Este é o CS50

Introdução à Ciência da Computação (CS50)

OpenCourseWare

Doar (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Aula 8

- Bem-vindo!
- A Internet
- Roteadores
- DNS
- DHCP
- HTTPS
- HTML
- Expressões regulares
- CSS
- Estruturas
- JavaScript
- Resumindo

Bem-vindo!

- Nas semanas anteriores, apresentamos a vocês o Python, uma linguagem de programação de alto nível que utiliza os mesmos blocos de construção que aprendemos em C. Hoje, vamos expandir esses blocos de construção ainda mais, utilizando HTML, CSS e JavaScript.

A Internet

- A internet é uma tecnologia que todos nós usamos.
- Utilizando as habilidades adquiridas nas semanas anteriores, podemos construir nossas próprias páginas e aplicativos da web.
- A *ARPANET* conectou os primeiros pontos da internet uns aos outros.
- Os pontos entre dois pontos podem ser considerados *roteadores* .

Roteadores

- Para encaminhar dados de um local para outro, precisamos tomar *decisões de roteamento* . Ou seja, alguém precisa programar como os dados serão transferidos do ponto A para o ponto B.
- É possível imaginar como os dados podem percorrer múltiplos caminhos entre o ponto A e o ponto B, de forma que, quando um roteador estiver congestionado, os dados possam fluir por outro caminho. *Pacotes* de dados são transferidos de um roteador para outro, de um computador para outro.
- *TCP/IP* são dois protocolos que permitem que computadores transfiram dados entre si pela internet.
- *IP* , ou *protocolo de internet* , é uma forma pela qual os computadores podem se identificar na internet. Cada computador tem um endereço único no mundo. Os endereços têm este formato:

  ```
  #.#.#.#
  ```

- Os números variam de 1000 `0` a 10000. `255` Os endereços IP têm 32 bits, o que significa que esses endereços poderiam acomodar mais de 4 bilhões de endereços. Versões mais recentes de endereços IP, que implementam 128 bits, podem acomodar muito mais computadores!
- No mundo real, os servidores fazem muito trabalho para nós.
- Os pacotes estão estruturados da seguinte forma:

  ```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |Version|  IHL  |Type of Service|          Total Length         |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |         Identification        |Flags|      Fragment Offset    |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |  Time to Live |    Protocol   |         Header Checksum       |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                       Source Address                          |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                    Destination Address                        |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  ```

```
|                     Options                     |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Os pacotes são padronizados. A origem e o destino estão contidos em cada pacote.
- *O TCP* , ou protocolo de controle de transmissão, ajuda a manter o controle da sequência de pacotes que estão sendo enviados.
- Além disso, o TCP é usado para distinguir serviços web uns dos outros. Por exemplo, 123456789 `80` é usado para denotar HTTP e 123456789 `443` é usado para denotar HTTPS. Esses números são *números de porta* .
- Quando informações são enviadas de um local para outro, são enviados um endereço IP de origem, um endereço IP de destino e um número de porta TCP.
- Esses protocolos também são usados para fragmentar arquivos grandes em várias partes ou pacotes. Por exemplo, uma foto grande de um gato pode ser enviada em vários pacotes. Quando um pacote é perdido, o TCP/IP pode solicitar novamente os pacotes ausentes ao servidor de origem.
- O TCP confirmará o recebimento de todos os dados.

DNS

- Seria muito tedioso ter que memorizar um endereço IP para visitar um site.
- *DNS* , ou *sistema de nomes de domínio* , é um conjunto de servidores na internet que são usados para direcionar endereços de sites, como *harvard.edu,* para um endereço IP específico.
- O DNS é simplesmente uma tabela ou banco de dados que vincula nomes de domínio específicos e totalmente qualificados a endereços IP específicos.

DHCP

- *DHCP* é um protocolo que determina o endereço IP do seu dispositivo.
- Além disso, este protocolo define o gateway padrão e os servidores de nomes que seu dispositivo utiliza.

HTTPS

- *O HTTP,* ou *protocolo de transferência de hipertexto* , é um protocolo de camada de aplicação que os desenvolvedores usam para criar ferramentas poderosas e úteis através da transferência de dados de um local para outro. *O HTTPS* é uma versão segura desse protocolo.
- Quando você vê um endereço como esse, `https://www.example.com` você está implicitamente visitando aquele endereço com um "a" `/` no final.

- O *caminho* é o que existe depois dessa barra. Por exemplo, `https://www.example.com/folder/file.html` visita `example.com` e navega até o `folder` diretório e, em seguida, visita o arquivo chamado `file.html` .
- Isso `.com` é chamado de *domínio de nível superior* e é usado para denotar a localização ou o tipo de organização associada a esse endereço.
- `https` Neste endereço está o protocolo usado para conectar-se a esse endereço web. Por protocolo, queremos dizer que o HTTP utiliza *requisições* `GET` para solicitar informações de um servidor. Por exemplo, você pode abrir o Google Chrome, clicar com o botão direito e selecionar "Conectar" . Ao abrir o navegador e visitar o endereço , selecionando "Conectar" , você verá o endereço . Você verá menções a "https://example.com" . Isso também é possível em outros navegadores, usando métodos ligeiramente diferentes. `POST` `inspect` `developer tools` `Network` `Preserve log` `Request Headers` `GET`
- Por exemplo, ao enviar uma solicitação GET, seu computador pode enviar o seguinte para um servidor:

  ```
  GET / HTTP/2
  Host: www.harvard.edu
  ```

  Observe que esta solicitação é feita via HTTP para acessar o conteúdo hospedado em www.harvard.edu.
- Geralmente, após fazer uma solicitação a um servidor, você receberá o seguinte em `Response Headers` :

  ```
  HTTP/2 200
  Content-Type: text/html
  ```

- Essa abordagem para inspecionar esses registros pode ser um pouco mais complicada do que o necessário. Você pode analisar o funcionamento dos protocolos HTTP em cs50.dev (https://cs50.dev) . Por exemplo, digite o seguinte na janela do seu terminal:

  ```
  curl -I https://www.harvard.edu/
  ```

  Observe que o resultado deste comando retorna todos os valores de cabeçalho das respostas do servidor.
- Por meio das ferramentas de desenvolvedor do seu navegador, você pode visualizar todas as requisições HTTP ao acessar o site acima.
- Além disso, execute o seguinte comando na janela do terminal:

  ```
  curl -I https://harvard.edu
  ```

  Observe que você verá uma `301` resposta, fornecendo uma dica ao navegador sobre onde ele pode encontrar o site correto.
- Da mesma forma, execute o seguinte comando na janela do terminal:

```
curl -I http://www.harvard.edu/
```

Observe que o `s` "in" `https` foi removido. A resposta do servidor mostrará que a resposta é `301`, o que significa que o site foi movido permanentemente.

- Semelhante a `301`, um código de `404` significa que um URL específico não foi encontrado. Existem vários outros códigos de resposta, como:

```
200 OK
301 Moved Permanently
302 Found
304 Not Modified
307 Temporary Redirect
401 Unauthorized
403 Forbidden
404 Not Found
418 I'm a Teapot
500 Internal Server Error
503 Service Unavailable
```

- Vale ressaltar que `500` os erros são sempre de sua responsabilidade como desenvolvedor quando se referem a um produto ou aplicativo de sua criação. Isso será especialmente importante para a lista de exercícios da próxima semana e, potencialmente, para o seu projeto final!

HTML

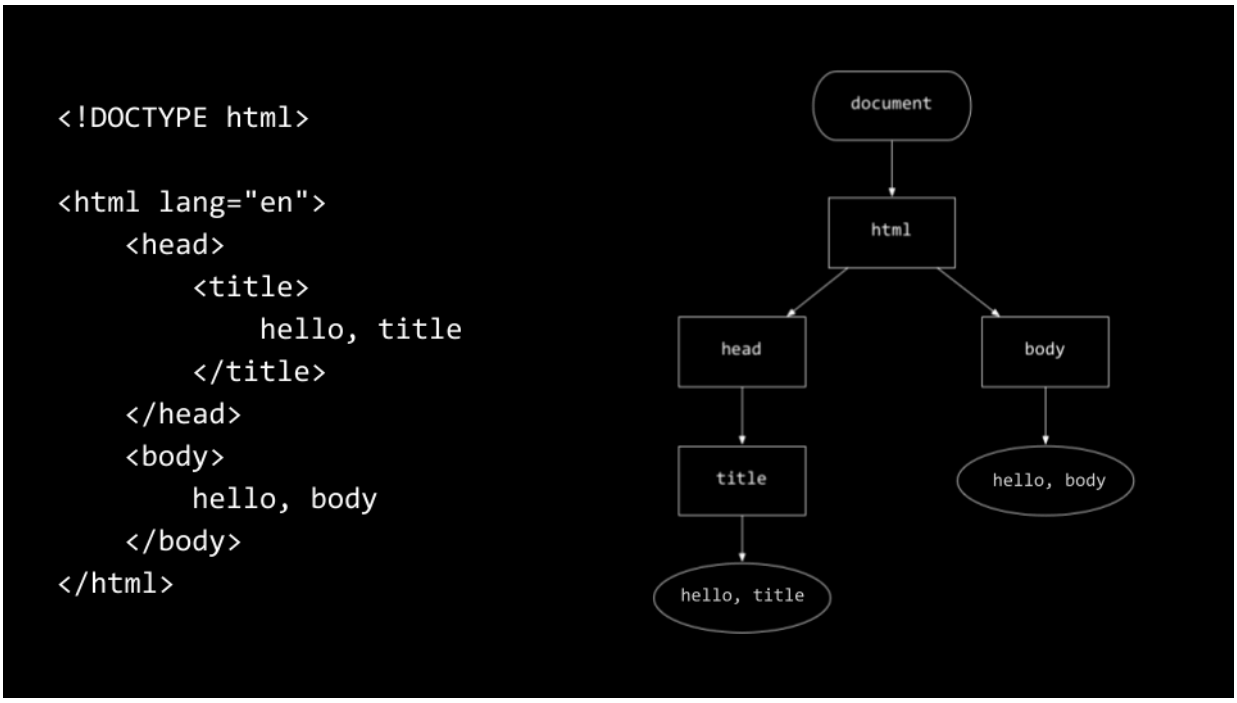
- *HTML,* ou *linguagem de marcação de hipertexto,* é composta por *tags* , cada uma das quais pode ter alguns *atributos* que a descrevem.
- No seu terminal, digite `code hello.html` e escreva o código da seguinte forma:

```
<!DOCTYPE html>

<!-- Demonstrates HTML -->

<html lang="en">
    <head>
        <title>hello, title</title>
    </head>
    <body>
        hello, body
    </body>
</html>
```

Observe que a `html` tag abre e fecha este arquivo. Além disso, observe o `lang` atributo, que modifica o comportamento da `html` tag. Observe também que existem `head` tags `<script>` e `body` `<script>`. O recuo não é obrigatório, mas sugere uma hierarquia.

- Você pode exibir seu código digitando `http-server`. O conteúdo exibido agora está disponível em uma URL bem longa. Se você clicar nela, poderá visitar o site gerado pelo seu próprio código.

- Ao visitar este URL, observe que o nome do arquivo `hello.html` aparece no final da URL. Além disso, observe, com base no URL, que o servidor está utilizando a porta 8080.
- A hierarquia de tags pode ser representada da seguinte forma:



- O conhecimento dessa hierarquia será útil mais tarde, quando aprendermos JavaScript.
- O navegador lerá seu arquivo HTML de cima para baixo e da esquerda para a direita.
- Como os espaços em branco e os recuos são efetivamente ignorados em HTML, você precisará usar `<p>` tags de parágrafo para abrir e fechar um parágrafo. Considere o seguinte:

```
<!DOCTYPE html>

<!-- Demonstrates paragraphs -->

<html lang="en">
    <head>
        <title>paragraphs</title>
    </head>
    <body>
        <p>
            Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus c
        </p>
        <p>
            Mauris ut dui in eros semper hendrerit. Morbi vel elit mi. Sed sit
        </p>
        <p>
            Aenean venenatis convallis ante a rhoncus. Nullam in metus vel dia
        </p>
        <p>
            Integer at justo lacinia libero blandit aliquam ut ut dui. Quisque
        </p>
        <p>
            Suspendisse rutrum vestibulum odio, sed venenatis purus condimentu
        </p>
        <p>
            Sed quis malesuada mi. Nam id purus quis augue sagittis pharetra.
```

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

Observe que os parágrafos começam com uma `<p>` tag e terminam com uma `</p>` tag.

- O HTML permite a representação de cabeçalhos:

```
<!DOCTYPE html>

<!-- Demonstrates headings (for chapters, sections, subsections, etc.) -->

<html lang="en">

    <head>
        <title>headings</title>
    </head>

    <body>

        <h1>One</h1>
        <p>
            Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus c
        </p>

        <h2>Two</h2>
        <p>
            Mauris ut dui in eros semper hendrerit. Morbi vel elit mi. Sed sit
        </p>

        <h3>Three</h3>
        <p>
            Aenean venenatis convallis ante a rhoncus. Nullam in metus vel dia
        </p>

        <h4>Four</h4>
        <p>
            Integer at justo lacinia libero blandit aliquam ut ut dui. Quisque
        </p>

        <h5>Five</h5>
        <p>
            Suspendisse rutrum vestibulum odio, sed venenatis purus condimentu
        </p>

        <h6>Six</h6>
        <p>
            Sed quis malesuada mi. Nam id purus quis augue sagittis pharetra.
        </p>

    </body>

</html>
```

Observe que `<h1>`, `<h2>`, e `<h3>` denotam diferentes níveis de títulos.

- Também podemos criar listas não ordenadas em HTML:

```
<!DOCTYPE html>

<!-- Demonstrates (ordered) lists -->

<html lang="en">
    <head>
        <title>list</title>
    </head>
    <body>
        <ul>
            <li>foo</li>
            <li>bar</li>
            <li>baz</li>
        </ul>
    </body>
</html>
```

Observe que a `<ul>` tag cria uma lista não ordenada contendo três itens.

- Também podemos criar listas ordenadas em HTML:

```
<!DOCTYPE html>

<!-- Demonstrates (ordered) lists -->

<html lang="en">
    <head>
        <title>list</title>
    </head>
    <body>
        <ol>
            <li>foo</li>
            <li>bar</li>
            <li>baz</li>
        </ol>
    </body>
</html>
```

Observe que a `<ol>` tag cria uma lista ordenada contendo três itens.

- Também podemos criar uma tabela em HTML:

```
<!DOCTYPE html>

<!-- Demonstrates table -->

<html lang="en">
    <head>
        <title>table</title>
    </head>
    <body>
        <table>
            <tr>
                <td>1</td>
                <td>2</td>
                <td>3</td>
            </tr>
            <tr>
                <td>4</td>
```

```
                <td>5</td>
                <td>6</td>
            </tr>
            <tr>
                <td>7</td>
                <td>8</td>
                <td>9</td>
            </tr>
            <tr>
                <td>*</td>
                <td>0</td>
                <td>#</td>
            </tr>
        </table>
    </body>
</html>
```

As tabelas também possuem tags que abrem e fecham cada elemento dentro delas. Observe também a sintaxe para comentários em HTML.

- As imagens também podem ser utilizadas em HTML:

```
<!DOCTYPE html>

<!-- Demonstrates image -->

<html lang="en">
    <head>
        <title>image</title>
    </head>
    <body>
        <img alt="photo of bridge" src="bridge.png">
    </body>
</html>
```

Observe que isso `src="bridge.png"` indica o caminho onde o arquivo de imagem pode ser encontrado.

- Os vídeos também podem ser incluídos em HTML:

```
<!DOCTYPE html>

<!-- Demonstrates video -->

<html lang="en">
    <head>
        <title>video</title>
    </head>
    <body>
        <video controls muted>
            <source src="video.mp4" type="video/mp4">
        </video>
    </body>
</html>
```

Observe que o `type` atributo indica que este é um vídeo do tipo `mp4`. Além disso, observe como `controls` e `muted` são passados para `video`.

- Você também pode criar links entre várias páginas da web:

```
<!DOCTYPE html>

<!-- Demonstrates link -->

<html lang="en">
    <head>
        <title>link</title>
    </head>
    <body>
       Visit <a href="https://www.harvard.edu">Harvard</a>.
    </body>
</html>
```

  Observe que a tag *de âncora* `<a>` ou link é usada para criar um texto clicável. `Harvard`

- Você também pode criar formulários que lembram a busca do Google:

```
<!DOCTYPE html>

<!-- Demonstrates form -->

<html lang="en">
    <head>
        <title>search</title>
    </head>
    <body>
        <form action="https://www.google.com/search" method="get">
            <input name="q" type="search">
            <input type="submit" value="Google Search">
        </form>
    </body>
</html>
```

  Observe que uma `form` tag é aberta e fornece o atributo do que `action` ela irá receber. O `input` campo é incluído, passando o nome `q` e o tipo como `search`.

- Podemos melhorar esta pesquisa da seguinte forma:

```
<!DOCTYPE html>

<!-- Demonstrates additional form attributes -->

<html lang="en">
    <head>
        <title>search</title>
    </head>
    <body>
        <form action="https://www.google.com/search" method="get">
            <input autocomplete="off" autofocus name="q" placeholder="Query" t
            <button>Google Search</button>
        </form>
    </body>
</html>
```

  Observe que `autocomplete` está ativado `off`. `autofocus`

- Vimos apenas alguns dos muitos elementos HTML que você pode adicionar ao seu site. Se você tiver alguma ideia para adicionar algo ao seu site que ainda não vimos (um botão, um arquivo de áudio, etc.), tente pesquisar no Google "X em HTML" para encontrar a sintaxe correta! Da mesma forma, você pode usar o cs50.ai (https://cs50.ai) para descobrir mais recursos do HTML!

Expressões regulares

- *Expressões regulares* , ou *regexes,* são um meio de garantir que os dados fornecidos pelo usuário se ajustem a um formato específico.
- Podemos implementar nossa própria página de cadastro que utiliza expressões regulares da seguinte forma:

```
<!DOCTYPE html>

<!-- Demonstrates type="email" -->

<html lang="en">
    <head>
        <title>register</title>
    </head>
    <body>
        <form>
            <input autocomplete="off" autofocus name="email" placeholder="Emai
            <button>Register</button>
        </form>
    </body>
</html>
```

Observe que a `input` tag inclui atributos que indicam que se trata de um campo do tipo ` `email` <email>`. O navegador sabe que deve verificar se o campo inserido é um endereço de e-mail.

- Embora o navegador utilize esses atributos integrados para verificar um endereço de e-mail, podemos adicionar um `pattern` atributo para garantir que apenas dados específicos sejam incluídos no endereço de e-mail:

```
<!DOCTYPE html>

<!-- Demonstrates pattern attribute -->

<html lang="en">
    <head>
        <title>register</title>
    </head>
    <body>
        <form>
            <input autocomplete="off" autofocus name="email" pattern=".+@.+\.e
            <button>Register</button>
        </form>
```

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

Observe que o `pattern` atributo recebe uma expressão regular para indicar que o endereço de e-mail deve incluir um `@` símbolo e um ponto `.edu`.

- Você pode aprender mais sobre expressões regulares na documentação da Mozilla (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions) . Além disso, você pode consultar o site cs50.ai (https://cs50.ai) para obter dicas.

CSS

- `CSS` CSS, ou *folha de estilo em cascata* , é uma linguagem de marcação que permite ajustar a estética dos seus arquivos HTML.
- O CSS está repleto de *propriedades* , que incluem pares de chave-valor.
- No seu terminal, digite `code home.html` e escreva o código da seguinte forma:

```
<!DOCTYPE html>

<!-- Demonstrates inline CSS with P tags -->

<html lang="en">
    <head>
        <title>css</title>
    </head>
    <body>
        <p style="font-size: large; text-align: center;">
            John Harvard
        </p>
        <p style="font-size: medium; text-align: center;">
            Welcome to my home page!
        </p>
        <p style="font-size: small; text-align: center;">
            Copyright &#169; John Harvard
        </p>
    </body>
</html>
```

Observe que alguns `style` atributos são fornecidos às `<p>` tags. O atributo ` `font-size` is` é definido como `large` `true`, `medium` `false` ou ` `small` false`. Em seguida, o `text-align` atributo `is` é definido como `center`.

- Embora correto, o código acima não está bem projetado. Podemos eliminar a redundância modificando-o da seguinte forma:

```
<!DOCTYPE html>

<!-- Removes outer DIV -->

<html lang="en">
    <head>
        <title>css</title>
```

```
    </head>
    <body style="text-align: center">
        <div style="font-size: large">
            John Harvard
        </div>
        <div style="font-size: medium">
            Welcome to my home page!
        </div>
        <div style="font-size: small">
            Copyright &#169; John Harvard
        </div>
    </body>
</html>
```

Observe que `<div>` as tags são usadas para dividir este arquivo HTML em regiões específicas. O atributo `@body` `text-align: center` é invocado em todo o corpo do HTML. Como tudo dentro de `@body` `body` é filho de `@body` `body`, o `center` atributo se propaga para esses filhos.

- Descobrimos que existem novas tags semânticas incluídas no HTML. Podemos modificar nosso código da seguinte forma:

```
<!DOCTYPE html>

<!-- Uses semantic tags instead of DIVs -->

<html lang="en">
    <head>
        <title>css</title>
    </head>
    <body style="text-align: center">
        <header style="font-size: large">
            John Harvard
        </header>
        <main style="font-size: medium">
            Welcome to my home page!
        </main>
        <footer style="font-size: small">
            Copyright &#169; John Harvard
        </footer>
    </body>
</html>
```

Observe que ambos `header` os `footer` elementos possuem estilos diferentes atribuídos a eles.

- Essa prática de colocar o estilo e as informações no mesmo local não é uma boa prática. Poderíamos mover os elementos de estilo para o início do arquivo, da seguinte forma:

```
<!-- Demonstrates class selectors -->

<html lang="en">
    <head>
        <style>

            .centered
            {
```

```
                text-align: center;
            }

            .large
            {
                font-size: large;
            }

            .medium
            {
                font-size: medium;
            }

            .small
            {
                font-size: small;
            }

        </style>
        <title>css</title>
    </head>
    <body class="centered">
        <header class="large">
            John Harvard
        </header>
        <main class="medium">
            Welcome to my home page!
        </main>
        <footer class="small">
            Copyright &#169; John Harvard
        </footer>
    </body>
</html>
```

Observe que todas as tags de estilo estão posicionadas dentro do elemento `<div>` `head` que envolve a `style` tag `<style>`. Observe também que atribuímos *classes* , chamadas `style` `centered` , ` class` `large` , `medium` `class` e `small` `class`, aos nossos elementos, e que selecionamos essas classes colocando um ponto antes do nome, como em `style`. `.centered`

- Descobrimos que podemos mover todo o nosso código de estilo para um arquivo especial chamado arquivo *CSS* . Podemos criar um arquivo chamado `style.css` `style.css` e colar nossas classes nele:

```
.centered
{
    text-align: center;
}

.large
{
    font-size: large;
}

.medium
{
    font-size: medium;
```

```
}

.small
{
    font-size: small;
}
```

Observe que isso é exatamente o que apareceu em nosso arquivo HTML.

- Em seguida, podemos informar ao navegador onde localizar o CSS para este arquivo HTML:

```
<!DOCTYPE html>

<!-- Demonstrates external stylesheets -->

<html lang="en">
    <head>
        <link href="style.css" rel="stylesheet">
        <title>css</title>
    </head>
    <body class="centered">
        <header class="large">
            John Harvard
        </header>
        <main class="medium">
            Welcome to my home page!
        </main>
        <footer class="small">
            Copyright &#169; John Harvard
        </footer>
    </body>
</html>
```

Observe que ele `style.css` está vinculado a este arquivo HTML como uma folha de estilo, informando ao navegador onde localizar os estilos que criamos.

Estruturas

- Assim como existem bibliotecas de terceiros que podemos usar em Python, também existem bibliotecas de terceiros chamadas *frameworks* que podemos utilizar com nossos arquivos HTML.
- *O Bootstrap* é uma dessas estruturas que podemos usar para embelezar nosso HTML e aperfeiçoar facilmente os elementos de design, tornando nossas páginas mais legíveis.
- Você pode utilizar o Bootstrap adicionando a seguinte `link` tag no cabeçalho `head` do seu arquivo HTML:

```
<head>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstra
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstra
    <title>bootstrap</title>
</head>
```

- Considere o seguinte HTML:

```
<!DOCTYPE html>

<!-- Demonstrates table -->

<html lang="en">
    <head>
        <title>phonebook</title>
    </head>
    <body>
        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Number</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>Carter</td>
                    <td>+1-617-495-1000</td>
                </tr>
                <tr>
                    <td>David</td>
                    <td>+1-617-495-1000</td>
                </tr>
                <tr>
                    <td>John</td>
                    <td>+1-949-468-2750</td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```

  Observe como, ao visualizar uma versão exibida desta página, ela se apresenta de forma bastante simples.

- Agora, considere o seguinte código HTML que implementa o uso do Bootstrap:

```
<!DOCTYPE html>

<!-- Demonstrates table with Bootstrap -->

<html lang="en">
    <head>
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/boot
        <title>phonebook</title>
    </head>
    <body>
        <table class="table">
            <thead>
                <tr>
                    <th scope="col">Name</th>
                    <th scope="col">Number</th>
                </tr>
            </thead>
            <tbody>
```

```
                <tr>
                    <td>Carter</td>
                    <td>+1-617-495-1000</td>
                </tr>
                <tr>
                    <td>David</td>
                    <td>+1-949-468-2750</td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```

Repare como este site está muito mais bonito agora.

- Da mesma forma, considere a seguinte expansão da nossa página de pesquisa criada anteriormente:

```
<!DOCTYPE html>

<!-- Demonstrates layout with Bootstrap -->

<html lang="en">
    <head>
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/boot
        <title>search</title>
    </head>
    <body>

        <div class="container-fluid">

            <ul class="m-3 nav">
                <li class="nav-item">
                    <a class="nav-link text-dark" href="https://about.google/"
                </li>
                <li class="nav-item">
                    <a class="nav-link text-dark" href="https://store.google.c
                </li>
                <li class="nav-item ms-auto">
                    <a class="nav-link text-dark" href="https://www.google.com
                </li>
                <li class="nav-item">
                    <a class="nav-link text-dark" href="https://www.google.com
                </li>
                <li class="nav-item">
                    <a class="nav-link text-dark" href="https://www.google.com
                        <svg xmlns="http://www.w3.org/2000/svg" width="16" hei
                            <path d="M1 2a1 1 0 0 1 1-1h2a1 1 0 0 1 1 1v2a1 1
                        </svg>
                    </a>
                </li>
                <li class="nav-item">
                    <a class="btn btn-primary" href="https://accounts.google.c
                </li>
            </ul>

            <div class="text-center">
```

```
                <!-- https://knowyourmeme.com/memes/happy-cat -->
                <img alt="Happy Cat" class="img-fluid w-25" src="cat.gif">

                <form action="https://www.google.com/search" class="mt-4" meth
                    <input autocomplete="off" autofocus class="form-control fo
                    <button class="btn btn-light">Google Search</button>
                    <button class="btn btn-light" name="btnI">I'm Feeling Luck
                </form>

            </div>

        </div>

    </body>
</html>
```

Esta versão da página é extremamente estilizada, graças ao Bootstrap.

- Você pode aprender mais sobre isso na documentação do Bootstrap (https://getbootstrap.com/docs/) .

JavaScript

- JavaScript é outra linguagem de programação que permite interatividade em páginas da web.
- Considere a seguinte implementação que `hello.html` inclui JavaScript e HTML:

```
<!DOCTYPE html>

<!-- Demonstrates onsubmit -->

<html lang="en">
    <head>
        <script>

            function greet()
            {
                alert('hello, ' + document.querySelector('#name').value);
            }

        </script>
        <title>hello</title>
    </head>
    <body>
        <form onsubmit="greet(); return false;">
            <input autocomplete="off" autofocus id="name" placeholder="Name" t
            <input type="submit">
        </form>
    </body>
</html>
```

Observe como este formulário usa uma `onsubmit` propriedade para acionar um `script` evento encontrado no início do arquivo. O script usa essa propriedade `alert` para

criar um alerta pop-up. `#name.value` Ele acessa a caixa de texto na página e obtém o valor digitado pelo usuário.

- Em geral, misturar o evento `onsubmit` com JavaScript é considerado uma má prática de design. Podemos aprimorar nosso código da seguinte forma:

```
<!DOCTYPE html>

<!-- Demonstrates DOMContentLoaded -->

<html lang="en">
    <head>
        <script>

            document.addEventListener('DOMContentLoaded', function() {
                document.querySelector('form').addEventListener('submit', func
                    alert('hello, ' + document.querySelector('#name').value);
                    e.preventDefault();
                });
            });

        </script>
        <title>hello</title>
    </head>
    <body>
        <form>
            <input autocomplete="off" autofocus id="name" placeholder="Name" t
            <input type="submit">
        </form>
    </body>
</html>
```

Observe que esta versão do código cria um `addEventListener` evento para detectar quando o formulário `submit` é acionado. Observe também como o `isLoadedPage` `DOMContentLoaded` garante que a página inteira seja carregada antes da execução do JavaScript.

- Podemos avançar este código da seguinte forma:

```
<!DOCTYPE html>

<!-- Demonstrates keyup and template literals -->

<html lang="en">
    <head>
        <script>

            document.addEventListener('DOMContentLoaded', function() {
                let input = document.querySelector('input');
                input.addEventListener('keyup', function(event) {
                    let name = document.querySelector('p');
                    if (input.value) {
                        name.innerHTML = `hello, ${input.value}`;
                    }
                    else {
                        name.innerHTML = 'hello, whoever you are';
```

```
                    }
                });
            });

        </script>
        <title>hello</title>
    </head>
    <body>
        <form>
            <input autocomplete="off" autofocus placeholder="Name" type="text"
        </form>
        <p></p>
    </body>
</html>
```

Observe que o DOM é atualizado dinamicamente na memória à medida que o usuário digita um nome. Se houver um valor dentro de `<div>` `input`, ao pressionar `keyup` a tecla Enter, o DOM é atualizado. Caso contrário, o texto padrão é apresentado.

- O JavaScript permite ler e modificar dinamicamente o documento HTML carregado na memória, de forma que o usuário não precise recarregar a página para ver as alterações.
- Considere o seguinte HTML:

```
<!DOCTYPE html>

<!-- Demonstrates programmatic changes to style -->

<html lang="en">
    <head>
        <title>background</title>
    </head>
    <body>
        <button id="red">R</button>
        <button id="green">G</button>
        <button id="blue">B</button>
        <script>

            let body = document.querySelector('body');
            document.querySelector('#red').addEventListener('click', function(
                body.style.backgroundColor = 'red';
            });
            document.querySelector('#green').addEventListener('click', functio
                body.style.backgroundColor = 'green';
            });
            document.querySelector('#blue').addEventListener('click', function
                body.style.backgroundColor = 'blue';
            });

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

Observe que o JavaScript fica à escuta quando um botão específico é clicado. Ao clicar nele, certos atributos de estilo da página são alterados. O elemento `<body>` `body` é

definido como o corpo da página. Em seguida, um ouvinte de eventos aguarda o clique em um dos botões. Então, o elemento `<body>` `body.style.backgroundColor` é alterado.

- Da mesma forma, considere o seguinte:

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <script>

            // Toggles visibility of greeting
            function blink()
            {
                let body = document.querySelector('body');
                if (body.style.visibility == 'hidden')
                {
                    body.style.visibility = 'visible';
                }
                else
                {
                    body.style.visibility = 'hidden';
                }
            }

            // Blink every 500ms
            window.setInterval(blink, 500);

        </script>
        <title>blink</title>
    </head>
    <body>
        hello, world
    </body>
</html>
```

Este exemplo pisca um texto em um intervalo definido. Observe que `window.setInterval` ele recebe dois argumentos: uma função a ser chamada e um período de espera (em milissegundos) entre as chamadas da função.

- Considere a seguinte implementação de JavaScript que completa automaticamente o texto:

```
<!DOCTYPE html>

<html lang="en">

    <head>
        <title>autocomplete</title>
    </head>

    <body>

        <input autocomplete="off" autofocus placeholder="Query" type="text">

        <ul></ul>

        <script src="large.js"></script>
```

```
        <script>

            let input = document.querySelector('input');
            input.addEventListener('keyup', function(event) {
                let html = '';
                if (input.value) {
                    for (word of WORDS) {
                        if (word.startsWith(input.value)) {
                            html += `<li>${word}</li>`;
                        }
                    }
                }
                document.querySelector('ul').innerHTML = html;
            });

        </script>

    </body>
</html>
```

Esta é uma implementação em JavaScript de autocompletar. Ela extrai informações de um arquivo (não mostrado aqui) chamado ` `large.js` words`, que contém uma lista de palavras.

- As funcionalidades do JavaScript são muitas e podem ser encontradas na Documentação do JavaScript (https://developer.mozilla.org/en-US/docs/Web/JavaScript) .

Resumindo

Nesta lição, você aprendeu como criar seus próprios arquivos HTML, estilizar esses arquivos, aproveitar frameworks de terceiros e utilizar JavaScript. Especificamente, discutimos...

- TCP/IP
- DNS
- HTML
- Expressões regulares
- CSS
- Estruturas
- JavaScript

Até a próxima!