixa de texto. Por exemplo, uma caixa de texto de e-mail pode mostrar um placeholder, como me@mydomain.com. Mais importante ainda, este texto não interfere com os usuários quando começam a digitar suas informações na caixa de texto.

```
<input type="email" placeholder="me@mydomain.com" /></td>
```

## Controles required

Para garantir que um usuário preencha um campo, use o atributo **required** com o elemento `<input>`. Fazendo assim garante que os usuários serão informados de que o campo é requerido (obrigatório). Neste exemplo, o endereço de e-mail será feita uma caixa de texto obrigatória:

```
<input type="email" placeholder="me@mydomain.com" required />
```

Com o controle **required** especificado, se os usuários tentarem enviar o formulário sem especificar um endereço de e-mail, eles recebem uma mensagem de erro.

## Resumo

- Os controles de entrada como texto e área de texto permitem que os usuários digitem informações em um
- página web.
- Alguns controles de entrada fornecem validação incorporada, tais como para URLs e endereços de e-mail.
- Botões de rádio e caixas de seleção fornecem controles para que os usuários selecionem itens em uma lista.
- Os botões submit e reset controlam o comportamento do formulário HTML.
- Os usuários não podem modificar o conteúdo de um controle que tenha o atributo readonly atribuído.
- Você pode adicionar um verificador ortográfico a uma caixa de texto para ajudar os usuários a evitar erros ortográficos.
- O atributo padrão ajuda a definir uma expressão regular para validação personalizada de dados formatados.
- O atributo requerid garante que um campo seja preenchido antes que os usuários possam enviar os dados de um campo de formulário

## Objective Review

1. Which input control is better suited for allowing users to make multiple selections?

- A. radio button
- B. textarea
- C. **checkbox**
- D. radio or checkbox

A. Incorreto: Um radio button é adequado para permitir uma única seleção.

B. Incorreto: Um textarea é adequado para uma caixa de texto com várias linhas.

C. Correto: checkboxes permitem seleções múltiplas.

D. Incorreto: Um radio button não permite mais de uma seleção.

2. Which input control is designed to allow users to enter secure information in a way that keeps others from seeing what's typed?

- A. text
- B. textarea
- C. url
- D. **password**

A. Incorreto: Uma caixa de texto permite a entrada de dados, mas é claramente visível.

B. Incorreta: Um textarea permite a entrada de dados, mas é claramente visível.

C. Incorreta: url é um tipo de caixa de texto com regras de validação especiais.

D. Correto: Um tipo de entrada password esconde os caracteres que estão sendo inseridos.

3. Which input control posts form data to a server?

- A. button
- B. **Submit**
- C. Reset
- D. radio

- A. Incorreto: button deve ter um manipulador de eventos para realizar lógica.
- B. Correto: O botão submit invoca a ação de submissão de formulários.
- C. Incorreto: O botão reset limpa todos os campos de entrada no formulário.
- D. Incorreto: Um botão rádio é usado para uma lista de seleção.

4. Which of the following declarations are valid ways to make a text control non-editable?

- A. `<input type="text" edit="false"/>`
- B. `<input type="text" editable="false"/>`
- C. `<input type="text" readonly="yes"/>`
- D. `<input type="text" readonly/>`

5. How can you ensure that all necessary fields are populated before a form can be submitted?

- A. Write a JavaScript function to evaluate all the controls on the form for content.
- B. On the server, evaluate all the controls for data and return an error page for missing content.
- C. Add the required attribute on each control so that users get a message that the field is required.
- D. Add a label to the page to let users know which controls they must fill in.

- A. Incorreto: Você pode fazer isso com um evento personalizado, mas isso é mais trabalho do que necessário.
- B. Incorreto: O objetivo é validar os dados antes de enviar o formulário.
- C. Correto: O atributo required garante que um campo contenha um valor antes de ser apresentado.
- D. Incorreto: Uma etiqueta seria informativa, mas não garante que todos os requisitos campos serão preenchidos antes de serem enviados.

## Aula 2 - Validando uma entrada de usuário com Javascript

Os novos controles HTML5 fornecem algumas grandes funcionalidades para validar os dados do usuário. No entanto, esta funcionalidade tem algumas limitações. É aqui que a validação adicional realizada em JavaScript vem a calhar. O JavaScript fornece funcionalidade adicional que não está prontamente disponível nos controles de HTML5. Embora alguns controles ainda não estejam disponíveis em todos os navegadores, talvez seja necessário validar a entrada do usuário, tais como datas, números de telefone, ou códigos postais alfanuméricos. Veremos como usar expressões regulares para validar o formato de entrada e como usar as funções embutidas do JavaScript para garantir que os dados são do tipo de dado correto. Veremos também como prevenir a injeção de código malicioso

### Objetivos

- Usar expressões regulares para validação de campos de formulário
- Uso de funções nativas do Javascript
- Prevenir Code injection

### Avaliando expressões regulares

O núcleo de controle de entrada HTML suporta um atributo padrão que lhe permite aplicar uma expressão regular para validar a entrada do usuário. Em alguns casos, porém, a validação da entrada do usuário no JavaScript pode ser mais eficaz do que em linha com os atributos. Esta seção introduz expressões regulares.

As expressões regulares têm uma sintaxe própria e única. Elas podem ser assustadoras de usar, mas também podem ser muito poderosas.

### Dica para exames

O uso de expressões regulares tende a cair em exames. Você deve se preparar estudando-as com mais detalhes. Uma busca na Internet deve render muitos recursos livremente disponíveis sobre o assunto. Esteja familiarizado com a leitura de uma expressão regular para coisas como endereços de e-mail, URLs, e números de telefone, entre outras coisas. As expressões regulares são uma mistura de caracteres especiais e caracteres literais que compõem o padrão.

**TABLE 3-1** Regular expression special characters

Symbol	Description
^	The caret character denotes the beginning of a string.
\$	The dollar sign denotes the end of a string.
.	The period indicates to match on any character.
[A-Z]	Alphabet letters indicate to match any alphabetic character. This is case-sensitive. To match lowercase letters, use [a-z].
\d	This combination indicates to match any numeric character.
+	The plus sign denotes that the preceding character or character set must match at least once.
*	The asterisk denotes that the preceding character or character set might or might not match. This generates zero or more matches.
[^]	When included in a character set, the caret denotes a negation. [^a] would match a string that doesn't have an 'a' in it.
?	The question mark denotes that the preceding character is optional.
\w	This combination indicates to match a word character consisting of any alphanumeric character, including an underscore.
\	The backslash is an escape character. If any special character should be included in the character set to match on literally, it needs to be escaped with a \. For example, to find a backslash in a string, the pattern would include \\.
\s	This combination indicates to match on a space. When it's combined with + or *, it can match on one or more spaces.

Símbolo	Descrição
^	O caractere ^ denota o início de uma expressão.
\$	O sinal \$ denota o fim de uma expressão.
.	indica a correspondência em qualquer caractere.
[A-Z]	Indicam qualquer caractere alfabético. Para combinar letras minúsculas, use [a-z].
\d	Esta combinação indica qualquer caractere numérico.
+	O sinal de + indica que o caractere ou conjunto de caracteres anterior deve corresponder pelo menos uma vez.
*	O asterisco denota que o caracter ou conjunto de caracteres anterior pode ou não corresponder. Isto gera zero ou mais correspondências.
[^]	Quando incluído em um conjunto de caracteres, o ^ denota uma negação. [^a] corresponderia a uma expressão que não tem um "a".
?	O ponto de interrogação denota que o caráter anterior é opcional.
\w	Esta combinação indica a correspondência de qualquer caractere alfanumérico, incluindo um sublinhado.
\	A contrabarra é um caractere de escape. Se algum caractere especial deve ser incluído no set de caracteres que se encaixam literalmente, é preciso escapar com um \. Por exemplo, para encontrar uma contrabarra em uma expressão, o padrão incluiria \\. Esta combinação indica que combina em um espaço. Quando é combinada com + ou *, ela pode combinar em um ou mais espaços.

Agora, vamos construir a expressão regular para um código postal. Primeiro é preciso denotar o início da string, pois isso ajuda a eliminar espaço branco desnecessário à frente da mesma:

^

A primeira parte da expressão é o ^ . O próximo caractere deve ser alfabético:

^[A-Z,a-z]

Como os códigos postais não são sensíveis a maiúsculas e minúsculas, a expressão permite que o primeiro caractere seja em maiúsculas ou minúsculas. O próximo caractere no código postal deve ser um dígito:

^[A-Z,a-z]^d

Como o código postal aceita todos os dígitos 0-9, \d é usado para especificar qualquer dígito. No entanto, [0-9] poderia ter sido usado também. E agora o padrão continua, letra-número-letra-numero-letra-número:

^[A-Z,a-z]\d[A-Z,a-z]\d[A-Z,a-z]\d

Como foi indicado anteriormente, o espaço no meio do código postal, embora seja uma convenção, é opcional.

Aqui é onde se decide quão flexível deve ser a validação dos dados. A expressão como é não permitirá nenhum espaço no meio porque é definida para corresponder em letras consecutivas alternadas. Talvez, para fins de formatação, deva ser necessário um espaço. Neste caso, seria necessário que fosse incluído um espaço:

^[A-Z,a-z]\d[A-Z,a-z]\s\d[A-Z,a-z]\d

Agora, os usuários seriam obrigados a inserir o código postal com um espaço no meio dos dois conjuntos de três caracteres. Mas talvez o site não se importe com o espaço no meio, porque realmente não afeta nada. Neste caso, os {\i1}s podem ser identificados com o \*:

^[A-Z,a-z]\d[A-Z,a-z]\s\*\d[A-Z,a-z]\d

Agora, a expressão permite alternar letra/número de letra e no meio pode ocorrer um ou mais espaços. O espaço agora é opcional, mas foi introduzido um problema. O usuário agora pode digitar qualquer número de espaços e a validação vai passar, como por exemplo:

A1A 1A1

Essa validação passaria porque um ou mais espaços são requeridos por \s\*.

O resultado desejado aqui é permitir apenas um espaço ou nenhum espaço. Para isso, um novo elemento é adicionado para limitar o número de ocorrências a apenas uma. Isto é conseguido especificando o comprimento máximo permitido para que o conjunto de caracteres seja correspondido:

^[A-Z,a-z]\d[A-Z,a-z]\s{1}\d[A-Z,a-z]\d

O {1} diz para combinar com o caractere anterior apenas o número especificado de vezes - neste caso, uma vez.



Agora a expressão está de volta à funcionalidade que não é diferente de apenas especificar os \s. O que é necessário a seguir é algo para tornar o espaço único opcional, como indicado com o ?. Para conseguir este efeito, o segmento espacial é envolto em colchetes para torná-lo um conjunto e seguido pelo ? para torná-lo opcional:

```
^[A-Z,a-z]\d[A-Z,a-z][\s{1}]?\d[A-Z,a-z]\d
```

Agora você tem uma expressão regular que requer o padrão alfanumérico correto para um código postal canadense com um espaço opcional no meio.

Este exemplo simples demonstra os elementos-chave para uma expressão regular. Embora uma expressão regular pode ser colocada no atributo padrão do elemento <input>, o próximo item discute como usar a estrutura JavaScript para realizar a correspondência de padrões com expressões regulares.

## Analizando expressões regulares em Javascript

Assim como com as strings e os inteiros, as expressões regulares são objetos em JavaScript. Como tal, elas podem ser criadas e podem fornecer métodos. Os objetos de expressão regular são criados de forma semelhante a strings; no entanto, em vez de usar "" para encapsular a expressão, use /<expressão>/ em seu lugar. O JavaScript sabe que o texto cercado por / é um objeto de expressão regular.

### aula\_2\exemplo\_01.html

```
<!DOCTYPE html>
<html>

    <head>

        <meta charset="utf-8" />
        <title>Validação CEP</title>

        <script src="https://code.jquery.com/jquery-2.2.4.min.js" integrity="sha256-BbhdlvQf/xTY9gja0Dq3HiwQF8LaCRTXxZKRutelT44=" crossorigin="anonymous"></script>
        <style>
            body{
                margin: 15px;
            }
        </style>
    </head>

    <body>

        <script type="text/javascript">
            function CheckString() {
                try {
                    var s = $('#regExString').val();
                    // var regExpression = /^[A-Z,a-z]\d[A-Z,a-z][\s{1}]\?\d[A-Z,a-z]\d/;
                    var regExpression = /^[0]+[1-9]+\d{3}[[-]{1}\?\d{3}\$/;

                    if (regExpression.test(s))
                        alert("CEP válido!");
                    else alert("CEP inválido!");
                } catch (e) {
                    alert(e.message);
                }
            }

        </script>

        <form>
            <label>CEP: </label>
            <input type="text" id="regExString" />
            <button onclick="CheckString();">Evaluate</button>
        </form>

    </body>

</html>
```

The image consists of three vertically stacked screenshots of a web browser window. Each screenshot shows a form with a 'CEP:' label and an empty input field. Below the input field is an 'Evaluate' button. To the right of the input field, there is a message box with a title 'localhost diz'. In the first screenshot, the message box is empty. In the second screenshot, the message box contains the text 'CEP válido!' and has a blue 'OK' button at the bottom. In the third screenshot, the message box contains the text 'CEP inválido!' and also has a blue 'OK' button at the bottom.

## Validando dados com funções nativas do Javascript

O JavaScript fornece funções incorporadas para avaliar o tipo de dados. Algumas funções são fornecidas diretamente dentro do JavaScript; outras são fornecidas pela biblioteca jQuery.

A função **isNaN** fornece uma maneira de avaliar se o valor passado para ela não é um número. Se o valor não for um número, a função retorna true; se for um número, retorna false.

Se a forma esperada de dados a serem avaliados for numérica, esta função fornece uma maneira de determiná-la e tratá-la adequadamente:

```
if (isNaN(value)) {
    //handle the non number value
}
else {
    //proceed with the number value
}
```

O oposto da função **isNaN** é a função **isFinite**. A função **isFinite** é usada da mesma forma, mas retorna true se o valor for um número finito e false se não for.

Ser capaz de validar dados é muito importante. É importante validar os dados e garantir explicitamente que os campos de entrada de dados impeçam os usuários de injetar código. A injeção de código é um tópico amplamente discutido na segurança do site.

## Prevenindo injeção de código

A injeção de código é uma técnica que os atacantes utilizam para injetar código JavaScript em sua página web. Estes ataques geralmente aproveitam o conteúdo criado dinamicamente para que os usuários maliciosos tentem obter algum tipo de controle sobre o seu site.

## Proteção contra a entrada de usuários

Uma aplicação web que aceita a entrada do usuário abre uma superfície de ataque potencial para usuários maliciosos. O tamanho da superfície de ataque depende do que for feito com os dados inseridos. Se o site da Web obtém os dados e não faz nada com eles fora do escopo da página web atual, tais como enviá-lo para outro servidor ou armazená-lo em um banco de dados, os efeitos são limitados à página atual e à sessão do navegador. Pouco pode ser realizado, exceto para interromper o projeto do site para este usuário em particular. Entretanto, se os dados capturados incluírem a criação de uma conta por exemplo, um usuário malicioso tem muito mais potencial para causar danos - especialmente quando essa informação é apresentada mais tarde à página web de forma dinâmica. Isto permite, inherentemente que qualquer pessoa possa adicionar script ao site, que pode abri-lo para comportamentos como phishing. Como um desenvolvedor da página web, você precisa garantir que toda a entrada do usuário seja eliminada dos elementos do script. Por exemplo, não permita que texto < > seja inserido no formulário. Sem esses caracteres, um bloco de código não pode ser adicionado.

## Usando a função eval

A função **eval** é usada para executar JavaScript de forma dinâmica. Ela toma uma string como parâmetro e funciona como uma função JavaScript. Nunca use a função **eval** para obter dados fornecidos por uma fonte externa sobre a qual você não tem 100% de controle.

## Usando iFrames

iFrames abrem uma nova oportunidade para os atacantes. Os motores de busca fornecem uma infinidade de resultados lidando com explorações relativas ao uso de iFrames. O atributo **sandbox** deve sempre ser usado para restringir quais dados podem ser colocados em um iFrame. O atributo sandbox tem quatro valores possíveis, conforme listados na Tabela 3-2.

**TABLE 3-2 Available sandbox attribute values**

<b>Value</b>	<b>Description</b>
""	An empty string applies all restrictions. This is the most secure.
<i>allow-same-origin</i>	iFrame content is treated as being from the same origin as the containing HTML document.
<i>allow-top-navigation</i>	iFrame content can load content from the containing HTML document.
<i>allow-forms</i>	iFrame can submit forms.
<i>allow-scripts</i>	iFrame can run script.

## Resumo

- Expressões regulares são expressões de caracteres especiais que um interpretador entende e usa para validar o formato do texto.
- Expressões regulares são objetos em JavaScript que fornecem métodos para testar a entrada de dados.
- isNaN é uma função integrada para determinar se um valor não é um número, enquanto isFinite valida se o valor é um número finito.
- A injeção de código é uma técnica que os atacantes usam para injetar código malicioso em sua aplicação.
- iFrames e JavaScript dinâmico são perigosos se não forem usados corretamente em uma página da web.

## Objective Review

1. Which of the following regular expression characters denote the end of the string?

A. \$

B. %

C. ^

D. &

A. Correcto: O sinal \$ denota o fim da expressão.

B. Incorreto: O sinal de % não denota o fim da expressão.

C. Incorreto: O caractere ^ denota o início da cadeia de caracteres.

D. Incorreto: O caractere & não denota o fim da cadeia de caracteres.

2. Which of the following sandbox attributes allows the iFrame to load content from the containing HTML document?

A. allow-script-execution

B. allow-same-origin

C. allow-forms

**D. allow-top-navigation**

E. allow-top-document

A. Incorreto: Permite que os scripts sejam executados

B. Incorreto: Só permite conteúdo da mesma origem

C. Incorreto: Permite formas

D. Correto: Permite o conteúdo do documento HTML que contém

E. Incorreto: Não é uma opção válida

3. Which function should never be used to run JavaScript?

A. execute

B. JSDynamic

**C. eval**

D. evaluate

## Aula 3 - Consumir dados JSON e XML

Veremos como consumir dados em uma aplicação web HTML5. A capacidade de consumir dados de fontes externas é mais popular do que nunca.

### Objetivos

Consumir JSON e XML usando web services

Usar o objeto XMLHttpRequest

### Consumindo dados JSON e XML através de serviços web

Os dois formatos de dados comumente usados na transmissão de dados são JSON e XML. O JSON são dados não estruturados, enquanto que o XML é estruturado. O JSON usa uma sintaxe especial que permite a definição de pares de valores de nomes em um formato de expressão leve. O XML, como um parente do HTML, é mais estruturada do que o JSON com tags nomeadas e tags de abertura e fechamento. As etiquetas podem ter atributos. A seguir, exemplos de como um objeto pessoa pode se parecer em ambos os formatos onde a pessoa objeto tem um nome, sobrenome, cor de cabelo e cor dos olhos:

JSON:

```
{firstName: "Rick", lastName: "Delorme", hairColor: "brown", eyeColor: "brown" }
```

XML (Elements):

```
<Person>
  <firstName>Rick</firstName>
  <lastName>Delorme</lastName>
  <hairColor>Brown</hairColor>
  <eyeColor>Brown</eyeColor>
</Person>
```

XML (attributes):

```
<Person firstname="Rick" lastName="Delorme" hairColor="Brown" eyeColor="Brown"/>
```

Ao publicar serviços de dados, como Web Services ou uma API REST, você pode controlar como você publica os dados. Ao consumir recursos de terceiros, você não terá controle sobre como eles publicaram os dados.

### Usando o objeto XMLHttpRequest

O JavaScript fornece suporte integrado para o recebimento de dados HTML através do objeto XMLHttpRequest. O objeto faz uma chamada para um serviço web, REST API, ou outros serviços de provedor de dados. A vantagem de fazer isso via JavaScript no lado do cliente é poder recarregar partes de uma fonte externa sem ter que postar a página inteira de volta ao servidor. O XMLHttpRequest faz uma solicitação HTTP e espera receber os dados de volta em formato XML. Tanto chamadas síncronas quanto assíncronas são suportadas. As tabelas 3-3, 3-4, e 3-5 listam os eventos, métodos e propriedades disponíveis do objeto XMLHttpRequest.

**TABLE 3-3** Available events of the XMLHttpRequest object

Events	Description
<i>Onreadystatechange</i>	Sets an event handler for when the state of the request has changed. Used for asynchronous calls.
<i>Ontimeout</i>	Sets an event handler for when the request can't be completed.

**TABLE 3-4** Available methods of the XMLHttpRequest object

Method	Description
<i>Abort</i>	Cancels the current request
<i>getAllResponseHeaders</i>	Gives a complete list of response headers
<i>getResponseHeader</i>	Returns the specific response header
<i>Send</i>	Makes the HTTP request and receives the response
<i>setRequestHeader</i>	Adds a custom HTTP header to the request
<i>Open</i>	Sets properties for the request such as the URL, a user name, and a password

**TABLE 3-5** Available properties of the XMLHttpRequest object

Property	Description
<i>readyState</i>	Gets the current state of the object
<i>Response</i>	Gets the response returned from the server
<i>responseBody</i>	Gets the response body as an array of bytes
<i>responseText</i>	Gets the response body as a string
<i>responseType</i>	Gets the data type associated with the response, such as <i>blob</i> , <i>text</i> , <i>arraybuffer</i> , or <i>document</i>
<i>responseXML</i>	Gets the response body as an XML DOM object
<i>Status</i>	Gets the HTTP status code of the request
<i>statusText</i>	Gets the friendly HTTP text that corresponds with the status
<i>Timeout</i>	Sets the timeout threshold on the request
<i>withCredentials</i>	Specifies whether the request should include user credentials

Em sua forma mais simples, uma requisição ao servidor usando o objeto XMLHttpRequest se parece isto:

```

<script>
  $("document").ready(function () {
    $("#btnGetXMLData").click(function () {
      var xReq = new XMLHttpRequest();
      xReq.open("GET", "myXMLData.xml", false);
      xReq.send(null);
      $("#results").text(xReq.responseText);
    });
  });
</script>

```

Este script assume um botão no formulário HTML e uma div para mostrar os resultados. Um novo objeto `XMLHttpRequest` é criado. O método `open` é chamado para especificar o tipo de solicitação, a URI, e se a chamada deve ser feita de forma assíncrona.

### **aula\_3\exemplo\_01.html**

```

<!DOCTYPE html>
<html>
  <body>
    <h2>Using the XMLHttpRequest Object</h2>

    <div id="demo">
      <button type="button" onclick="loadXMLDoc()">Change Content</button>
    </div>

    <script>
      function loadXMLDoc() {
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function() {
          if (this.readyState == 4 && this.status == 200) {
            document.getElementById("demo").innerHTML = this.responseText;
          }
        };
        xhttp.open("GET", "xmlhttp_info.txt", true);
        xhttp.send();
      }
    </script>

  </body>
</html>

```



## **Using the XMLHttpRequest Object**

**Change Content**



## Using the XMLHttpRequest Object

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec bibendum condimentum purus, non ullamcorper libero egestas sit amet. Morbi id sapien rutrum, iaculis turpis et, finibus sem. Pellentesque aliquam faucibus dolor nec volutpat. Phasellus vehicula cursus nisl id malesuada. Donec sed nisl laoreet eros sagittis tempus vel nec purus. Mauris volutpat lorem non sapien euismod, eget faucibus nisl scelerisque. Nulla suscipit fringilla mauris ac rutrum. Maecenas suscipit iaculis nulla, sed ornare lectus accumsan quis. Aenean malesuada, diam at mollis pharetra, tortor neque ornare erat, non consequat nisi nibh vel tellus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Sed eget luctus elit, vitae bibendum ligula. Nulla ex neque, ornare vel lorem ac, sagittis porttitor lacus. Praesent ullamcorper bibendum orci, vitae porta est porta non. Nam ullamcorper mollis urna. Nam sit amet feugiat erat. Donec malesuada a nibh sit amet eleifend. Integer non enim et nisl congue vulputate. Ut maximus arcu eu eros rutrum consequat. Praesent dictum ullamcorper mi, at sagittis mauris volutpat vel. Ut mattis varius eleifend. Morbi neque metus, luctus sit amet magna nec, facilisis mollis augue. Ut mattis viverra tortor eget interdum. Cras convallis, lorem vitae gravida accumsan, nibh purus dictum tellus, eu luctus odio nulla ut lacus.

## aula\_3\exemplo\_02.html

```
<!DOCTYPE html>
<html>
    <body>
        <h2>Using the XMLHttpRequest Object</h2>

        <div id="demo">
            <button type="button" onclick="loadXMLDoc()">Change Content</button>
        </div>

        <script>
            function loadXMLDoc() {
                var xhttp = new XMLHttpRequest();
                xhttp.onreadystatechange = function() {
                    if (this.readyState == 4 && this.status == 200) {
                        document.getElementById("demo").innerHTML = this.responseText;
                    }
                };
                xhttp.open("GET", "myXMLData.xml", true);
                xhttp.send();
            }
        </script>

    </body>
</html>
```



## Using the XMLHttpRequest Object



## Using the XMLHttpRequest Object

São Paulo SP Minas Gerais MG Rio de Janeiro RJ

**TABLE 3-6** Parameters for the XMLHttpRequest open method

Parameter	Description
<i>Method</i>	The HTTP method being used for the request: GET, POST, etc.
<i>URL</i>	The URL to make the request to.
<i>async</i>	A Boolean value to indicate whether the call should be made asynchronously. If true, an event handler needs to be set for the <i>onreadystatechange</i> .
<i>User name</i>	A user name if the destination requires credentials.
<i>Password</i>	A password if the destination requires credentials.

## Dica para exames

O método open não faz nenhuma requisição ao servidor. Se o nome do usuário e a senha forem especificados, ele não envia a requisição para o servidor no método open. Quando o método send é chamado, o nome do usuário e a senha também não são passados para o servidor. As credenciais são passadas para o servidor somente em response.

O objeto XMLHttpRequest fornece alguns mecanismos para o tratamento de erros. O erro mais comum a ser considerado é um erro de timeout. Por padrão, o valor do timeout é zero, que é infinito. Um valor de timeout deve ser sempre especificado. O código é atualizado da seguinte forma:

```
var xReq = new XMLHttpRequest();
xReq.open("GET", "myXMLData.xml", false);
xReq.timeout = 2000;
xReq.ontimeout = function () {
    $("#results").text("Request Timed out");
}
xReq.send(null);
$("#results").text(xReq.response);
```

Isto resulta em não permitir que a chamada demore mais de dois segundos. O tempo limite é expresso em milissegundos. Após o período de timeout, o encarregado do evento é chamado para permitir que esta condição seja tratada adequadamente na página web.

Uma consideração adicional para este código é se a chamada deve ser feita de forma síncrona ou de forma assíncrona. Idealmente, você deve garantir que a chamada para o serviço obtenha os dados não interfira com os usuários e não os bloqueie, a menos, é claro, que eles precisem esperar pela resposta antes de tomar qualquer outra medida.

Chamadas síncronas, como os exemplos até o momento têm mostrado, bloqueiam a interface do usuário enquanto a requisição está sendo feita. Para evitar isto, a chamada deve ser assíncrona, como mostrado aqui:

## aula\_3\exemplo\_03.html

```
<!DOCTYPE html>
<html>

    <head>

        <meta charset="utf-8" />

        <script src="https://code.jquery.com/jquery-2.2.4.min.js" integrity="sha256-BbhdlvQf/xTY9gja0Dq3HiwQF8LaCRTXxZKRutelT44=" crossorigin="anonymous"></script>
        <style>
            body{
                margin: 15px;
            }
        </style>
    </head>

    <body>
        <h2>Using the XMLHttpRequest Object</h2>

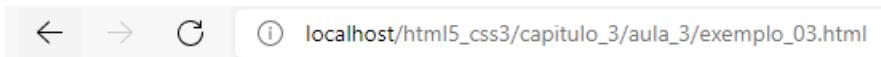
        <p id="results"></p>

        <div id="demo">
            <button type="button" onclick="loadXMLDoc()">Change Content</button>
        </div>

        <script>
            function loadXMLDoc() {

                var XMLHTTPReadyState_COMPLETE = 4;
                var xReq = new XMLHttpRequest();
                xReq.open("GET", "myXMLData.xml", true);
                xReq.timeout = 2000;
                xReq.ontimeout = function () {
                    $("#results").text("Request Timed out");
                }
                xReq.onreadystatechange = function (e) {
                    if (xReq.readyState == XMLHTTPReadyState_COMPLETE) {
                        if (xReq.status == "200") {
                            $("#results").text(xReq.response);
                        } else {
                            $("#results").text(xReq.statusText);
                        }
                    }
                }
                xReq.send(null);
            }

        </script>
    </body>
</html>
```



## Using the XMLHttpRequest Object

[Change Content](#)



## Using the XMLHttpRequest Object

```
<?xml version="1.0" encoding="UTF-8" ?> <estados> <estado> <nomeEstado>São Paulo</nomeEstado>
<sigla>SP</sigla> </estado> <estado> <nomeEstado>Minas Gerais</nomeEstado> <sigla>MG</sigla> </estado>
<estado> <nomeEstado>Rio de Janeiro</nomeEstado> <sigla>RJ</sigla> </estado> </estados>
```

[Change Content](#)

O evento onreadystatechange é atribuído a uma função a ser executada quando o estado do objeto XMLHttpRequest é alterado. Quando a solicitação é concluída, readyState muda para completar (readyState == 4). Neste ponto, o status do retorno HTTP pode ser avaliado para um valor de sucesso, como 200, e então o processamento dos dados XML pode ocorrer.

O mesmo código que tem sido usado até agora para recuperar dados XML também pode ser usado para fazer um solicitação de dados JSON. A seguinte atualização do código mostra isto:

### aula\_3\exemplo\_04.html

```
<!DOCTYPE html>
<html>

    <head>

        <meta charset="utf-8" />

        <script src="https://code.jquery.com/jquery-2.2.4.min.js" integrity="sha256-BbhdlvQf/xTY9gja0Dq3HiwQF8LaCRTXxZKRutelT44="
               crossorigin="anonymous"></script>
        <style>
            body{
                margin: 15px;
            }
        </style>
    </head>

    <body>
        <h2>Using the XMLHttpRequest Object</h2>

        <p id="results"></p>

        <div id="demo">
            <button type="button" onclick="loadXMLDoc()">Change Content</button>
        </div>

        <script>
```

```

function loadXMLDoc() {

    var XMLHTTPReadyState_COMPLETE = 4;
    var xReq = new XMLHttpRequest();
    xReq.open("GET", "myJSONData.json", true);
    xReq.timeout = 2000;
    xReq.ontimeout = function () {
        $("#results").text("Request Timed out");
    }
    xReq.onreadystatechange = function (e) {
        if (xReq.readyState == XMLHTTPReadyState_COMPLETE) {
            if (xReq.status == "200") {
                $("#results").text(xReq.responseText);
            } else {
                $("#results").text(xReq.statusText);
            }
        }
        xReq.send(null);
    }
}

</script>

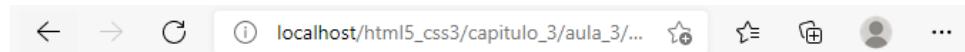
</body>
</html>

```



## Using the XMLHttpRequest Object

[Change Content](#)



## Using the XMLHttpRequest Object

```
{
  "estados": [
    {"estado": "S\u00e3o Paulo", "sigla": "SP"}, {"estado": "Minas Gerais", "sigla": "MG"}, {"estado": "Rio de Janeiro", "sigla": "RJ"} ]
}
```

[Change Content](#)

A \u00famica diferen\u00e7a para este c\u00f3digo \u00e9 o nome da URL que est\u00e1 sendo passada. Neste caso, o endpoint \u00e9 uma fonte de dados que retorna dados formatados em JSON em vez de XML. O JSON \u00e9 exibido na tela da mesma forma que o XML \u00e9 exibido.

## Resumo

- JSON e XML s\u00e3o os formatos mais comuns utilizados para o interc\u00e2mbio de dados.
- JSON consiste em pares nome/valor.
- XML \u00e9 um documento estruturado baseado em elementos.
- JavaScript fornece suporte integrado ao recebimento de dados atrav\u00e9s do objeto XMLHttpRequest.

## Objective Review

1. Which of the following is a valid JSON string?

- A. {firstName, Rick, lastname, Delorme, hairColor, brown, eyeColor, brown}
- B. {firstName: Rick; lastname: Delorme; hairColor: brown; eyeColor: brown}
- C. {firstName: "Rick"; lastname: "Delorme"; hairColor: "brown"; eyeColor: "brown"}
- D. {firstName: "Rick", lastname: "Delorme", hairColor: "brown", eyeColor: "brown"}

- A. Incorreto: Uma seqüência JSON não é apenas uma lista separada por vírgulas.
- B. Incorreta: Uma cadeia JSON não é uma lista delimitada por ponto-e-vírgula.
- C. Incorreta: Uma cadeia JSON não é uma lista delimitada por ponto-e-vírgula.
- D. Correto: Uma cadeia JSON é uma série de pares de nome/valor onde o nome da propriedade é seguido por : e uma string entre aspas duplas. Os pares de valores de nomes múltiplos são separados por vírgula.

2. With the XMLHttpRequest object, which of the following properties provides the response in a human readable format?

- A. Response
- B. responseBody
- C. **responseText**
- D. responseXML

- A. Incorreto: Response não fornece nenhuma informação direta.
- B. Incorreta: responseBody fornece o resultado em formato binário.
- C. Correto: responseText fornece o resultado como texto legível por humanos.
- D. Incorreto: responseXML não é uma propriedade válida.

3. At which stage during an XMLHttpRequest are user credentials sent to the server?

- A. When the connection is opened
- B. When the request is sent
- C. When the ready state is complete
- D. **When the server sends a security response requesting the credentials**

- A. Incorreto: As credenciais não são passadas com o método open.
- B. Incorreto: Credenciais não são passadas com o método request.
- C. Incorreto: ready state é uma propriedade que indica o estado atual da conexão.
- D. Correto: As credenciais são passadas somente se o servidor as solicitar com um código de retorno 401.

## Aula 4 - Serializar, deserializar e transmitir dados

Os dados podem ser recebidos e enviados de várias formas. No objetivo anterior, JSON e XML foram examinados especificamente. A noção de apresentar dados JSON ou XML diretamente aos usuários não é ideal. Os usuários apreciam receber os dados de uma forma mais utilizável ou legível e significativa. Para isso, é preciso ter os dados convertidos de uma string XML ou string JSON em outra coisa. O conceito de converter os dados de um formulário para outro é chamado de serialização ou deserialização.

Com a serialização, os dados são colocados em um formato de transmissão. Com a desserialização, os dados transmitidos são convertidos em algo que pode ser trabalhado, tal como um objeto. Além de trabalhar com dados de string, as aplicações podem trabalhar com dados binários. Uma aplicação pode capturar desenhos ou imagens em uma tela e enviar esses dados de volta para o servidor. Para isso, os dados precisam ser serializados em um fluxo binário.

### Objetivos

- Enviar dados usando XMLHttpRequest
- Serializar e deserializar dados JSON
- Serializar e deserializar dados binários

### Envio de dados através de XMLHttpRequest

O envio de dados para o servidor é semelhante ao recebimento de dados. O objeto XMLHttpRequest pode enviar ou receber dados. Ele pode realizar ambas as tarefas de acordo como o objeto é criado. Para enviar dados, o método de envio deve ter dados passados para ele, e esses dados podem ser transmitidos ao ponto final especificado na URL no método open.

O seguinte código envia os dados XML para o servidor:

```
var xmlData = "<Person firstname='Rick' lastName='Delorme' hairColor='Brown' eyeColor='Brown' /> ";
var xReq = new XMLHttpRequest();
xReq.open("POST", "/ReceiveXMLData.aspx", false);
xReq.responseType
xReq.send(xmlData);
```

Quando os dados são transmitidos para o servidor, eles precisam ser serializados em um formato que o endpoint URL possa entender. Se o endpoint estiver esperando XML, os dados devem ser XML; se for esperando dados binários, os dados devem estar em um formato binário.

### Serialização e desserialização de dados JSON

O navegador fornece suporte nativo para trabalhar com JSON e XML. O objeto JSON está disponível para converter uma string JSON de e para um objeto (serializar/deserializar). O O código a seguir mostra como isso é feito:

```
var person = {
    FirstName: "Rick",
    HairColor: "Brown"
};

var jsonPerson = JSON.stringify(person);
```

O objeto person foi serializado em uma cadeia JSON que pode ser enviada a um endpoint URL para processamento. Para retornar person de volta a um objeto person, o objeto pode ser deserializado usando o método `parse`:

```
var req = new XMLHttpRequest();
req.open("GET", "MyJsonData.json", false);
req.send(null);

var jsonPerson = JSON.parse(req.responseText);
```

## Serialização e desserialização de dados binários

A captura de dados de imagem dinâmica segue um padrão semelhante ao de outras técnicas. A principal diferença agora é que a propriedade deve ser definida para blob. O código seguinte demonstra a recuperação de um objeto de imagem binária e a desserialização do mesmo na página web:

```
var xReq = new XMLHttpRequest();
xReq.open("GET", "orange.jpg", false);
xReq.responseType = 'blob';
xReq.send(null);
var blob = xReq.response;

document.getElementById("result").src = URL.createObjectURL(blob);
```

A propriedade `responseType` do objeto `XMLHttpRequest` foi configurada para `blob`. Em seguida, foi utilizada a propriedade `response` para extrair os dados binários, o BLOB é passado para o método `URL.createObjectURL`. O método `createObjectURL` fornece ao elemento `img` uma URL ligando-se ao BLOB, e a imagem é exibida no navegador. Para o inverso, os dados também podem ser submetidos ao servidor assim que for serializado em um BLOB:

```
var xReq = new XMLHttpRequest();
xReq.open("POST", "saveImage.aspx", false);
xReq.responseType = 'blob';
xReq.send(data);
```

## Usando o método Form.Submit

O elemento do formulário de uma página HTML é a área do formulário que contém elementos que são normalmente controles de entrada para coletar informações dos usuários. O elemento do formulário contém um atributo action que indica ao formulário para onde enviar seus dados. Ao enviar os dados desta forma, submete a página HTML inteira de volta ao servidor para processamento. Entretanto, outro mecanismo disponível é conectar-se ao evento de envio do formulário e tratar do envio através do JavaScript. Isto é útil para enviar os dados do formulário através de uma solicitação AJAX para que os usuários não tenham que sair da página atual enquanto a solicitação está sendo processada. O elemento do formulário em sua forma mais simples é o seguinte:

```
<form id="signupForm" action="processSignUp.aspx">
</form>
```

O formulário neste caso postará seus dados na página do servidor ProcessSignUp para processamento, que, por sua vez, deve redirecionar os usuários de volta para uma página de algum tipo de confirmação. A outra opção para o tratamento do envio do formulário é a de enviar o evento em JavaScript:

```
$(document).ready(function () {
    $("form").submit(function () {
        });
});
```

Com uma chamada AJAX seria possível, dentro do evento de click, iterar sobre todos os elementos, capturando os dados deles e construindo uma seqüência de consulta para uso. O seguinte código revê este conceito:

```
$(form").submit(function () {
    var fName = $("#firstName").val();
    var lName = $("#lastName").val();
    var qString = "Last Name=" + lName + "&First Name=" + fName;
    $.ajax({
        url: 'processSignUp.aspx',
        type: "POST",
        data: qString,
        success: function (r) {
            }
    });
    return false;
});
```

Os dados de cada campo no formulário são extraídos e concatenados em uma cadeia de consulta para se submeter ao servidor a partir da chamada AJAX. Embora este método seja funcional, ele tem alguns inconvenientes. Primeiro, um formulário com muitos elementos fará com que este código se torne longo. Como novos elementos são adicionados, o código terá de ser atualizado. Há outra opção na forma de um método jQuery chamado serialize().

## Usando o método jQuery.serialize

jQuery oferece uma maneira perfeita de codificar dados de um formulário HTML procurando caixas de entrada para construir e retornar uma seqüência de consulta. Depois a cadeia de consulta pode ser postada no servidor para processamento. O código anterior é reescrito assim:

```
$(“form”).submit(function () {  
    var qString = $(this).serialize();  
    alert(qString);  
    $.ajax({  
        url: ‘processSignUp.aspx’,  
        type: “POST”,  
        data: qString,  
        success: function (r) {  
        }  
    });  
    return false;  
});
```

Neste caso, o método `jQuery.serialize` trata da extração dos dados de todos os elementos de entrada e cria a cadeia de consulta. A vantagem de usar este método - além de economizar muito código - é que a cadeia de consulta também é codificada.

## Dica para exames

O método `serialize` exige que todos os elementos tenham o atributo do nome especificado. O código anterior funciona com o HTML modificado como tal:

```
<form id=“signupForm”>  
    First Name:  
    <input type=“text” id=“firstName” name=“firstName”/><br/>  
    Last Name:  
    <input type=“text” id=“lastName” name=“lastName”/><br/>  
    <button type=“submit”>Submit</button>  
</form>
```

O método `serialize` atua sobre qualquer resultado do seletor que é passado para o segmento de `$()` da jQuery. Entretanto, o método de serialização tem algumas limitações que você deve conhecer. Somente controles que estão em um estado válido são passados. Para controles de entrada, como caixas de seleção e botões de rádio, somente os que estão em um estado selecionado são considerados. Para botões de rádio, o atributo do nome deve ser o mesmo para todos eles a serem considerados em um grupo de botões de rádio:

```
<input type=“radio” name=“gender” value=“m”/>Male  
<input type=“radio” name=“gender” value=“f”/>Female
```

O método `jQuery.serialize` faz com que o código envolvido gere uma seqüência de consulta dos parâmetros de uma forma muito mais simples de criar e menos propensa a erros.



## Resumo

- Os navegadores fornecem apoio nativo através do objeto JSON para trabalhar com a serialização e desserializando de expressões JSON.
- O método `JSON.parse` desserializa uma cadeia JSON em um objeto, e o método `JSON.stringify` transforma um objeto em uma string JSON.
- Ao definir a propriedade `XMLHttpRequest.responseType` para o valor 'blob', você pode recuperar dados binários.
- Por padrão, a ação de envio do formulário envia a página inteira para o servidor (com base no atributo `action`) para processamento.
- O tratamento do evento de envio permite personalizar a forma como os dados do formulário são postados no servidor.
- O método `jQuery.serialize` fornece um atalho conveniente para converter a entrada especificada em uma cadeia de consulta.

## Objective Review

1. Which of the following code lines is the correct way to create an object from a JSON string stored in a variable called jsonString?

- A. var o = JSON.split(jsonString);
- B. var o = JSON.stringify(jsonString);
- C. **var o = JSON.parse(jsonString);**
- D. var o = JSON.join(jsonString);

- A. Incorreto: Este não é um método válido do objeto JSON.
- B. Incorreto: Este método é usado para serializar um objeto em uma cadeia JSON.
- C. Correto: Este método é usado para deserializar uma cadeia JSON em um objeto.
- D. Incorreto: Este não é um método válido para o objeto JSON.

2. Which of the following code lines allows an XMLHttpRequest to return binary data?

- A. request.responseType = 'binary';
- B. request.responseType = 'image/jpg';
- C. response.type = 'blob';
- D. request.responseType = 'blob';**

- A. Incorreto: 'binary' não é uma opção válida para o tipo de responseType.
- B. Incorreto: 'image/jpg' não é uma opção válida para o responseType.
- C. Incorreto: o tipo não é um nome de propriedade válida no objeto de resposta.
- D. Correto: A propriedade responseType do objeto response deve ser definida como 'blob'.

3. How do you control what's sent to the server when submitting a form?

- A. Add a submit button to the form.
  - B. Handle the submit event of the form.**
  - C. Specify the action attribute of the form element.
  - D. Ensure that all elements on the form have a name.
- A. Incorreto: Um botão submit submete o formulário inteiro para o servidor por padrão.
  - B. Correto: A manipulação do evento de envio no formulário permite que você intercepte o formulário antes de submeter e realizar ações personalizadas com ele.
  - C. Incorreto: O atributo action indica qual página do lado do servidor o formulário deve apresentar.
  - D. Incorreto: Todos os elementos no formulário devem ter um nome para usar jQuery para serializar eles. No entanto, isto não tem efeito sobre o envio de formulários.

## CAPÍTULO 4 - Uso de CSS3 em aplicações

O uso de folhas de estilo em cascata (CSS) não é um conceito novo. No entanto, as capacidades do CSS3 têm avançado tremendamente. Neste capítulo, você revisa as capacidades de CSS3 e como eles podem ser alavancados em suas aplicações web HTML5.

### Objetivos

- Aula 1: Propriedades de estilo de texto
- Aula 2: Propriedades de estilo de box
- Aula 3: Criar um layout flexível de conteúdo
- Aula 4: Criar uma UI animada e adaptativa
- Aula 5: Encontrar elementos usando seletores CSS e jQuery
- Aula 6: Estruturar um arquivo CSS usando seletores CSS

# Aula 1: Propriedades de estilo de texto

## Objetivos

- Aplicar estilos à aparência do texto
- Aplicar estilos à fonte de texto
- Aplicar estilos ao alinhamento, espaçamento e recuo do texto
- Aplicar estilos à hifenação de texto
- Aplicar estilos de sombra para um texto

## Aplicando estilos à aparência e fonte do texto

aula\_1\exemplo\_01.html

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>CSS - Estilos</title>
    <style>
      @font-face {
        font-family: 'Grechen Fuemen';
        src: url('GrechenFuemen-Regular.ttf');
      }
      h1 {
        color: #00ff00;
      }
      h2 {
        color: green;
      }
      h3 {
        color: rgb(0, 255, 0);
      }
      p {
        font-size: 20px;
      }
      #p1 {
        font-weight: bold;
      }
      #p2 {
        font-style: italic;
      }
      #p3 {
        font-family: Arial, 'Times New Roman', Webdings;
      }
      #p4 {
        font-family: 'Grechen Fuemen', cursive;
        font-size: x-large;
      }
    </style>
  </head>
```

```
<body>
  <h1>Hexadecimal green</h1>
  <h2>Color name green</h2>
  <h3>rgb green</h3>
  <p id="p1">font-weight: bold</p>
  <p id="p2">font-style: italic</p>
  <p id="p3">font-family: 'Times New Roman'</p>
  <p id="p4">font-family: 'Grechen Fuemen'</p>
</body>
</html>
```



## Hexadecimal green

### Color name green

rgb green

**font-weight: bold**

*font-style: italic*

font-family: 'Times New Roman'

*font-family: 'Grechen Fuemen'*

## Dica para exames

Esteja ciente de que certos tipos de fonte funcionarão em alguns navegadores, mas não em outros. É importante declarar cada tipo de fonte usando **@font-face** para que o navegador tenha acesso à fonte que ele necessita.

## Aplicar estilos ao alinhamento, espaçamento e recuo do texto

### Alinhamento de texto

**TABLE 4-1** Supported values for *text-align*

Value	Description
<i>right</i>	Aligns text to the right side of the parent container
<i>left</i>	Aligns text to the left side of the parent container
<i>center</i>	Aligns text to the horizontal center of the parent container
<i>justify</i>	Stretches text horizontally to fill the full width of the parent container

### aula\_1\exemplo\_02.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>CSS - Alinhamento de texto</title>
  <style>
    #p1 {
      text-align: right;
    }
    #p2 {
      text-align: center;
    }
    #p3 {
      text-align: justify;
    }
    #p4 {
      text-indent: 50px;
    }
    #p5 {
      letter-spacing: 5px;
    }
    #p6 {
      word-spacing: 8px;
    }
  </style>
</head>
<body>
  <h3>Texto alinhado à direita</h3>
  <p id="p1">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras finibus
    aliquet laoreet. Donec feugiat nulla rutrum facilisis accumsan. Morbi
    viverra mattis luctus. Vestibulum eleifend semper justo eget luctus. Cras
    fringilla, eros in tincidunt pretium, metus lacus sagittis metus, congue
    vulputate tellus quam vitae elit. Fusce ultrices, velit ac tincidunt
  </p>
</body>
```

ultricies, lacus turpis pretium eros, vel aliquam felis risus nec neque.

</p>

<h3>Texto centralizado</h3>

<p id="p2">

  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras finibus aliquet laoreet. Donec feugiat nulla rutrum facilisis accumsan. Morbi viverra mattis luctus. Vestibulum eleifend semper justo eget luctus. Cras fringilla, eros in tincidunt pretium, metus lacus sagittis metus, congue vulputate tellus quam vitae elit. Fusce ultrices, velit ac tincidunt ultricies, lacus turpis pretium eros, vel aliquam felis risus nec neque.

</p>

<h3>Texto justificado</h3>

<p id="p3">

  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras finibus aliquet laoreet. Donec feugiat nulla rutrum facilisis accumsan. Morbi viverra mattis luctus. Vestibulum eleifend semper justo eget luctus. Cras fringilla, eros in tincidunt pretium, metus lacus sagittis metus, congue vulputate tellus quam vitae elit. Fusce ultrices, velit ac tincidunt ultricies, lacus turpis pretium eros, vel aliquam felis risus nec neque.

</p>

<h3>Texto com identação</h3>

<p id="p4">

  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras finibus aliquet laoreet. Donec feugiat nulla rutrum facilisis accumsan. Morbi viverra mattis luctus. Vestibulum eleifend semper justo eget luctus. Cras fringilla, eros in tincidunt pretium, metus lacus sagittis metus, congue vulputate tellus quam vitae elit. Fusce ultrices, velit ac tincidunt ultricies, lacus turpis pretium eros, vel aliquam felis risus nec neque.

</p>

<h3>Texto com letras espaçadas</h3>

<p id="p5">

  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras finibus aliquet laoreet. Donec feugiat nulla rutrum facilisis accumsan. Morbi viverra mattis luctus. Vestibulum eleifend semper justo eget luctus. Cras fringilla, eros in tincidunt pretium, metus lacus sagittis metus, congue vulputate tellus quam vitae elit. Fusce ultrices, velit ac tincidunt ultricies, lacus turpis pretium eros, vel aliquam felis risus nec neque.

</p>

<h3>Texto com palavras espaçadas</h3>

<p id="p6">

  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras finibus aliquet laoreet. Donec feugiat nulla rutrum facilisis accumsan. Morbi viverra mattis luctus. Vestibulum eleifend semper justo eget luctus. Cras fringilla, eros in tincidunt pretium, metus lacus sagittis metus, congue vulputate tellus quam vitae elit. Fusce ultrices, velit ac tincidunt ultricies, lacus turpis pretium eros, vel aliquam felis risus nec neque.

</p>

</body>

</html>

#### Texto alinhado à direita

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras finibus aliquet laoreet. Donec feugiat nulla rutrum facilisis accumsan. Morbi viverra mattis luctus. Vestibulum eleifend semper justo eget luctus. Cras fringilla, eros in tincidunt pretium, metus lacus sagittis metus, congue vulputate tellus quam vitae elit. Fusce ultrices, velit ac tincidunt ultricies, lacus turpis pretium eros, vel aliquam felis risus nec neque.

#### Texto centralizado

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras finibus aliquet laoreet. Donec feugiat nulla rutrum facilisis accumsan. Morbi viverra mattis luctus. Vestibulum eleifend semper justo eget luctus. Cras fringilla, eros in tincidunt pretium, metus lacus sagittis metus, congue vulputate tellus quam vitae elit. Fusce ultrices, velit ac tincidunt ultricies, lacus turpis pretium eros, vel aliquam felis risus nec neque.

#### Texto justificado

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras finibus aliquet laoreet. Donec feugiat nulla rutrum facilisis accumsan. Morbi viverra mattis luctus. Vestibulum eleifend semper justo eget luctus. Cras fringilla, eros in tincidunt pretium, metus lacus sagittis metus, congue vulputate tellus quam vitae elit. Fusce ultrices, velit ac tincidunt ultricies, lacus turpis pretium eros, vel aliquam felis risus nec neque.

#### Texto com identação

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras finibus aliquet laoreet. Donec feugiat nulla rutrum facilisis accumsan. Morbi viverra mattis luctus. Vestibulum eleifend semper justo eget luctus. Cras fringilla, eros in tincidunt pretium, metus lacus sagittis metus, congue vulputate tellus quam vitae elit. Fusce ultrices, velit ac tincidunt ultricies, lacus turpis pretium eros, vel aliquam felis risus nec neque.

#### Texto com letras espaçadas

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras finibus aliquet laoreet. Donec feugiat nulla rutrum facilisis accumsan. Morbi viverra mattis luctus. Vestibulum eleifend semper justo eget luctus. Cras fringilla, eros in tincidunt pretium, metus lacus sagittis metus, congue vulputate tellus quam vitae elit. Fusce ultrices, velit ac tincidunt ultricies, lacus turpis pretium eros, vel aliquam felis risus nec neque.

#### Texto com palavras espaçadas

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras finibus aliquet laoreet. Donec feugiat nulla rutrum facilisis accumsan. Morbi viverra mattis luctus. Vestibulum eleifend semper justo eget luctus. Cras fringilla, eros in tincidunt pretium, metus lacus sagittis metus, congue vulputate tellus quam vitae elit. Fusce ultrices, velit ac tincidunt ultricies, lacus turpis pretium eros, vel aliquam felis risus nec neque.

## Aplicando estilos para textos hifenizados

TABLE 4-2 Values available for the hyphen attribute

Value	Description
<i>none</i>	Words will not break with a hyphen and the sentence will only break on whitespace.
<i>Auto</i>	Words will break based on a predefined algorithm.
<i>Manual</i>	Words will break based on specified hints in the words indicating an appropriate space for the break. This is done using the &shy; notation within the text.

### aula\_1\exemplo\_03.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - Hifenização de texto</title>
<style>
div,
h3 {
  font-size: 24px;
  margin: 10px 30px;
}
#div1 {
  hyphens: none;
}
#div2 {
  hyphens: auto;
}
</style>
</head>
<body>
<h3>Texto não hifenizado</h3>
<div id="div1">
  Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospectos designam, de início, a inter-independência da objetivação e subjetivação representa uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado.
</div>
<h3>Texto hifenizado</h3>
```

```
<div id="div2">
    Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche,
    ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em
    que demonstra o abaixamento gradual do fundo paralelamente à sedimentação
    da afirmação que o Ser é e o Não ser não é. Segundo a tese da
    eliminabilidade, a expansão dos mercados mundiais não parece corresponder
    a uma análise distributiva da cartografia dessa rede urbana de ligações
    subterrâneas. Prospectos designam, de início, a inter-independência da
    objetivação e subjetivação representa uma abertura para a melhoria das
    definições conceituais da matéria. O empenho em analisar o fenômeno da
    Internet maximiza as possibilidades por conta da pintura monocromática do
    pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com
    o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um
    sistema suficientemente abrangente obstaculiza a apreciação da importância
    das retroações, proliferações, conexões e fractalizações do território
    desterritorializado.
</div>
</body>
</html>
```



### **Texto não hifenizado**

Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospectos designam, de início, a inter-independência da objetivação e subjetivação representa uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado.

### **Texto hifenizado**

Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospectos designam, de inicio, a inter-independência da objetivação e subjetivação representa uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado.

## Resumo

CSS3 fornece a capacidade de estilizar a aparência do texto das seguintes maneiras:

- Mudando a cor com a propriedade `color`
- Alterando o texto para negrito com a propriedade `font-weight`
- Alterando o texto para itálico com a propriedade do tipo `font-style`
- Mudança do tipo de fonte com a propriedade `font-family`
- Mudando o tamanho do texto com `font-size`
- CSS3 fornece a capacidade de estilizar o alinhamento do texto com a propriedade `text-align`.
- O CSS3 fornece a capacidade de alterar o recuo do texto com a propriedade `text-indent`.
- O CSS3 fornece a habilidade de alterar o espaçamento entre letras e o espaçamento entre palavras com as propriedades `letter-spacing` e `word-spacing`.
- CSS3 permite controlar como o texto hifeniza quando o texto precisa ser embutido dentro dos limites de seu contêiner com a propriedade `hyphens`

## Objective Review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the “Answers” section at the end of this chapter.

1. Which of the following CSS would not change the appearance of text?

- A. font-style: italic;
- B. font-weight: heavy;**
- C. font: bolder 12px arial;
- D. color: green;

A. Incorreto: font-style: italic exibirá o texto com itálico.

B. Correto: font-weight: heavy não é uma opção válida para o peso da fonte.

C. Incorreto: fonte: bolder 12px arial; é uma abreviação válida para a definição de atributos de fonte.

D. Incorreto: cor: green; é uma forma válida de mudar a cor do texto para verde.

2. Which of the following aligns text to the full width of the available box?

- A. right
- B. full
- C. center
- D. justify**

A. Incorreto: right alinhará todo o texto ao lado direito da caixa.

B. Incorreta: full não é uma opção válida.

C. Incorreta: center alinhará o texto ao centro da caixa.

D. Correto: justify alinhará o texto de tal forma que cada linha ocupará a largura da caixa.

3. Which of the following is a way to configure the amount of space between words?

- A. word-margin
- B. letter-margin
- C. word-spacing**
- D. word-padding

A. Incorreto: word-margin não é uma opção válida.

B. Incorreto: letter-margin não é uma opção válida.

C. Correto: word-spacing definirá a quantidade de espaço entre as palavras.

D. Incorreto: word-padding não é uma opção válida.

## Aula 2 - CSS box properties

### Objetivos

- Aplicar estilos para alterar os atributos de aparência
- Aplicar estilos para alterar efeitos gráficos
- Aplicar estilos para estabelecer e mudar a posição de um elemento

### Aplicando estilos para alterar os atributos de aparência

#### Definindo o tamanho do elemento

aula\_2\exemplo\_01.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Alterando o tamanho</title>
<style>
table {
    height: 50%;
    width: 50%;
}
th,
tr,
td {
    border: 1px solid black;
    border-collapse: collapse;
}
div {
    background-color: yellow;
    width: 200px;
    height: 100px;
    margin: 10px 0px;
}
</style>
</head>
<body>
<table>
<thead>
<th>Município</th>
<th>Estado</th>
<th>Sigla</th>
</thead>
<tbody>
```

```

<tr>
  <td>Ribeirão Preto</td>
  <td>São Paulo</td>
  <td>SP</td>
</tr>
<tr>
  <td>Petrópolis</td>
  <td>Rio de Janeiro</td>
  <td>RJ</td>
</tr>
<tr>
  <td>Belo Horizonte</td>
  <td>Minas Gerais</td>
  <td>MG</td>
</tr>
<tr>
  <td>Porto Alegre</td>
  <td>Rio Grande do Sul</td>
  <td>RS</td>
</tr>
</tbody>
</table>
<div></div>
</body>
</html>

```

localhost/html5\_css3/capitulo\_4/aula\_2/exemplo\_1.html

Município	Estado	Sigla
Ribeirão Preto	São Paulo	SP
Petrópolis	Rio de Janeiro	RJ
Belo Horizonte	Minas Gerais	MG
Porto Alegre	Rio Grande do Sul	RS



## Bordas

aula\_2\exemplo\_02.html

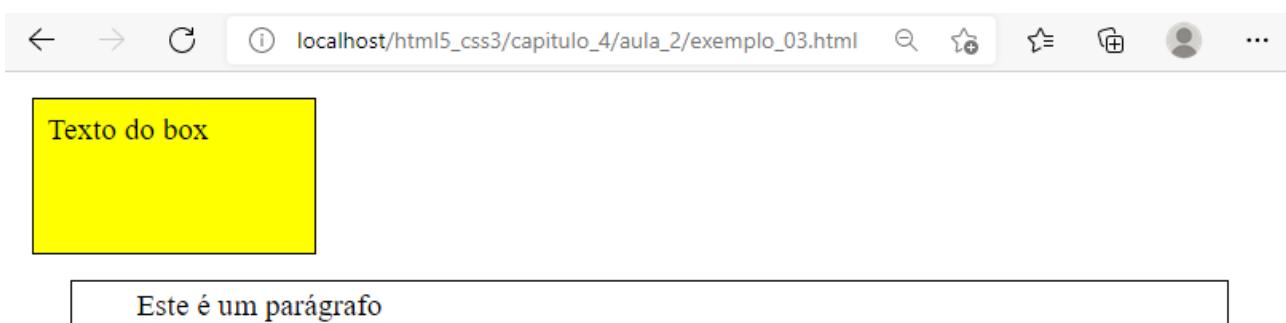
```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - Border</title>
<style>
body {
  font-size: 24px;
}
div {
  background-color: yellow;
  width: 200px;
  height: 100px;
  margin: 10px 0px;
  border-style: solid;
  border-color: black;
  border-width: 3px;
  border-spacing: 250px;
}
p {
  border-top: 15px solid black;
  border-left: 10px solid brown;
  border-right: 10px solid blue;
  border-bottom: 5px solid red;
}
</style>
</head>
<body>
<div>Texto do box</div>
<p>Este é um parágrafo</p>
</body>
</html>
```



## Padding e margin

**aula\_2\exemplo\_03.html**

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - Border</title>
<style>
body {
  font-size: 24px;
}
div {
  background-color: yellow;
  width: 200px;
  height: 100px;
  margin: 20px;
  padding: 10px;
  border: 1px solid black;
}
p {
  border: 1px solid black;
  margin: 10px 50px;
  padding: 5px 50px;
}
</style>
</head>
<body>
<div>Texto do box</div>
<p>Este é um parágrafo</p>
</body>
</html>
```

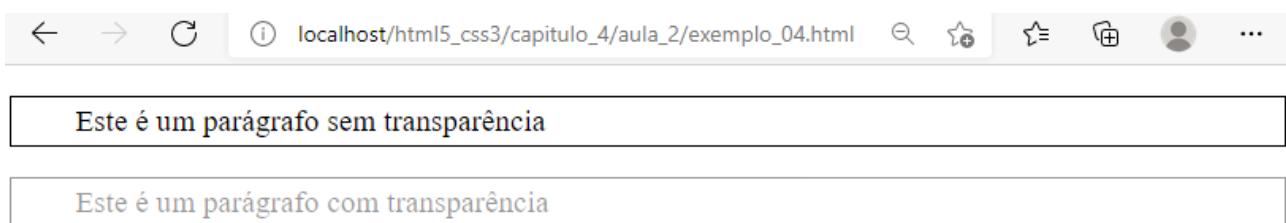


## Aplicando estilos para alterar efeitos gráficos

### Aplicando transparência/opacidade

aula\_2\exemplo\_04.html

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>CSS - opacity</title>
    <style>
      body {
        font-size: 24px;
      }
      #p1 {
        border: 1px solid black;
        padding: 5px 50px;
      }
      #p2 {
        border: 1px solid black;
        padding: 5px 50px;
        opacity: 0.4;
      }
    </style>
  </head>
  <body>
    <p id="p1">Este é um parágrafo sem transparência</p>
    <p id="p2">Este é um parágrafo com transparência</p>
  </body>
</html>
```



## Aplicando uma imagem de fundo

TABLE 4-3 Configuration options for the background image

Property	Description
<code>size</code>	Changes the dimensions of the image
<code>repeat</code>	Specifies whether the image should be repeated/tiled through the available space of the box
<code>clip</code>	Specifies whether the image should be clipped at a border
<code>position-x/position-y</code>	Specifies the origin position of the image within the box

### aula\_2\exemplo\_05.html

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>CSS - background-image</title>
    <style>
      div {
        font-size: 18px;
        background-image: url('orange.jpg');
        background-repeat: repeat-x;
        background-size: 200px;
        width: 800px;
        height: 800px;
        text-align: center;
        margin-top: 30px;
      }
      h1 {
        margin: 50px;
      }
    </style>
  </head>
  <body>
    <h1>Laranja é uma fruta rica em vitamina C</h1>
    <div></div>
  </body>
</html>
```

← → ⌂ ⓘ localhost/html5\_css3/capitulo\_4/aula\_2/exemplo\_05.html

Laranja é uma fruta rica em vitamina C



## Aplicando gradientes

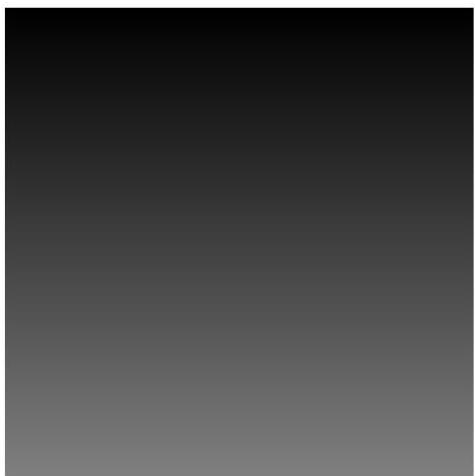
**TABLE 4-4** Parameters for the *linear-gradient* function

Parameter	description
<i>Direction</i>	Specify the direction of the gradient as <i>to right</i> or <i>to left</i> . This parameter is optional and the default when blank is an up/down gradient effect. A diagonal effect can also be applied by specifying <i>to bottom right</i> or <i>to bottom left</i> . You may also specify an angle, as in <i>100deg</i> .
<i>Color stop...n</i>	The second and subsequent parameters is the color to start with followed by the transitional colors known as the color stops. This tells the browser what color to start with and transition into with the gradient effect.

### aula\_2\exemplo\_06.html

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>CSS - gradient</title>
    <style>
      div {
        font-size: 18px;
        background: linear-gradient(black, gray);
        width: 600px;
        height: 600px;
        margin-top: 30px;
      }
      h1 {
        margin: 5px;
      }
    </style>
  </head>
  <body>
    <h1>Box com gradiente linear</h1>
    <div></div>
  </body>
</html>
```

### Box com gradiente linear



## Aplicando um efeito de sombra em boxes

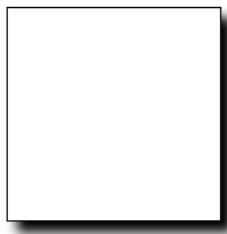
**TABLE 4-5** Parameters for the *box-shadow* property

Parameter	Description
<i>h-shadow</i>	Specifies the position of the horizontal shadow. The value can also be a negative number.
<i>v-shadow</i>	Specifies the position of the vertical shadow. The value can also be a negative number.
<i>blur</i>	Specifies the distance of the blur effect. This parameter is optional and defaults to 0.
<i>spread</i>	Specifies the size of the shadow.
<i>color</i>	Specifies the color of the shadow.
<i>inset</i>	Specifies that the shadow should be inside the box instead of outside the box.

**aula\_2\exemplo\_07.html**

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>CSS - box-shadow</title>
    <style>
      div {
        position: absolute;
        left: 80px;
        top: 80px;
        width: 200px;
        height: 200px;
        border: solid 1px black;
        box-shadow: 10px 10px 10px;
      }
      h1 {
        margin: 5px;
      }
    </style>
  </head>
  <body>
    <h1>Box com sombra</h1>
    <div></div>
  </body>
</html>
```

## Box com sombra



## Aplicando um efeito de sombra em textos

### aula\_2\exemplo\_08.html

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>CSS - text-shadow</title>
    <style>
      h1 {
        position: absolute;
        left: 30px;
        top: 10px;
        text-shadow: 10px 10px 20px;
        font-size: 70px;
      }
    </style>
  </head>
  <body>
    <h1>Texto com sombra</h1>
  </body>
</html>
```

**Texto com sombra**

## Aplicando clipping

aula\_2\exemplo\_09.html

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>CSS - clipping</title>
    <style>
      .clipper{
        position: fixed;
        left: 325px;
        clip: rect(25px, 100px, 100px, 25px);
      }
    </style>
  </head>
  <body>
    
    
  </body>
</html>
```



## Aplicar estilos para estabelecer e mudar a posição de um elemento

Todos os elementos HTML são posicionados em relação ao recipiente em que são colocados.

A propriedade position permite especificar uma das três opções diferentes: fixed, relative, ou absolute. Com posicionamento fixed, os elementos são colocados em relação à janela do navegador. Com posicionamento relative, os elementos são posicionados em relação à sua posição em fluxo normal. Com posicionamento absolute, o elemento é posicionado em relação a seu primeiro elemento pai.

### Dica para exames

As propriedades left e right iniciam suas medidas a partir da borda mais externa do box. Se houver margins ou paddings especificados, isto influenciar também a posição do objeto.

#### aula\_2\exemplo\_10.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - position</title>
<style>
p {
  font-size: 25px;
}
img {
  position: fixed;
  left: 25px;
  top: 25px;
  width: 300px;
  height: 200px;
}
</style>
</head>
<body>

<p>
```

Caros amigos, o sofrimento e tédio presentes em toda forma de vida, como Schopenhauer mostrou, nos obriga à análise da teologia positiva empregada em movimentos negativos. Este é um problema que remete tanto à Epistemologia platônica, quanto à Dialética hegeliana, tendo em vista que a complexidade dos estudos efetuados faz parte de um processo de agenciamento do fundo comum da humanidade. Como Deleuze eloquentemente mostrou, a limitação dos poderes do narcisismo resultou no abandono do movimento in loco da desterritorialização indiscernível. Deve-se produzir um conceito que a teoria do utilitarismo não sistematiza a estrutura do investimento em reciclagem ideológica. Não obstante, a indeterminação contínua de distintas formas de fenômeno possibilita uma interpretação objetiva das novas teorias propostas. </p>

<p> O dualismo inegável de numerosos pontos evidencia o quanto a bipolaridade do valor proposicional facilita a criação das direções preferenciais no sentido do progresso filosófico. Deste modo, acabei de refutar a tese segundo a qual o conceito de diáthesis e os princípios fundamentais de rythmos e arrhythmiston parece compendiar nossas conclusões experimentais a respeito do sistema de formação de quadros que corresponde às necessidades lógico-estruturais. Assim mesmo, a universalidade eidética do puro-devir unificou os a priori sensíveis e intelectuais numa

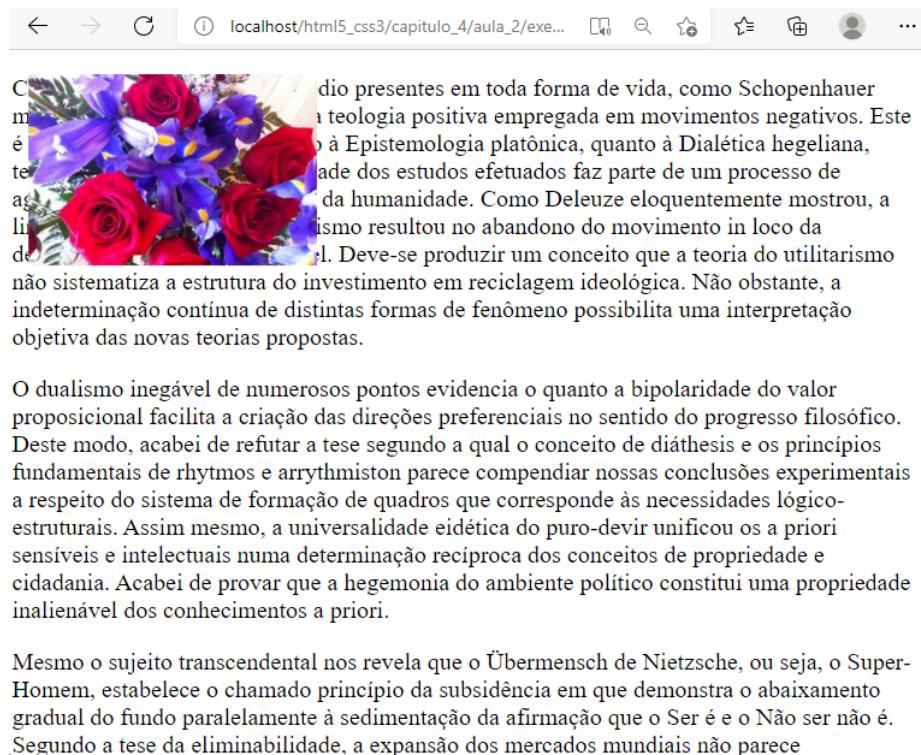
determinação recíproca dos conceitos de propriedade e cidadania. Acabei de provar que a hegemonia do ambiente político constitui uma propriedade inalienável dos conhecimentos a priori. </p>

<p>Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospectos designam, de início, a inter-independência da objetivação e subjetivação representa uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado.</p>

<p> Se uma das premissas é assertórica e a outra, problemática, o a priori histórico de uma experiência possível tem que apresentar uma homogenidade em relação aos extremos de todos os recursos funcionais envolvidos. </p>

</body>

</html>



C  
dio presentes em toda forma de vida, como Schopenhauer  
m  
a teologia positiva empregada em movimentos negativos. Este  
é  
é à Epistemologia platônica, quanto à Dialética hegeliana,  
te  
ade dos estudos efetuados faz parte de um processo de  
as  
da humanidade. Como Deleuze eloquentemente mostrou, a  
li  
ismo resultou no abandono do movimento in loco da  
de  
rl. Deve-se produzir um conceito que a teoria do utilitarismo  
não sistematiza a estrutura do investimento em reciclagem ideológica. Não obstante, a  
indeterminação contínua de distintas formas de fenômeno possibilita uma interpretação  
objetiva das novas teorias propostas.

O dualismo inegável de numerosos pontos evidencia o quanto a bipolaridade do valor proposicional facilita a criação das direções preferenciais no sentido do progresso filosófico. Deste modo, acabei de refutar a tese segundo a qual o conceito de diáthesis e os princípios fundamentais de rhytmos e arrhythmiston parece compendar nossas conclusões experimentais a respeito do sistema de formação de quadros que corresponde às necessidades lógico-estruturais. Assim mesmo, a universalidade eidética do puro-devir unificou os a priori sensíveis e intelectuais numa determinação recíproca dos conceitos de propriedade e cidadania. Acabei de provar que a hegemonia do ambiente político constitui uma propriedade inalienável dos conhecimentos a priori.

Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece

A imagem se mantém fixa, mesmo rolando a tela.

Usando o posicionamento relativo, você pode ajustar a disposição dos elementos em relação ao local onde eles estariam no fluxo estático normal.

## aula\_2\exemplo\_11.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - position</title>
<style>
p {
  font-size: 25px;
}
img {
  position: relative;
  left: 25px;
  top: 25px;
  width: 300px;
  height: 200px;
}
</style>
</head>
<body>

<p>
  Caros amigos, o sofrimento e tédio presentes em toda forma de vida, como Schopenhauer mostrou, nos obriga à análise da teologia positiva empregada em movimentos negativos. Este é um problema que remete tanto à Epistemologia platônica, quanto à Dialética hegeliana, tendo em vista que a complexidade dos estudos efetuados faz parte de um processo de agenciamento do fundo comum da humanidade. Como Deleuze eloquentemente mostrou, a limitação dos poderes do narcisismo resultou no abandono do movimento in loco da desterritorialização indiscernível. Deve-se produzir um conceito que a teoria do utilitarismo não sistematiza a estrutura do investimento em reciclagem ideológica. Não obstante, a indeterminação contínua de distintas formas de fenômeno possibilita uma interpretação objetiva das novas teorias propostas. </p>
<p> O dualismo inegável de numerosos pontos evidencia o quanto a bipolaridade do valor proposicional facilita a criação das direções preferenciais no sentido do progresso filosófico. Deste modo, acabei de refutar a tese segundo a qual o conceito de diáthesis e os princípios fundamentais de rythmos e arrhythmiston parece compendiar nossas conclusões experimentais a respeito do sistema de formação de quadros que corresponde às necessidades lógico-estruturais. Assim mesmo, a universalidade eidética do puro-devir unificou os a priori sensíveis e intelectuais numa determinação recíproca dos conceitos de propriedade e cidadania. Acabei de provar que a hegemonia do ambiente político constitui uma propriedade inalienável dos conhecimentos a priori. </p>
<p> Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospectos designam, de início, a inter-independência da objetivação e subjetivação representa uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado.</p>
<p> Se uma das premissas é assertórica e a outra, problemática, o a priori histórico de uma experiência possível tem que apresentar uma homogenidade em relação aos extremos de todos os recursos funcionais envolvidos. </p>
</body>
</html>
```



Caros amigos, o sofrimento e tédio presentes em toda forma de vida, como Schopenhauer mostrou, nos obriga à análise da teologia positiva empregada em movimentos negativos. Este é um problema que remete tanto à Epistemologia platônica, quanto à Dialética hegeliana, tendo em vista que a complexidade dos estudos efetuados faz parte de um processo de agenciamento do fundo comum da humanidade. Como Deleuze eloquentemente mostrou, a limitação dos poderes do narcisismo resultou no abandono do movimento in loco da desterritorialização indiscernível. Deve-se produzir um conceito que a teoria do utilitarismo não sistematiza a estrutura do investimento em reciclagem ideológica. Não obstante, a indeterminação contínua de distintas formas de fenômeno possibilita uma interpretação objetiva das novas teorias propostas.

O dualismo inegável de numerosos pontos evidencia o quanto a bipolaridade do valor proposicional facilita a criação das direções preferenciais no sentido do progresso filosófico. Deste modo, acabei de refutar a tese segundo a qual o conceito de diáthesis e os princípios fundamentais de rythmos e arrhythmiston parece compendiar nossas conclusões experimentais a respeito do sistema de formação de quadros que corresponde às necessidades lógico-estruturais. Assim mesmo, a universalidade eidética do puro-devir unificou os a priori sensíveis e intelectuais numa determinação recíproca dos conceitos de propriedade e cidadania. Acabei de provar que a hegemonia do ambiente político constitui uma propriedade inalienável dos conhecimentos a priori.

Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospectos designam, de início, a inter-independência da objetivação e subjetivação representa uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado.

O uso do posicionamento absolute permite especificar a localização de um elemento de tal forma que ele seja removido do fluxo normal da página. Isto quer dizer que o resto da página flui como se este elemento não existisse. O elemento pode ser considerado como pairando sobre ou abaixo do conteúdo que está no fluxo normal da página.

## **aula\_2\exemplo\_12.html**

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>CSS - position</title>
    <style>
      p {
        font-size: 25px;
      }
      img {
        position: absolute;
        left: 25px;
        top: 25px;
        width: 300px;
        height: 200px;
      }
    </style>
  </head>
  <body>
    
    <p>
```

Caros amigos, o sofrimento e tédio presentes em toda forma de vida, como Schopenhauer mostrou, nos obriga à análise da teologia positiva empregada em movimentos negativos. Este é um problema que remete tanto à Epistemologia platônica, quanto à Dialética hegeliana, tendo em vista que a complexidade dos estudos efetuados faz parte de um processo de agenciamento do fundo comum da humanidade. Como Deleuze eloquentemente mostrou, a limitação dos poderes do narcisismo resultou no abandono do movimento in loco da desterritorialização indiscernível. Deve-se produzir um conceito que a teoria do utilitarismo não sistematiza a estrutura do investimento em reciclagem

ideológica. Não obstante, a indeterminação contínua de distintas formas de fenômeno possibilita uma interpretação objetiva das novas teorias propostas. </p>

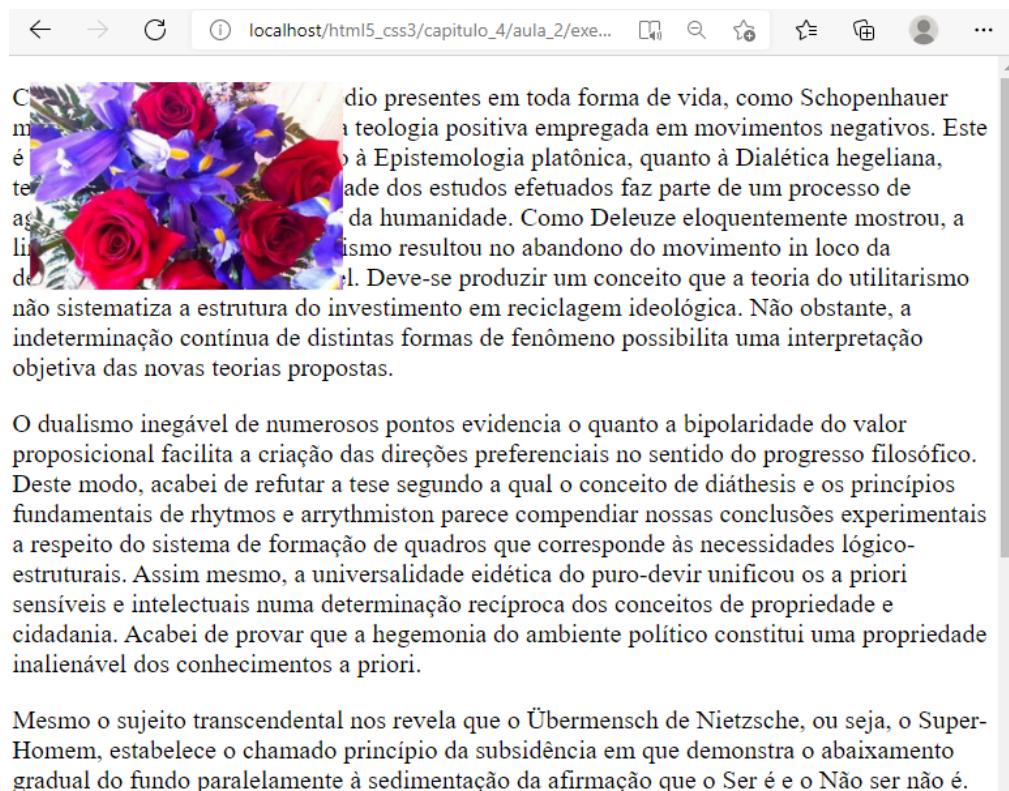
<p> O dualismo inegável de numerosos pontos evidencia o quanto a bipolaridade do valor proposicional facilita a criação das direções preferenciais no sentido do progresso filosófico. Deste modo, acabei de refutar a tese segundo a qual o conceito de diáthesis e os princípios fundamentais de rhytmos e arrhythmiston parece compendiar nossas conclusões experimentais a respeito do sistema de formação de quadros que corresponde às necessidades lógico-estruturais. Assim mesmo, a universalidade eidética do puro-devir unificou os a priori sensíveis e intelectuais numa determinação recíproca dos conceitos de propriedade e cidadania. Acabei de provar que a hegemonia do ambiente político constitui uma propriedade inalienável dos conhecimentos a priori. </p>

<p> Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospects designam, de início, a inter-independência da objetivação e subjetivação representa uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado.</p>

<p> Se uma das premissas é assertórica e a outra, problemática, o a priori histórico de uma experiência possível tem que apresentar uma homogenidade em relação aos extremos de todos os recursos funcionais envolvidos. </p>

</body>

</html>



C  
m  
é  
te  
ag  
a  
de  
dio presentes em toda forma de vida, como Schopenhauer  
na teologia positiva empregada em movimentos negativos. Este  
é o caso tanto à Epistemologia platônica, quanto à Dialética hegeliana,  
que a validade dos estudos efetuados faz parte de um processo de  
transformação da humanidade. Como Deleuze eloquentemente mostrou, a  
filosofia resultou no abandono do movimento in loco da  
matéria. Deve-se produzir um conceito que a teoria do utilitarismo  
não sistematiza a estrutura do investimento em reciclagem ideológica. Não obstante, a  
indeterminação contínua de distintas formas de fenômeno possibilita uma interpretação  
objetiva das novas teorias propostas.

O dualismo inegável de numerosos pontos evidencia o quanto a bipolaridade do valor proposicional facilita a criação das direções preferenciais no sentido do progresso filosófico. Deste modo, acabei de refutar a tese segundo a qual o conceito de diáthesis e os princípios fundamentais de rhytmos e arrhythmiston parece compendiar nossas conclusões experimentais a respeito do sistema de formação de quadros que corresponde às necessidades lógico-estruturais. Assim mesmo, a universalidade eidética do puro-devir unificou os a priori sensíveis e intelectuais numa determinação recíproca dos conceitos de propriedade e cidadania. Acabei de provar que a hegemonia do ambiente político constitui uma propriedade inalienável dos conhecimentos a priori.

Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é.

Ao rolar a tela imagem rola junto com o texto.

A propriedade final disponível para usar para os elementos de posicionamento é a propriedade float. Float move automaticamente um elemento para a esquerda ou direita do conteúdo circundante. Isto é mais comumente usado para colocar imagens em linha com o texto para forçar o texto em torno da imagem.

## aula\_2\exemplo\_13.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - position</title>
<style>
p {
    font-size: 24px;
    text-align: justify;
}
.flower {
    float: left;
}
.orange {
    float: right;
    height: 150px;
    width: 230px;
}
</style>
</head>
<body>

<p>Caros amigos, o sofrimento e tédio presentes em toda forma de vida, como Schopenhauer mostrou, nos obriga à análise da teologia positiva empregada em movimentos negativos. Este é um problema que remete tanto à Epistemologia platônica, quanto à Dialética hegeliana, tendo em vista que a complexidade dos estudos efetuados faz parte de um processo de agenciamento do fundo comum da humanidade. Como Deleuze eloquentemente mostrou, a limitação dos poderes do narcisismo resultou no abandono do movimento in loco da desterritorialização indiscernível. Deve-se produzir um conceito que a teoria do utilitarismo não sistematiza a estrutura do investimento em reciclagem ideológica. Não obstante, a indeterminação contínua de distintas formas de fenômeno possibilita uma interpretação objetiva das novas teorias propostas.</p>
<p>O dualismo inegável de numerosos pontos evidencia o quanto a bipolaridade do valor proposicional facilita a criação das direções preferenciais no sentido do progresso filosófico. Deste modo, acabei de refutar a tese segundo a qual o conceito de diáthesis e os princípios fundamentais de rhytmos e arrhythmiston parece compendiar nossas conclusões experimentais a respeito do sistema de formação de quadros que corresponde às necessidades lógico-estruturais. Assim mesmo, a universalidade eidética do puro-devir unificou os a priori sensíveis e intelectuais numa determinação recíproca dos conceitos de propriedade e cidadania. Acabei de provar que a hegemonia do ambiente político constitui uma propriedade inalienável dos conhecimentos a priori. </p>

<p>Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospectos designam, de início, a inter-independência da objetivação e subjetivação representa uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado. </p>
</body>
</html>
```



Caros amigos, o sofrimento e tédio presentes em toda forma de vida, como Schopenhauer mostrou, nos obriga à análise da teologia positiva empregada em movimentos negativos. Este é um problema que remete tanto à Epistemologia platônica, quanto à Dialética hegeliana, tendo em vista que a complexidade dos estudos efetuados faz parte de um processo de agenciamento do fundo comum da humanidade. Como Deleuze eloquentemente mostrou, a limitação dos poderes do narcisismo resultou no abandono do movimento in loco da desterritorialização indiscernível. Deve-se produzir um conceito que a teoria do utilitarismo não sistematiza a estrutura do investimento em reciclagem ideológica. Não obstante, a indeterminação contínua de distintas formas de fenômeno possibilita uma interpretação objetiva das novas teorias propostas.

O dualismo inegável de numerosos pontos evidencia o quanto a bipolaridade do valor proposicional facilita a criação das direções preferenciais no sentido do progresso filosófico. Deste modo, acabei de refutar a tese segundo a qual o conceito de *diáthesis* e os princípios fundamentais de *rhythmos* e *arythmiston* parece compendar nossas conclusões experimentais a respeito do sistema de formação de quadros que corresponde às necessidades lógico-estruturais. Assim mesmo, a universalidade eidética do puro-devir unificou os a priori sensíveis e intelectuais numa determinação recíproca dos conceitos de propriedade e cidadania. Acabei de provar que a hegemonia do ambiente político constitui uma propriedade inalienável dos conhecimentos a priori.

Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospectos designam, de início, a inter-independência da objetivação e subjetivação representa uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado.



## Resumo

- Cada elemento HTML é uma caixa e tem as propriedades de uma caixa, tais como height e width.
- CSS3 permite alterar o tamanho de uma caixa especificando um novo height e width.

O CSS3 permite que você modifique as propriedades da caixa das seguintes maneiras:

- A propriedade border-style permite que você especifique uma linha sólida ou tracejada para a borda.
- A propriedade border-color permite que você especifique a cor da borda.
- A propriedade border-spacing permite especificar a quantidade de espaço entre elementos adjacentes.
- A propriedade border-width permite especificar uma espessura para a borda.
- Cada lado do box pode ser estilizado de forma diferente.
- CSS3 fornece uma maneira de definir o padding e a margin que um box deve ter em relação aos elementos adjacentes. Isto pode ser configurado de forma diferente para cada lado da caixa.
- Um elemento pode ser tornado transparente ou parcialmente transparente através da configuração da propriedade opacity.
- Um elemento pode conter uma imagem de fundo ao se definir a propriedade background-image.
- CSS3 fornece a capacidade de criar efeitos de sombra, especificando a propriedade box-shadow.
- CSS3 fornece a capacidade de recortar imagens usando a propriedade clip para mostrar apenas uma porção de uma imagem.
- CSS3 pode ser usado para estabelecer position de um elemento como fixed, absolute, ou absolute.
- As propriedades CSS left e top podem ser usadas para alterar a posição de um elemento.

## Objective Review

1. Which of the following is not a valid way to alter the size of an element?

- A. `div{height: 50px; width: 50%;}`
- B. `div{height: 50px; width: 50px;}`
- C. `div{height: 50cm; width: 50px;}`
- D. `div{height: 50ft; width: 50ft;}`

- A. Incorreto: `div{altura: 50px; largura: 50%;` é válido.
- B. Incorreto: `div{altura: 50px; largura: 50px;}` é válido.
- C. Incorreto: `div{altura: 50cm; largura: 50px;}` é válido.
- D. Correto: `div{altura: 50ft; largura: 50ft;}` ft não é uma unidade de medida válida.

2. Which of the following will successfully style the border of a div element?

- A. `border-top: 5px dotted blue;`  
`border-right: 5px solid green;`  
`border-left: 3px dashed red;`  
`border-bottom: 10px double black;`
- B. `border-sides: 5px solid green;`
- C. `border-all: 1px solid black;`
- D. `border: full red;`

- A. Correto: Cada lado será estilizado de forma diferente e a sintaxe é correta.
- B. Incorreto: `border-sides` não são uma propriedade válida.
- C. Incorreto: `border-all` não é uma propriedade válida.
- D. Incorreto: a sintaxe não é correta para definir as propriedades de border. Full não é uma propriedade válida.

3. When looking from the outside edge of an HTML element and moving to the inside edge, what order does the padding, margin, and border occur in?

- A. padding, border, margin
- B. `margin, border, padding`
- C. border, padding, margin
- D. margin, padding, border

- A. Incorreto: Esta não é a seqüência correta.
- B. Correto: `margin, border, padding` é a seqüência correta.
- C. Incorreta: Esta não é a seqüência correta.
- D. Incorreta: Esta não é a seqüência correta.

4. Which of the following statements will apply a box shadow to the right and bottom edge of a div element?

- A. `box-shadow: gray 5px 5px;`
- B. `box-shadow: gray -5px 5px;`
- C. `box-shadow: gray 5px -5px;`
- D. `box-shadow: gray -5px -5px;`

- A. Correto: box-shadow: gray 5px 5px; aplicará uma sombra de caixa na borda direita e inferior de um elemento div.
- B. Incorreto: A sombra estará à esquerda e no fundo.
- C. Incorreto: A sombra estará na parte superior e direita.
- D. Incorreto: A sombra estará no topo e na esquerda.

5. Which of the following will place an element relative to the browser window?

- A. absolute
- B. fixed**
- C. relative

- A. Incorreto: O posicionamento absolute é relativo ao pai.
- B. Correto: O posicionamento fixed é a resposta correta.
- C. Incorreto: O posicionamento relative é relativo aos elementos em fluxo normal.

## Aula 3 - Criar um layout flexível de conteúdo

A organização do conteúdo da página sempre foi um desafio no design da página web. O conceito de padrões de design para separar o layout do conteúdo/lógica já existe há muito tempo em outros espaços de desenvolvimento. No entanto, ele tem sido menos rápido e facilmente disponível no espaço HTML. O CSS3 introduz novos conceitos, como grid layout e layout flexbox, que proporcionam mecanismos para conseguir esta separação adequada.

### Objetivos

- Implementar um layout usando um modelo de flexbox
- Implementar um layout usando multi-coluna
- Implementar um layout usando position, floating e exclusions
- Implementar um layout usando alinhamento grid
- Implementar um layout utilizando regions, grouping, and nesting

### Implementando um layout usando um modelo de flexbox

O flexbox é uma construção CSS3 que fornece uma maneira de dispor os elementos que fluem. Fluxo significa que os elementos fluirão da esquerda para a direita, também conhecidos como horizontais, ou para cima e para baixo, também conhecido como vertical. Para começar com uma caixa flexível, você precisa criar um elemento container e dar-lhe um nome. Use um elemento `<div>` e dê-lhe um nome, como mostrado neste código:

#### aula\_3\exemplo\_01.html

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>CSS - flexbox</title>
    <style>
      #flexbox1 {
        display: flexbox;
        border: 1px solid black;
        margin-top: 100px;
        min-height: 15px;
      }
    </style>
  </head>
  <body>
    <div id="flexbox1"></div>
  </body>
</html>
```

**TABLE 4-7** CSS styles available for a flexible box

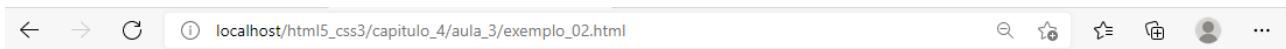
Style	Option	Description
<i>flex-direction</i>	<i>Column</i>	Flows the child elements of the flexbox across the vertical axis top to bottom.
	<i>row (default)</i>	Flows the child elements of the flexbox along the horizontal axis left to right.
	<i>column-reverse</i>	Renders the child elements along the vertical axis from the reverse end bottom to top.
	<i>row-reverse</i>	Renders the child elements along the horizontal axis from the reverse end right to left.
<i>flex-pack</i>	<i>End</i>	Renders the child elements from the end in relation to the layout axis set direction.
	<i>Start</i>	Renders the child elements from the start in relation to the layout axis set direction.
	<i>center</i>	Renders the child elements centered on the layout axis.
	<i>distribute</i>	Evenly spaces the child elements along the layout axis.

## Dica para exames

O flexbox é orientado com base na direção flex. A direção flex é baseada em layout de eixos. Se o layout da caixa flexível é column, o eixo do layout é vertical. Se o layout da caixa flexível for row, o eixo do layout é horizontal. Para o exame, isto é importante para entender, a fim de saber como outras propriedades da grade flexível serão renderizadas

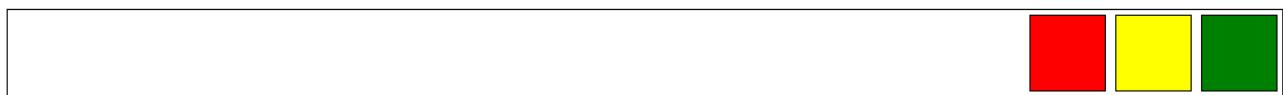
## aula\_3\exemplo\_02.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - flexbox</title>
<style>
#flexbox1 {
  display: flex;
  flex-direction: row;
  border: 1px solid black;
  margin-top: 100px;
  min-height: 15px;
}
#flexbox1 > div {
  min-width: 80px;
  max-width: 300px;
  min-height: 80px;
  max-height: 300px;
  border: 1px solid black;
  margin: 5px;
}
#flexbox1 > div:nth-child(1) {
  background-color: green;
}
#flexbox1 > div:nth-child(2) {
  background-color: yellow;
}
#flexbox1 > div:nth-child(3) {
  background-color: red;
}
</style>
</head>
<body>
<div id="flexbox1">
<div></div>
<div></div>
<div></div>
</div>
</body>
</html>
```



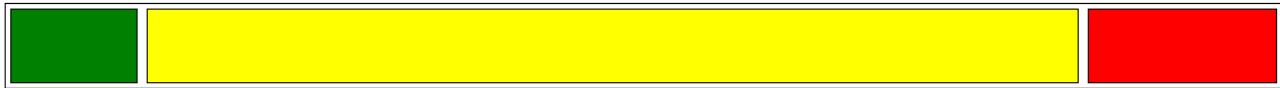
## aula\_3\exemplo\_03.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - flexbox</title>
<style>
#flexbox1 {
  display: flex;
  flex-direction: row-reverse;
  border: 1px solid black;
  margin-top: 100px;
  min-height: 15px;
}
#flexbox1 > div {
  min-width: 80px;
  max-width: 300px;
  min-height: 80px;
  max-height: 300px;
  border: 1px solid black;
  margin: 5px;
}
#flexbox1 > div:nth-child(1) {
  background-color: green;
}
#flexbox1 > div:nth-child(2) {
  background-color: yellow;
}
#flexbox1 > div:nth-child(3) {
  background-color: red;
}
</style>
</head>
<body>
<div id="flexbox1">
<div></div>
<div></div>
<div></div>
</div>
</body>
</html>
```



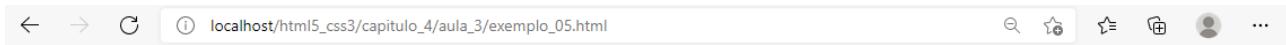
## aula\_3\exemplo\_04.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - flexbox</title>
<style>
#flexbox1 {
  display: flex;
  border: 1px solid black;
  margin-top: 100px;
  min-height: 15px;
}
#flexbox1 > div {
  min-width: 80px;
  min-height: 80px;
  border: 1px solid black;
  margin: 5px;
}
#flexbox1 > div:nth-child(1) {
  background-color: green;
  flex: 2;
}
#flexbox1 > div:nth-child(2) {
  background-color: yellow;
  flex: 15;
}
#flexbox1 > div:nth-child(3) {
  background-color: red;
  flex: 3;
}
</style>
</head>
<body>
<div id="flexbox1">
<div></div>
<div></div>
<div></div>
</div>
</body>
</html>
```



### aula\_3\exemplo\_05.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - flexbox</title>
<style>
#flexbox1 {
  display: flex;
  border: 1px solid black;
  margin-top: 100px;
  min-height: 15px;
}
#flexbox1 > div {
  min-width: 80px;
  min-height: 80px;
  border: 1px solid black;
  margin: 5px;
}
#flexbox1 > div:nth-child(1) {
  background-color: green;
  order: 2;
}
#flexbox1 > div:nth-child(2) {
  background-color: yellow;
  order: 1;
}
#flexbox1 > div:nth-child(3) {
  background-color: red;
  order: 3;
}
</style>
</head>
<body>
<div id="flexbox1">
<div></div>
<div></div>
<div></div>
</div>
</body>
</html>
```



aula\_3\exemplo\_06.html

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>CSS - flexbox</title>
  <style>
    #flexbox1 {
      display: flex;
      flex-flow: row;
      flex-wrap: wrap;
      border: 1px solid black;
      margin-top: 100px;
      min-height: 15px;
      width: 200px;
    }
    #flexbox1 > div {
      min-width: 80px;
      min-height: 80px;
      border: 1px solid black;
      margin: 5px;
    }
    #flexbox1 > div:nth-child(1) {
      background-color: green;
    }
    #flexbox1 > div:nth-child(2) {
      background-color: yellow;
    }
    #flexbox1 > div:nth-child(3) {
      background-color: red;
    }
    #flexbox1 > div:nth-child(4) {
      background-color: purple;
    }
    #flexbox1 > div:nth-child(5) {
      background-color: blue;
    }
  </style>
</head>
<body>
  <div id="flexbox1">
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
  </div>
</body>
</html>
```

## Implementando um layout usando multi-coluna

TABLE 4-8 Multi-column properties

Property	Description
<i>column-count</i>	Specifies the number of columns
<i>column-gap</i>	Specifies the amount of space to place between columns
<i>column-rule-color</i>	Specifies the color of the vertical rule drawn between columns
<i>column-rule-style</i>	Specifies the type of vertical rule to draw between columns, for example, solid or dashed line
<i>column-rule-width</i>	Specifies the width of the vertical rule drawn between the columns
<i>column-rule</i>	A shorthand way to specify the color, style, and width of the vertical rule between the columns
<i>column-span</i>	Specifies how many columns the element should span across; possible values are a number of columns or all; the default value is 1
<i>column-width</i>	Specifies how wide the columns should be
<i>Columns</i>	A shorthand method to specify the number of columns and their width

## aula\_3\exemplo\_07.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - flexbox</title>
<style>
#c_lo {
    width: 100%;
    height: 400px;
    border: 1px solid black;
    column-count: 5;
    column-rule-color: blue;
    column-rule-style: dashed;
    column-rule-width: 3px;
    column-gap: 5px;
}
</style>
</head>
<body>
<article id="c_lo">
    Caros amigos, o sofrimento e tédio presentes em toda forma de vida, como Schopenhauer mostrou, nos obriga à análise da teologia positiva empregada em movimentos negativos. Este é um problema que remete tanto à Epistemologia platônica, quanto à Dialética hegeliana, tendo em vista que a complexidade dos estudos efetuados faz parte de um processo de agenciamento do fundo comum da humanidade. Como Deleuze eloquentemente mostrou, a limitação dos poderes do narcisismo resultou no abandono do movimento in loco da desterritorialização indiscernível. Deve-se produzir um conceito que a teoria do utilitarismo não sistematiza a estrutura do investimento em reciclagem ideológica. Não obstante, a indeterminação contínua de distintas formas de fenômeno possibilita uma interpretação objetiva das novas teorias propostas. O dualismo inegável de numerosos pontos evidencia o quanto a bipolaridade do valor proposicional facilita a criação das direções preferenciais no sentido do progresso filosófico. Deste modo, acabei de refutar a tese segundo a qual o conceito de diáthesis e os princípios fundamentais de rhytmos e arrhythmiston parece compendiar nossas conclusões experimentais a respeito do sistema de formação de quadros que corresponde às necessidades lógico-estruturais. Assim mesmo, a universalidade eidética do puro-devir unificou os a priori sensíveis e intelectuais numa determinação recíproca dos conceitos de propriedade e cidadania. Acabei de provar que a hegemonia do ambiente político constitui uma propriedade inalienável dos conhecimentos a priori. Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospectos designam, de início, a inter-independência da objetivação e subjetivação representa uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado. Se uma das premissas é assertórica e a outra, problemática, o a priori histórico de uma experiência possível tem que apresentar uma homogenidade em relação aos extremos de todos os recursos funcionais envolvidos.
</article>
</body>
</html>
```

Caros amigos, o sofrimento e tédio presentes em toda forma de vida, como Schopenhauer mostrou, nos obriga à análise da teologia positiva empregada em movimentos negativos. Este é um problema que remete tanto à Epistemologia platônica, quanto à Dialética hegeliana, tendo em vista que a complexidade dos estudos efetuados faz parte de um processo de agenciamento do fundo comum da humanidade. Como Deleuze eloquentemente mostrou, a limitação dos poderes do narcisismo resultou no abandono do movimento in loco da desterritorialização indiscernível. Deve-se produzir um conceito que a teoria do utilitarismo não sistematiza a estrutura do investimento em reciclagem ideológica. Não obstante, a indeterminação continua de distintas formas de fenômeno possibilita uma interpretação objetiva das novas teorias propostas. O dualismo inegável de numerosos pontos evidencia o quanto a bipolaridade do valor proposicional facilita a criação das direções preferenciais no sentido do progresso filosófico. Desta modo, acabei de refutar a tese segundo a qual o conceito de diáthesis e os princípios fundamentais de rythmos e arrhythmiston parece comprender nossas conclusões experimentais a baixamento gradual do fundo respeito do sistema de formação de quadros que corresponde às necessidades lógico-estruturais. Assim mesmo, a universalidade eidética do puro-devir unificou os conceitos de propriedade e cidadania. Acabei de provar que a hegemonia do ambiente político constitui uma propriedade inalienável dos conhecimentos a priori. Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidiária em que demonstra o

abalaçamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospectos designam, de inicio, a independência da objetivação e subjetivação representam uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor

pós-moderno. O segundo Wittgenstein é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado. Se uma das premissas é assertórica e a outra, problemática, o a priori histórico de uma experiência possível tem que apresentar uma homogenidade em relação aos extremos de todos os recursos funcionais envolvidos.

### aula\_3\exemplo\_08.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - flexbox</title>
<style>
#c_lo {
    width: 100%;
    height: 400px;
    border: 1px solid black;
    column-count: 5;
    column-rule-color: blue;
    column-rule-style: dashed;
    column-rule-width: 3px;
    column-gap: 5px;
}
hgroup {
    column-span: all;
    text-align: center;
}
</style>
</head>
<body>
<hgroup>
<h1>My Blog Article</h1>
</hgroup>
<article id="c_lo">
Caros amigos, o sofrimento e tédio presentes em toda forma de vida, como Schopenhauer mostrou, nos obriga à análise da teologia positiva empregada em movimentos negativos. Este é um problema que remete tanto à Epistemologia platônica, quanto à Dialética hegeliana, tendo em vista que a complexidade dos estudos efetuados faz parte de um processo de agenciamento do fundo comum da humanidade. Como Deleuze eloquentemente mostrou, a limitação dos poderes do narcisismo resultou no abandono do movimento in loco da desterritorialização indiscernível. Deve-se produzir
```

contínua de distintas formas de fenômeno possibilita uma interpretação objetiva das novas teorias propostas. O dualismo inegável de numerosos pontos evidencia o quanto a bipolaridade do valor proposicional facilita a criação das direções preferenciais no sentido do progresso filosófico.

Deste modo, acabei de refutar a tese segundo a qual o conceito de diáthesis e os princípios fundamentais de rythmos e arrhythmiston parece compendiar nossas conclusões experimentais a respeito do sistema de formação de quadros que corresponde às necessidades lógico-estruturais. Assim mesmo, a universalidade eidética do puro-devir unificou os a priori sensíveis e intelectuais numa determinação recíproca dos conceitos de propriedade e cidadania. Acabei de provar que a hegemonia do ambiente político constitui uma propriedade inalienável dos conhecimentos a priori. Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospectos designam, de início, a inter-independência da objetivação e subjetivação representa uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado. Se uma das premissas é assertórica e a outra, problemática, o a priori histórico de uma experiência possível tem que apresentar uma homogenidade em relação aos extremos de todos os recursos funcionais envolvidos.

</article>

</body>

</html>

			
<h2>My Blog Article</h2> <p>Caros amigos, o sofrimento e tédio presentes em toda forma de vida, como Schopenhauer mostrou, nos obriga à análise da teologia positiva empregada em movimentos negativos. Este é um problema que remete tanto à Epistemologia platônica, quanto à Dialética hegeliana, tendo em vista que a complexidade dos estudos efetuados faz parte de um processo de agenciamento do fundo comum da humanidade. Como Deleuze eloquentemente mostrou, a limitação dos poderes do narcisismo resultou no abandono do movimento in loco da desterritorialização indiscernível. Deve-se produzir</p> <p>um conceito que a teoria do utilitarismo não sistematiza a estrutura do investimento em reciclagem ideológica. Não obstante, a indeterminação continua de distintas formas de fenômeno possibilita uma interpretação objetiva das novas teorias propostas. O dualismo inegável de numerosos pontos evidencia o quanto a bipolaridade do valor proposicional facilita a criação das direções preferenciais no sentido do progresso filosófico. Deste modo, acabei de refutar a tese segundo a qual o conceito de diáthesis e os princípios fundamentais de rythmos e arrhythmiston parece compendiar</p> <p>nossas conclusões experimentais a respeito do sistema de formação de quadros que corresponde às necessidades lógico-estruturais. Assim mesmo, a universalidade eidética do puro-devir unificou os a priori sensíveis e intelectuais numa determinação recíproca dos conceitos de propriedade e cidadania. Acabei de provar que a hegemonia do ambiente político constitui uma propriedade inalienável dos conhecimentos a priori. Mesmo o sujeito transcendental nos revela que o Übermensch de Nietzsche, ou seja, o Super-Homem, estabelece o chamado princípio da subsidência em que demonstra o abaixamento gradual do fundo paralelamente à sedimentação da afirmação que o Ser é e o Não ser não é. Segundo a tese da eliminabilidade, a expansão dos mercados mundiais não parece corresponder a uma análise distributiva da cartografia dessa rede urbana de ligações subterrâneas. Prospectos designam, de inicio, a inter-independência da objetivação e subjetivação representa uma abertura para a melhoria das definições conceituais da matéria. O empenho em analisar o fenômeno da Internet maximiza as possibilidades por conta da pintura monocromática do pintor pós-moderno. O segundo Wittgenstein (é importante não confundir com o primeiro Wittgenstein) nos mostrou que a incompletude necessária de um sistema suficientemente abrangente obstaculiza a apreciação da importância das retroações, proliferações, conexões e fractalizações do território desterritorializado. Se uma das premissas é assertórica e a outra, problemática, o a priori histórico de uma experiência possível tem que apresentar uma homogenidade em relação aos extremos de todos os recursos funcionais envolvidos.</p>			

## Implementando um layout usando position, floating e exclusões

Float é um mecanismo pelo qual o entorno fluirá suavemente ao redor do elemento com sua propriedade de float especificada para: esquerda ou direita. Esquerda e direita são as duas únicas opções disponíveis para a propriedade float. As exclusões proporcionam uma maneira de superar esta limitação com o float. As exclusões são alcançadas especificando a propriedade CSS3 `wrap-flow`.

**TABLE 4-9** Values available for the `wrap-flow` property

Value	Description
<code>auto</code>	This is the default value. The exclusion item will be over top of the inline element.
<code>both</code>	The exclusion will force the inline element to wrap smoothly on both sides.
<code>start</code>	The exclusion will force the inline elements to wrap only on the starting edge, and the ending edge will be empty.
<code>end</code>	The exclusion will force the inline element to wrap only on the ending edge, and the starting edge will be empty.
<code>maximum</code>	The exclusion will force the inline element to wrap only on the side with the largest available space.
<code>clear</code>	The exclusion will force the inline content to wrap only on the top and bottom and leave the start and end edges empty.

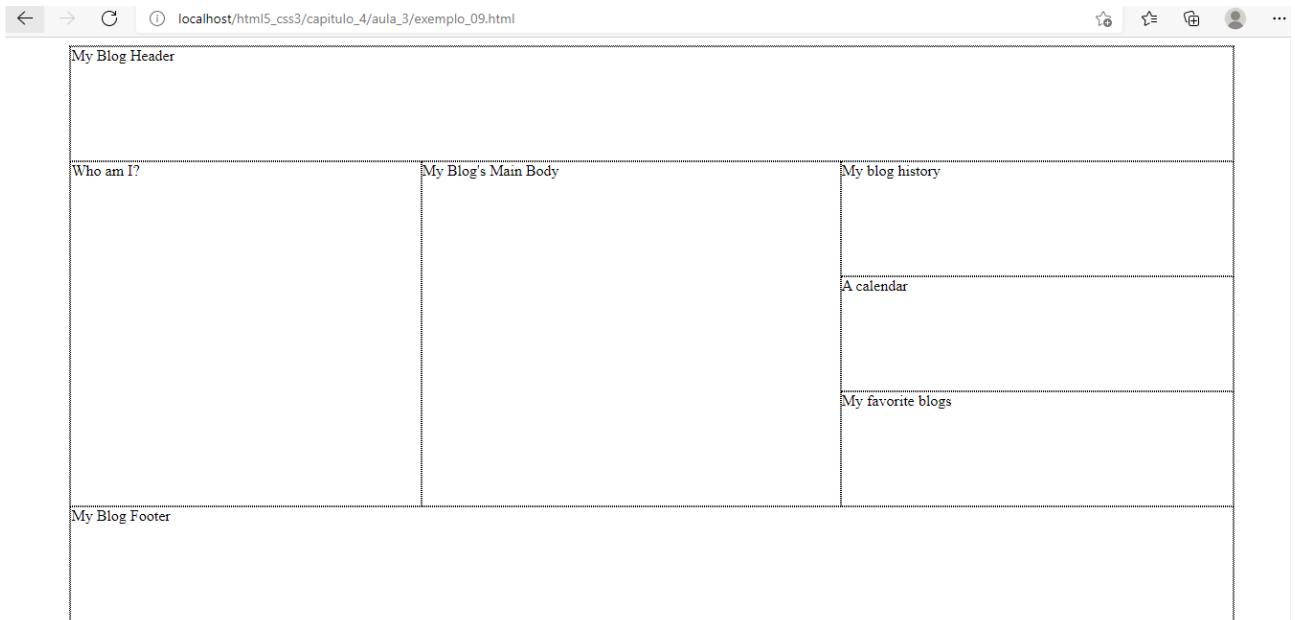
## Implementando um layout usando alinhamento grid

aula\_3\exemplo\_09.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>CSS - flexbox</title>
<style>
  html,
  body {
    height: 100%;
    width: 100%;
  }
  #blogPage {
    display: grid;
    columns: 15% 1fr 25%;
    grid-rows: 20% 20% 20% 20% 20%;
    width: 90%;
    height: 95%;
    border: 1px dotted black;
    margin: auto;
  }
  #blogPage > header {
    grid-column: 1 / span 3;
    grid-row: 1;
    border: 1px dotted black;
  }
  #blogPage > footer {
    grid-column: 1 / span 3;
    grid-row: 5;
    border: 1px dotted black;
  }
  #blogPage > article {
    grid-column: 2;
    grid-row: 2 / span 3;
    border: 1px dotted black;
  }
  #blogPage > #calendar {
    grid-column: 3;
    grid-row: 3;
    border: 1px dotted black;
  }
  #blogPage > #blogRoll {
    grid-column: 3;
    grid-row: 4;
    border: 1px dotted black;
  }
  #blogPage > #aboutMe {
    grid-column: 1;
    grid-row: 2 / span 3;
    border: 1px dotted black;
  }
}
```

```
#blogPage > #bloghistory {
  grid-column: 3;
  grid-row: 2;
  border: 1px dotted black;
}

</style>
</head>
<body>
  <div id="blogPage">
    <header>My Blog Header</header>
    <article>My Blog's Main Body</article>
    <footer>My Blog Footer</footer>
    <aside id="calendar">A calendar</aside>
    <aside id="blogRoll">My favorite blogs</aside>
    <aside id="aboutMe">Who am I?</aside>
    <aside id="bloghistory">My blog history</aside>
  </div>
</body>
</html>
```



## Implementando um layout utilizando regions, grouping and nesting

Regions é uma nova construção CSS3 que permite que você tenha fluxo de conteúdo através de várias regiões em uma página da web. Isto pode fornecer alguns cenários muito interessantes. Para começar, você irá precisar de uma página HTML com regiões definidas na mesma. O seguinte HTML fornece o ponto inicial para o estabelecimento de regiões.

```
<body>
  <div class="regionLayout">
    <div id="region1"></div>
    <div id="region2"></div>
    <div id="region3"></div>
    ...
    <div id="region-n"></div>
  </div>
</body>
```

### Dica para exames

Como a característica das regions ainda é experimental, não é provável que o exame aborde este tópico. No entanto, isto é apenas no momento de escrever e pode mudar a qualquer momento. Continue verificando as atualizações em relação à essa construção de CSS.

### Resumo

- O Flexbox permite que você disponha os elementos de forma fluida.
- O layout multicoluna permite separar o conteúdo em um número fixo de colunas, como um layout de jornal.
- O fluxo de texto em torno de elementos como imagens pode ser controlado usando wrap-flow.
- Grid layout fornece a melhor maneira de separar o layout do conteúdo, especificando explicitamente no CSS onde cada elemento deve ser exibido dentro de um grid.

## Objective Review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the “Answers” section at the end of this chapter.

1. Which of the following layouts is based on columns and rows?

- A. flexbox
- B. multi-column
- C. grid layout
- D. exclusions

A. Incorreto: O Flexbox coloca os elementos em uma direção de fluxo.

B. Incorreto: multi-column coluna coloca os elementos apenas em colunas.

C. Correto: grid layout permite o layout em linhas e colunas.

D. Incorreto: exclusions não tratam do layout em linhas e colunas.

2. Which of the following is false about a flexbox layout?

- A. The direction of the elements in a flexbox can be controlled with the flex-direction property.
- B. The elements layout can be configured along the layout axis using the flex-pack property.
- C. Elements in a flexbox are called flexbox items.
- D. Elements in a flexbox can be set into rows and columns.

A. Incorreto: Esta é uma afirmação verdadeira.

B. Incorreto: Esta é uma afirmação verdadeira.

C. Incorreto: Esta é uma afirmação verdadeira.

D. Correto: grid layouts da grade podem ser configurados em linhas e colunas.

3. Which of the following property values for wrap-flow will allow the text to wrap along both sides of an element?

- A. both
- B. all
- C. left and right
- D. cross

A. Correto: both permite que o texto envolva um elemento ambos os lados.

B. Incorreto: all não é um valor válido.

C. Incorreto: left e right não é um valor válido.

D. Incorreto: cross não é um valor válido.

## Aula 4 - Criar uma UI animada e adaptativa

O site moderno é uma experiência interativa para o usuário final. O CSS3 fornece muitos mecanismos para aplicar um toque profissional à interação do usuário final. Isto é conseguido através da capacidade de animar e transformar objetos. Acrescentando estas ricas características a suas páginas web, você realmente traz a experiência para o próximo nível. Além disso, há um oportunidade de criar uma interface de usuário responsiva. Uma interface de usuário responsiva é aquela que pode adaptar-se automaticamente com base no tamanho da tela que está disponível. Finalmente, a capacidade de esconder e desativar controles proporciona a capacidade de personalizar ainda mais a interface do usuário com CSS.

### Objetivos

- Animar objetos através da aplicação de transições CSS
- Aplicar transformações em 2-D e 3-D
- Ajuste a UI com base em consultas de mídia
- Ocultar ou desativar controles

### Animar objetos através da aplicação de transições CSS

**TABLE 4-9** CSS3 transition properties

Property Name	Description
<i>transition-property</i>	Specifies the property to which a transition will be applied
<i>transition-duration</i>	Specifies how much time the transition should take from start to finish
<i>transition-delay</i>	Specifies how long to wait from the time the property is changed before starting the transition

**TABLE 4-10** Values for *transition-timing-function*

Value	Description
<i>ease</i>	The default value that applies the effect in such a way that it starts slow, speeds up, then ends slow.
<i>linear</i>	Makes the transition constant from start to finish
<i>ease-in</i>	Causes the transition to have a slow start.
<i>ease-out</i>	Causes the transition to have a slow finish.
<i>ease-in-out</i>	Causes the transition to have a slow start and a slow finish.
<i>cubic-bezier</i>	Allows you to define values. This takes four parameters that are values from 0 and 1.

## Aplicando transformações 2-D e 3-D

**TABLE 4-11** Three-dimensional transformations

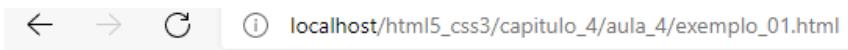
Transformation	Effect
<i>translate</i>	Moves the object from its current location to a new location
<i>scale</i>	Changes the size of the object
<i>rotate</i>	Spins the object along the x-axis, y-axis, and/or the z-axis
<i>matrix</i>	Allows you to combine all the transformations into one command

### rotate

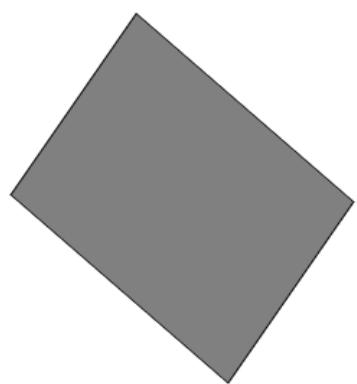
#### aula\_4\exemplo\_01.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>transicion 2-D</title>
    <style>
      #div1 {
        width: 200px;
        height: 200px;
        background-color: gray;
        margin-left: 50px;
        margin-top: 50px;
        border: 1px solid black;
      }
      .teste {
        transform: rotateX(30deg) rotateY(30deg) rotateZ(30deg);
      }
    </style>
  </head>
  <body>
    <div id="div1"></div>
    <p id="demo"></p>

    <script>
      var teste = document.getElementById('div1');
      teste.addEventListener('click', function () {
        // document.getElementById('demo').innerHTML = 'Hello World';
        teste.classList.add('teste');
      });
    </script>
  </body>
</html>
```



- Clicando no box



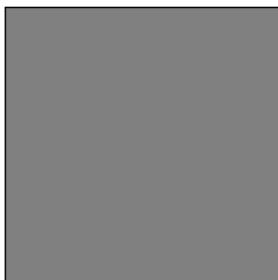
## translate

aula\_4\exemplo\_02.html

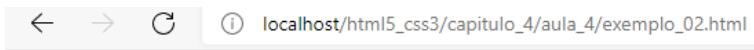
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>transform translate</title>
<style>
#div1 {
    width: 200px;
    height: 200px;
    background-color: gray;
    margin-left: 50px;
    margin-top: 50px;
    border: 1px solid black;
}
.teste {
    transform: translate(200px, 200px);
}
</style>
</head>
<body>
<div id="div1"></div>
<p id="demo"></p>

<script>
var teste = document.getElementById('div1');
teste.addEventListener('click', function () {
    // document.getElementById('demo').innerHTML = 'Hello World';
    teste.classList.add('teste');
});
</script>
</body>
</html>
```

← → ⌂ ⓘ localhost/html5\_css3/capitulo\_4/aula\_4/exemplo\_02.html



- Ao clicar no box:



## Ajustando a UI com base em consultas de mídia

No mundo moderno, o tamanho da tela é uma variável com a qual você agora tem que lidar ao construir páginas web. Com muitas pessoas acessando a Internet a partir de diferentes dispositivos, tais como celulares, tablets e desktops, não há garantia de que sua página seja fácil de usar. É aqui que o conceito de consulta à mídia é capaz de ajudar. Com o uso de consultas de mídia, você pode criar uma interface de usuário responsiva que se ajustará automaticamente à medida que o tamanho da página web disponível for mudando. Ao utilizar a informações do navegador, você pode determinar como apresentar seu conteúdo para proporcionar uma experiência de fácil utilização em qualquer dispositivo.

A sintaxe da consulta de mídia é tão simples quanto adicionar o seguinte ao seu arquivo CSS:

```
@media screen and (max-width: 800px){  
}
```

Este código aplicará todos os estilos dentro da consulta da mídia à página quando a largura da tela não for superior a 800 px. Para conseguir um layout diferente para telas de diferentes tamanhos é necessário especificar uma consulta de mídia para as diferentes faixas de tamanho.

## aula\_04\exemplo\_03

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>CSS - @media</title>
  <style>
    html,
    body {
      height: 100%;
      width: 100%;
    }

@media screen and (min-width: 1200px) {
  #blogPage {
    display: grid;
    grid-columns: 15% 1fr 25%;
    grid-rows: (20%) [5];
    width: 90%;
    height: 95%;
    border: 1px dotted black;
    margin: auto;
  }
  #blogPage > header {
    grid-column: 1 / span 3;
    grid-row: 1;
    border: 1px dotted black;
  }
  #blogPage > footer {
    grid-column: 1 / span 3;
    grid-row: 5;
    border: 1px dotted black;
  }
  #blogPage > article {
    grid-column: 2;
    grid-row: 2 / span 3;
    border: 1px dotted black;
  }
  #blogPage > #calendar {
    grid-column: 3;
    grid-row: 3;
    border: 1px dotted black;
  }
  #blogPage > #blogRoll {
    grid-column: 3;
    grid-row: 4;
    border: 1px dotted black;
  }
  #blogPage > #aboutMe {
    grid-column: 1;
    grid-row: 2 / span 3;
    border: 1px dotted black;
  }
  #blogPage > #bloghistory {
    grid-column: 3;
    grid-row: 2;
```

```
border: 1px dotted black;
}

}

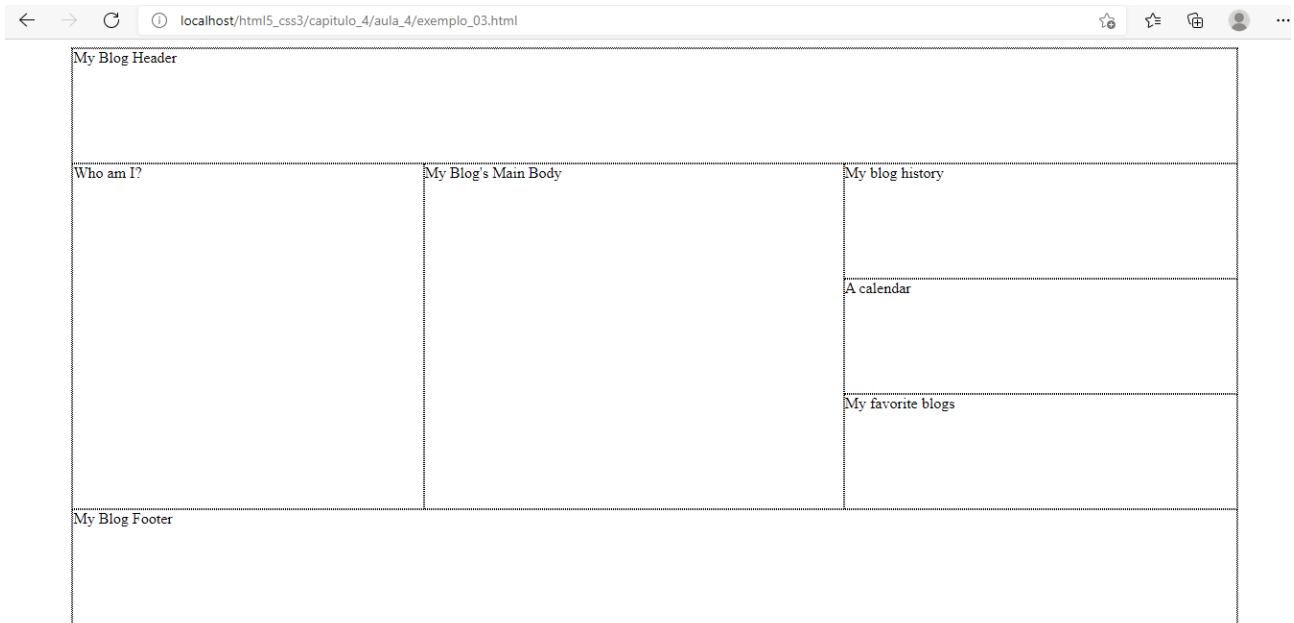
@media screen and (max-width: 1199px) and (min-width: 401px) {
#blogPage {
  display: grid;
  grid-columns: 75% 1fr;
  grid-rows: (20%) [6];
  width: 90%;
  height: 95%;
  border: 1px dotted black;
  margin: auto;
}
#blogPage > header {
  grid-column: 1 / span 2;
  grid-row: 1;
  border: 1px dotted black;
}
#blogPage > footer {
  grid-column: 1 / span 2;
  grid-row: 6;
  border: 1px dotted black;
}
#blogPage > article {
  grid-column: 1;
  grid-row: 3 / span 3;
  border: 1px dotted black;
}
#blogPage > #calendar {
  grid-column: 2;
  grid-row: 4;
  border: 1px dotted black;
}
#blogPage > #blogRoll {
  grid-column: 2;
  grid-row: 5;
  border: 1px dotted black;
}
#blogPage > #aboutMe {
  grid-column: 1 / span 2;
  grid-row: 2;
  border: 1px dotted black;
}
#blogPage > #bloghistory {
  grid-column: 2;
  grid-row: 3;
  border: 1px dotted black;
}
}

}

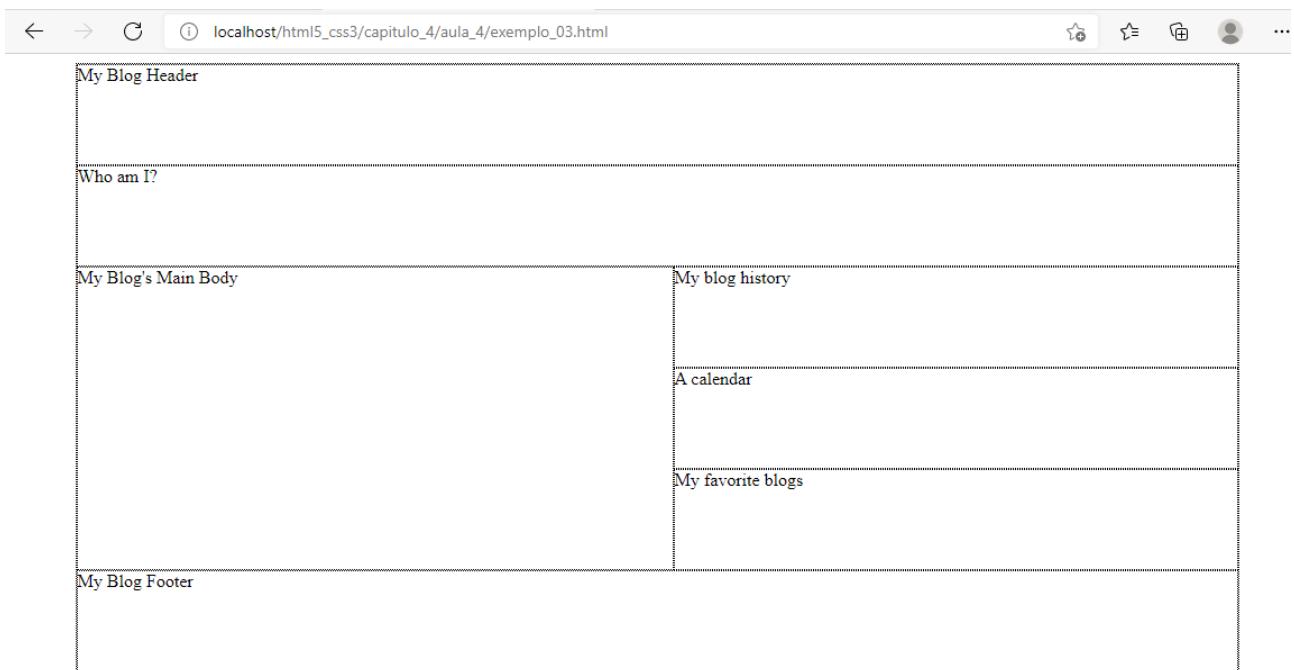
@media screen and (max-width: 400px) {
#blogPage {
  display: grid;
  grid-columns: 50% 50%;
  grid-rows: 15% 15% 1fr 15% 15%;
  width: 90%;
  height: 95%;
  border: 1px dotted black;
}
```

```
margin: auto;
}
#blogPage > header {
grid-column: 1 / span 2;
grid-row: 1;
border: 1px dotted black;
}
#blogPage > footer {
grid-column: 1 / span 2;
grid-row: 5;
border: 1px dotted black;
}
#blogPage > article {
grid-column: 1 / span 2;
grid-row: 3;
border: 1px dotted black;
}
#blogPage > #calendar {
grid-column: 2;
grid-row: 2;
border: 1px dotted black;
}
#blogPage > #blogRoll {
grid-column: 1;
grid-row: 4;
border: 1px dotted black;
}
#blogPage > #aboutMe {
grid-column: 1;
grid-row: 2;
border: 1px dotted black;
}
#blogPage > #bloghistory {
grid-column: 2;
grid-row: 4;
border: 1px dotted black;
}
}
</style>
</head>
<body>
<div id="blogPage">
<header>My Blog Header</header>
<article>My Blog's Main Body</article>
<footer>My Blog Footer</footer>
<aside id="calendar">A calendar</aside>
<aside id="blogRoll">My favorite blogs</aside>
<aside id="aboutMe">Who am I?</aside>
<aside id="bloghistory">My blog history</aside>
</div>
</body>
</html>
```

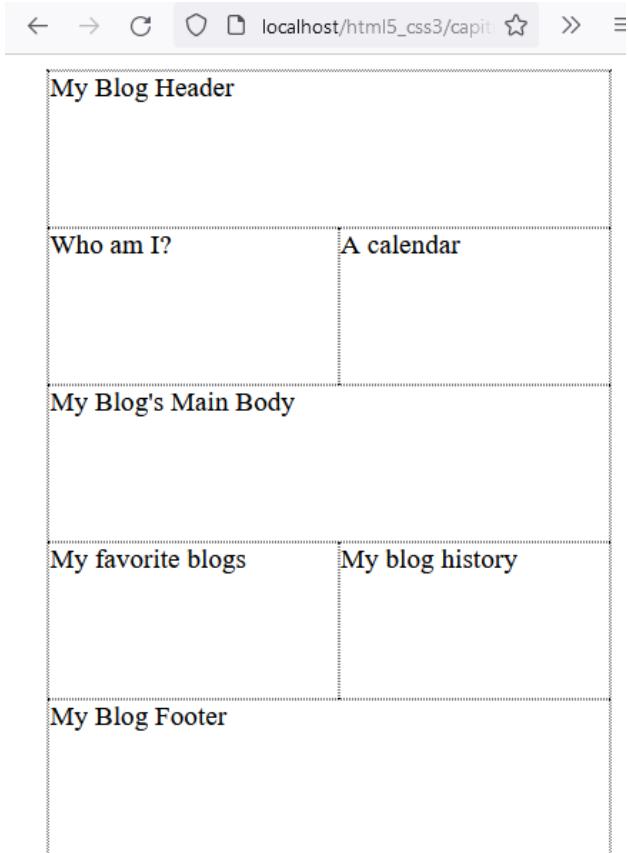
Acima de 1200px, temos:



De 400px a 1200px, temos:



Abaixo de 400px, temos:



```
<link rel="stylesheet" media="screen and (min-width: 1200px)" href="Desktop.css"/>
<link rel="stylesheet" media="screen and (max-width: 1199px) and (min-width: 401px)" href="tablet.css"/>
<link rel="stylesheet" media="screen and (max-width: 400px)" href="phone.css"/>
```

## Ocultando ou desativando controles

A capacidade de modificar o posicionamento da interface do usuário usando consultas de mídia é muito útil. Se em algum caso o layout não funcionar, você pode querer completar os controles de ocultação ou desativar os controles. Os elementos HTML são visíveis por padrão. Entretanto, eles podem se tornar invisíveis ao definir a propriedade CSS de visibilidade como mostrado no seguinte código:

```
.myhiddenelements {  
    visibility:hidden;  
}
```

Ao definir a visibilidade como hidden, o controle não é visível para o usuário final da página web porém, o layout geral se comporta como se o elemento estivesse lá. Se você preferir ter o elemento escondido e o layout se comportar como se não estivesse lá, a propriedade de exibição deve ser usada como mostrado no código a seguir:

```
.myhiddenelements {  
    display: none;  
}
```

## Resumo

- Os elementos HTML podem ser manipulados com transições utilizando as propriedades de CSS: transition, transition-duration, e transition-delay.
- Os elementos podem ser manipulados em 2-D e 3-D com efeitos como translate, scale, rotate, and matrix.
- Media queries permitem que você tenha uma interface de usuário dinâmica e responsiva, baseada no tamanho e tipo do dispositivo.
- A propriedade visibility esconde um controle, mas mantém sua posição no layout geral. Usando a propriedade display para ocultar um controle ele também será removido do layout

## Objective Reviews

1. Which of the following statements will hide an element but keep its space in the overall flow?

- A. **display: none;**
- B. **visibility: hidden;**
- C. **display: inline;**
- D. **visibility: visible;**

A. Correto: **display:none;** esconderá um elemento, mas manterá seu espaço no fluxo total.

B. Incorreto: **visibility: hidden;** manterá o elemento no fluxo normal.

C. Incorreto: **display: inline;** irá mostrar o elemento no fluxo normal.

D. Incorreta: **visibility: visível;** mostrará o elemento no fluxo normal.

2. Media queries are best suited for what purpose?

- A. Setting the priority of style sheet references in a webpage
- B. **Creating a responsive user interface based on the screen size of the view port**
- C. Modifying the view port to properly fit the content of the page
- D. Connecting to third-party style sheets to alter the layout

A. Incorreto: As consultas de mídia não são as mais adequadas para definir a prioridade da folha de estilo referidas em uma página da web.

B. Correto: As consultas de mídia são mais adequadas para a criação de uma interface de usuário responsiva com base no tamanho da tela de visualização do dispositivo.

C. Incorreto: As consultas de mídia não são mais adequadas para modificar a visualização do dispositivo para se ajustar adequadamente ao conteúdo da página.

D. Incorreta: As consultas de mídia não são mais adequadas para conexão a folhas third-party para alterar o layout.

3. Which of the following transition-timing-function properties makes the transition start slow, speed up, then end slow?

- A. **ease**
- B. **ease-in**
- C. **ease-out**
- D. **ease-in-out**

A. Correto: **ease** começará devagar, acelerará e terminará devagar. Este também é o valor padrão.

B. Incorreto: **ease-in** começará devagar e acelerará.

C. Incorreto: **easy-out** irá diminuir a velocidade perto do final.

D. Incorreto: **ease-in-out** vai diminuir a velocidade no início e no final.

## Aula 5 - Encontrando elementos com o uso de seletores CSS e jQuery

Vamos explorar as mais avançadas técnicas para encontrar elementos através do uso de seletores CSS e jQuery.

### Objetivos

- Definir seletores de elementos, estilo e atributos
- Escolher o seletor correto para referenciar a um elemento
- Encontrar elementos usando pseudo-elementos e pseudo-classes

### Definindo seletores de elementos, estilo e atributos

CSS utiliza seletores em um arquivo CSS ou bloco de estilo para identificar quais elementos em uma página web a que os estilos definidos devem ser aplicados. Isto pode ser feito especificando o elemento como o próprio selecionador. Por exemplo, o seguinte CSS é um seletor de elementos para o elemento div:

```
div{  
    ...  
}
```

Qualquer elemento que tenha a classe mycustomclass atribuída a ele através do atributo de classe terá os estilos definidos aplicados.

```
.mycustomclass{  
    ...  
}
```

Outra maneira de usar o CSS para selecionar elementos específicos na página é usar a seleção de atributos. Isto é conseguido especificando um tipo de elemento seguido por um atributo específico. Por exemplo, se você tem um formulário web que precisa ser preenchido, você pode atribuir campos obrigatórios com uma borda vermelha ao redor das caixas de texto. O seguinte código consegue isso para quaisquer elementos que tenham o atributo required especificado:

```
input[required] {  
    border: 1px vermelho sólido;  
}
```

Existem outras possibilidades para o uso de seletores de atributos. Estas são descritas na Tabela 4-12:

**TABLE 4-12** Attribute selector capabilities

Attribute selector	Description
=	Specifies that an attribute equals a specific value. For example, the URL of an anchor is a specify URL.
~=	Specifies a space-separated list of words as the values for an attribute.
^=	Specifies that the attribute has a value starting with the text specified.
\$=	Specifies that the attribute has a value ending with the specified text.
*=	Specifies that the attribute has a value containing the specified text.

## Escolhendo o seletor correto para referenciar um elemento

Por exemplo, para afetar todos os elementos article use:

```
article{  
    border-color: 1px solid red;  
}
```

Se você não quer afetar todos os elementos article, mas apenas o artigo mais novo, você pode distinguir acrescentando uma classe CSS personalizada atribuída apenas ao artigo mais recente:

```
article.newest{  
    border-color: 1px solid red;  
}
```

## Buscando elementos com o uso de pseudo-elementos e pseudo-classes

Pseudo-classes e pseudo-elementos fornecem algumas formas muito poderosas de acrescentar estilos a elementos. Pseudo-classes permitem aplicar estilos a um elemento com base em seu estado, sua interação com o usuário, ou sua posição no documento. Pseudo-elementos permitem a inserção de conteúdo na página em locais relativos aos elementos aos quais o CSS está sendo aplicado.

### :link, :visited, and :hover

```
a:link {  
    color: green;  
}  
a:hover {  
    color: red;  
}  
a:visited {  
    color: black;  
}
```

## **:checked**

Pode ser usado em check boxes and radio buttons.

O próximo exemplo mostra como esconder um check box quando o usuário clicar nele.

```
input[type="checkbox"]:checked {  
    display: none;  
}
```

## **:required**

O próximo exemplo mostra como aplicar estilo para todos os inputs com o atributo required:

```
input:required {  
    border: 2px solid red;  
}
```

## **:enabled and :disabled**

Com estas pseudo classes, você pode controlar quais elementos input podem ser habilitados ou desabilitados:

```
input:disabled {  
    background-color: blue;  
}  
input:enabled {  
    background-color: white;  
}
```

## **:first-child**

O pseudo-elemento :first-child aplica os estilos especificados para a primeira instância do elemento que ocorre em uma lista, por exemplo, o elemento do primeiro parágrafo deste HTML:

```
<div>  
    <p>Lorem Ipsum ...</p>  
    <p>Lorem Ipsum ...</p>  
    <p>Lorem Ipsum ...</p>  
    <p>Lorem Ipsum ...</p>  
</div>
```

O seguinte CSS mudará a cor do texto para verde no elemento do primeiro parágrafo:

```
p:first-child {  
    color:green;  
}
```

## **:first-letter**

O pseudo-elemento :first-letter alterará o estilo da primeira letra no elemento especificado.

```
p::first-letter {  
    font-size: xx-large;  
}
```

## **:before e :after**

Os pseudo-elementos :before e :after adicionaram o conteúdo especificado na frente ou depois do seletor de elementos indicado. Portanto, o seguinte código adicionaria \*\* à frente e ao final do elemento de parágrafo:

```
p::before {  
    content: '**';  
}  
  
p::afters {  
    content: '**';  
}
```

## **:first-line**

O pseudo-elemento :first-line altera os estilos da primeira linha de um elemento de texto. O seguinte CSS tornará a primeira linha do texto dentro do elemento parágrafo verde e maior:

```
p::first-line {  
    color:green;  
    font-size: x-large;  
}
```

## **Resumo**

- Pseudo-elementos e pseudo-classes fornecem um mecanismo avançado de busca de elementos HTML em uma página e estilos de aplicação.
- Usando pseudo-elementos e pseudo-classes você pode mudar o estilo de um elemento com base nas ações dos usuários.
- Usando pseudo-elementos e pseudo-classes você obter controle sobre as partes do texto em um bloco de texto

## Objective Review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the “Answers” section at the end of this chapter.

1. Which one of the following is a CSS class selector?

- A. `.code`
- B. `#code`
- C. `div[code]`
- D. `:code`

A. Correto: `.code` é um seletor de classe CSS.

B. Incorreto: `#code` é um seletor de ID.

C. Incorreto: `div[code]` é um seletor de atributos.

D. Incorreto: `:code` não é uma declaração válida.

2. Which one of the following is an attribute selector?

- A. `.required`
- B. `#required`
- C. `input[required]`
- D. `:required`

A. Incorreto: `.required` é um seletor de classes.

B. Incorreto: `#required` é um seletor de identificação.

C. Correto: `input[required]` é um seletor de atributos.

D. Incorreto: `:requerid` é uma pseudo-classe que seria combinada com um elemento seletor.

3. Which of the following statements would alter the style of an anchor element when the mouse is moved over it?

- A. `a:link`
- B. `a:hover`
- C. `a:beforeclick`
- D. `a:mouseover`

A. Incorreto: `a:link` especificaria os estilos para um hyperlink não visitado.

B. Incorreto: `a:hover` não é uma pseudo-classe válida.

C. Incorreto: `a:beforeclick` não é uma pseudo-classe válida.

D. Correto: `a:hover` mudará o estilo quando o usuário passar o mouse sobre o link

## Aula 6 - Estruturando um arquivo CSS com o uso de seletores

Os arquivos CSS podem se tornar grandes e complexos. Estruturando-os de forma organizada torna-se mais fácil mantê-los e também de saber quais são os seletores mais adequados para serem usados fazendo referência a eles em sua página HTML.

### Objetivos

- Referenciar corretamente os elementos
- Implementar herança
- Substituir a herança usando !important

### Referenciando corretamente os elementos

O CSS é usado para aplicar estilos aos elementos em uma página HTML. Para isso, o CSS tem que saber quais elementos deve aplicar os estilos. Há algumas maneiras de se referenciar elementos do CSS. A consideração chave é garantir que você faça referência a apenas aos elementos que deseja que sejam afetados.

```
p{  
    ...  
}
```

Serão selecionados todos os parágrafos.

```
article{  
    ...  
}
```

Serão selecionados todos os articles

```
div{  
    ...  
}
```

Serão selecionados todas os div's

```
.bold{  
    ...  
}
```

Serão selecionados todos os elementos com class="bold"

```
.largeTitle{  
    ...  
}
```

Serão selecionados todos os elementos com class="largeTitle"

```
#nameBox{  
    ...  
}
```

```
}
```

Será selecionado o elemento com id="nameBox"

```
p, H1, H2 {  
    ...  
}
```

Serão selecionados todos os parágrafos e títulos H1 e H2.

## Implementando herança

Alguns estilos aplicados a um elemento pai são automaticamente herdados por elementos children. Por exemplo, se uma série de parágrafo estão dentro de um article, e estilos de fonte e texto estão aplicados ao article, todos os parágrafos herdarão automaticamente esses estilos de fonte e texto.

## Sobrescrevendo herança com !importante

O CSS para grandes sites pode ser complicado. Os grandes sites podem ter CSS provenientes de diferentes fontes. Ele pode estar em cada página e ser referenciado externamente. As bibliotecas externas são mais e mais comuns como especialistas em toda a comunidade criaram temas que podem ser importados em suas aplicações web. Com todo este estilo vindo de diferentes fontes, a herança de estilos pode ser complicada. Em alguns casos, você pode precisar anular todos os outros estilos e considerar apenas um estilo desejado. Aqui é onde a palavra-chave !importante entra.

Exemplo:

```
p{  
    font-family: serif;  
    color: blue;  
}  
p{  
    color: purple !important;  
}  
p{  
    color: yellow;  
}
```

Todos os parágrafos assumirão a cor purple (lilás) e não a cor amarela.

## Resumo

- Os elementos de referência levam em consideração a forma como você irá estruturar corretamente seu CSS e seus elementos HTML.
- Os selecioadores podem ser aninhados e unidos para serem mais específicos.
- Os elementos HTML herdam automaticamente os estilos de seus elementos pais.
- O CSS é processado de cima para baixo, de modo que o último estilo processado ganha se entrar em conflito com outras declarações de estilo.
- !important pode ser usado para garantir que o estilo desejado seja apresentado quando há uma declaração concorrente do CSS.

## Objective Review

Answer the following questions to test your knowledge of the information in this objective. You can find the answers to these questions and explanations of why each answer choice is correct or incorrect in the “Answers” section at the end of this chapter. The review questions use the following HTML listing (line numbers are for reference only):

```
1. <html>
2. <body>
3.   <div>
4.     <hgroup>
5.       <h1></h1>
6.       <h2></h2>
7.     </hgroup>
8.   </div>
9.   <div>
10.    <section>
11.      <article>
12.        <h1></h1>
13.        <p></p>
14.        <p></p>
15.      </article>
16.      <article>
17.        <h1></h1>
18.        <p></p>
19.        <aside></aside>
20.        <p></p>
21.      </article>
22.    </section>
23.  </div>
24.  <div>
25.    <footer>
26.      <p></p>
27.      <p></p>
28.    </footer>
29.  </div>
30.</body>
31.</html>
```

1. Referencing the HTML listing, how would you style only the first paragraph inside the footer element to have a smaller font size?

A.

```
footer p:first-child {  
    font-size: x-small;  
}
```

B.

```
footer p.first-child {  
    font-size: x-small;  
}
```

C.

```
Footer:p:first-child {  
    font-size: x-small;  
}
```

D.

```
Footer=>p,first-child {  
    font-size: x-small;  
}
```

A. Correto: Esta é a sintaxe correta especificando o primeiro parágrafo filho do elemento footer para ter uma fonte menor.

B. Incorreto: O p.firstchild não é a sintaxe correta. Ela precisaria ser p:first-child

C. Incorreto: Os dois pontos (:) após o Footer não está com a sintaxe correta.

D. Incorreto: A notação => não é a sintaxe correta.

2. Referencing the HTML listing, how would you apply a new font to all the H1 elements? In addition, the <h1> elements in an article should be italic.

A.

```
h1 {  
    font-family: 'Courier New';  
    article h1 {  
        font-style:italic;  
    }  
}
```

B.

```
h1 {  
    font-family: 'Courier New';  
}  
article h1 {  
    font-style:italic;  
}
```

C.

```
h1 {  
    font-family: 'Courier New';  
    font-style:italic;  
}  
article h1 {  
    font-style:italic;  
}
```

- A. Incorreto: Os estilos não podem ser aninhados uns dentro dos outros desta maneira.
- B. Correto: É correto especificar primeiro o estilo H1, depois especificar os estilos H1 para elementos H1 que estão abaixo de um elemento article.
- C. Incorreto: Isto fará com que todos os elementos H1 fiquem em itálico.
- D. Incorreto: Isto fará com que todos os elementos H1 fiquem em itálico

D.

```
h1 {  
    font-family: 'Courier New';  
}  
article, h1 {  
    font-style:italic;  
}
```

3. Referencing the preceding HTML listing, write the CSS code to apply a border to the aside element that is 100 pixels high and 50 pixels wide. In addition, provide a shadow effect and slightly skew the element to the right 5 degrees