

# Programação em JavaScript

ServiceNow - Deloitte

Rodrigo Costa

# Fundamentos Web

## Módulo 1

# CSS3 Fundamentos e Flexbox

Sessão 3

# Objetivo geral

Dotar os formandos dos conhecimentos fundamentais de CSS3, permitindo-lhes compreender o funcionamento do estilo aplicado em cascata e das regras de prioridade, e aplicar layouts flexíveis utilizando Flexbox.

# Objetivos específicos

Ao final da sessão, os formandos deverão ser capazes de:

- Aplicar corretamente as regras de cascata e especificidade do CSS;
- Definir estilos tipográficos e esquemas de cores;
- Compreender e aplicar corretamente o Box Model;
- Criar layouts flexíveis com Flexbox;
- Selecionar e utilizar unidades fixas e relativas (px, %, rem, em) de forma adequada ao contexto e ao layout pretendido.

# Fundamentos do CSS e Cascata

# O que é CSS e para que serve

- CSS significa Cascading Style Sheets, é a linguagem responsável pelo estilo visual das páginas web, ele define:
  - Cores;
  - Tamanho;
  - Espaçamentos;
  - Layout.

# CSS como sistema de regras

- CSS funciona como um **sistema de regras**;
- Várias regras podem aplicar-se ao mesmo elemento;
- O navegador precisa decidir qual regra vence.



# O que significa “cascata” no CSS

- Cascata refere-se à forma como o navegador:
  - Lê as regras;
  - Resolve conflitos.

# Ordem importa

- A ordem das regras influencia o resultado;
- Estilos declarados depois, podem sobrescrever os anteriores;
- Estilos do autor sobrescrevem estilos do navegador.

```
# index.css > ...
1   p {
2   |   color: ■ red;
3   }
4
5   p {
6   |   color: ■ blue;
7   }
8
9   /* Quem vence? */
10
```

# Ordem importa

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

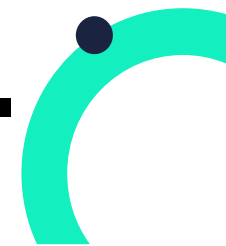
```
p { color: red; }
```

Sou um elemento <p> e minha cor vem da cascata do CSS.

# Quando várias regras entram em conflito

- Quando dois ou mais estilos aplicam-se ao mesmo elemento;
- O navegador segue uma ordem lógica:
  1. Cascata;
  2. Especificidade;
  3. Herança.

# Especificidade e Boas Práticas



# O que é especificidade no CSS

- **Especificidade** define qual regra CSS tem mais prioridade e é usada para determinar quando múltiplos seletores conflitam ao direcionar o mesmo elemento HTML.
- A especificidade é o **segundo critério de decisão do CSS**, em sequência a cascata.

# Por que a especificidade é importante?

- Evita conflitos inesperados;
- Permite prever o resultado final;
- Torna o CSS mais limpo e legível.

# Seletores CSS

- São padrões que identificam quais elementos HTML devem receber estilos e são divididos em sete tipos:
  1. Seletor de tag/elemento (ex: p, h1, div);
  2. Seletor de classe (ex: .nome-da-classe);
  3. Seletor de ID (ex: #nome-do-id);
  4. Seletor universal (\*);
  5. Seletor de atributo (ex: [autoplay], a[target="\_blank"] );
  6. Seletor de pseudo-classe (ex: :hover, :focus, :checked);
  7. Seletor de pseudo-elemento (ex: ::first-line, ::selection).



# Hierarquia de prioridade dos seletores

- Ordem de prioridade (Do menor para o maior prioridade):
  1. Seletor de elemento (tag);
  2. Classe;
  3. ID;
  4. !important força prioridade (evitar).

# Prioridade dos seletores

```
p {
  color: ■ red;
}
/* Baixa prioridade */
```

```
.texto {
  color: ■ blue;
}
/* Prioridade média */
```

```
#texto {
  color: ■ green;
}
/* Prioridade Alta */
```

```
<p class="texto">Texto</p>
```

```
<p id="texto">Texto</p>
```

# Conflito resolvido pela especificidade

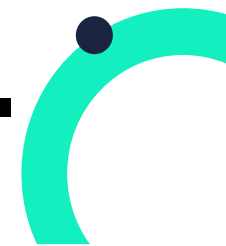
```
p {  
  color: red;  
}
```

```
.texto {  
  color: blue;  
}
```

Classe > Elemento  
(especificidade)

Sou um `<p class="texto">` e minha cor é definida pela especificidade.

# Tipografia e Cores



# O que é Tipografia?

- Tipografia é a forma como o **texto é apresentado** e envolve:
  - Tipo de letra;
  - Tamanho;
  - Espaçamento;
  - Cor.
- A tipografia afeta diretamente a leitura e compreensão.

# Tipografia no CSS

- Tipografia afeta leitura e compreensão;
- Tamanho, espaçamento e fonte importam;
- Texto deve ser legível antes de ser bonito;
- Tipografia legível melhora a acessibilidade e reduz o esforço de leitura.

## Do This

### TYPOGRAPHY

Web typography is one of the essential aspects of designing good websites. This affects, to a greater extent, readability, user engagement, brand identity and ultimately customer experience.

## Don't Do This

### TYPOGRAPHY

Web typography is one of the essential aspects of designing good websites. This affects, to a greater extent, readability, user engagement, brand identity and ultimately customer experience.

# Propriedades básicas de tipografia

```
body {  
  font-family: Arial, sans-serif;  
  font-size: 1rem;  
  line-height: 1.5;  
}
```

# Hierarquia tipográfica

- Hierarquia visual:
  - Diferentes tamanhos ajudam:
    - Distinguir títulos;
    - Separar secções;
    - Guiar a leitura.

```
h1 { font-size: 2rem; }  
h2 { font-size: 1.5rem; }  
p  { font-size: 1rem; }
```



# Cores contraste e acessibilidade

- Cores comunicam hierarquia e significado;
- Contraste insuficiente dificulta a leitura;
- Acessibilidade depende de bom contraste;
- Menos cores resulta em interfaces mais claras.



# Box Model



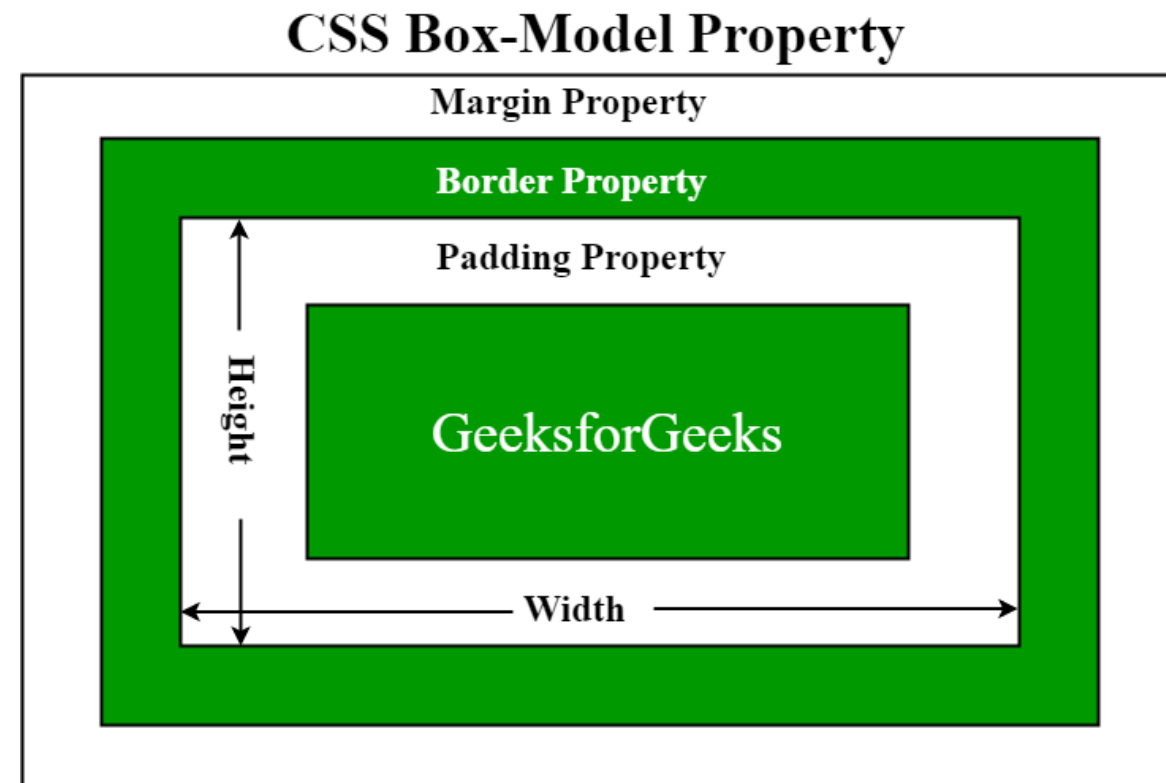
# O que é o Box model?

O **Box Model** é um conceito fundamental em CSS que trata cada elemento HTML como uma caixa retangular, definindo como o espaço é distribuído dentro e ao redor dela, ele é essencial para controlar o layout e a aparência dos elementos na página web.



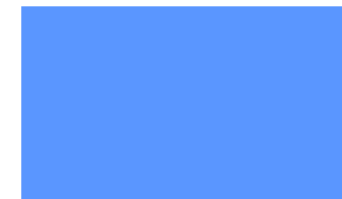
# Box model

- O Box Model é composto por:
  - Content: área onde o conteúdo aparece (texto, imagem, etc.):
    - Definido por: width e height;
  - Padding: espaço entre o conteúdo e a borda, ele aumenta o tamanho visual do elemento;
  - Border: contorno da caixa, também irá contribuir para o tamanho final;
  - Margin: espaço fora da caixa, afasta um elemento dos outros;



# O problema clássico do Box model

```
.box {  
  box-sizing: content-box; /* padrão */  
  width: 200px;  
  padding: 20px;  
  border: 2px solid;  
}  
/* Tamanho real não é 200px */  
/* É maior (content + padding + border) */
```

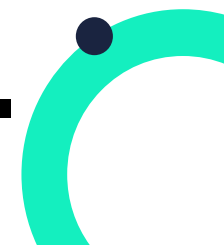


# Box-sizing: border-box

```
.box {  
  box-sizing: border-box;  
}
```

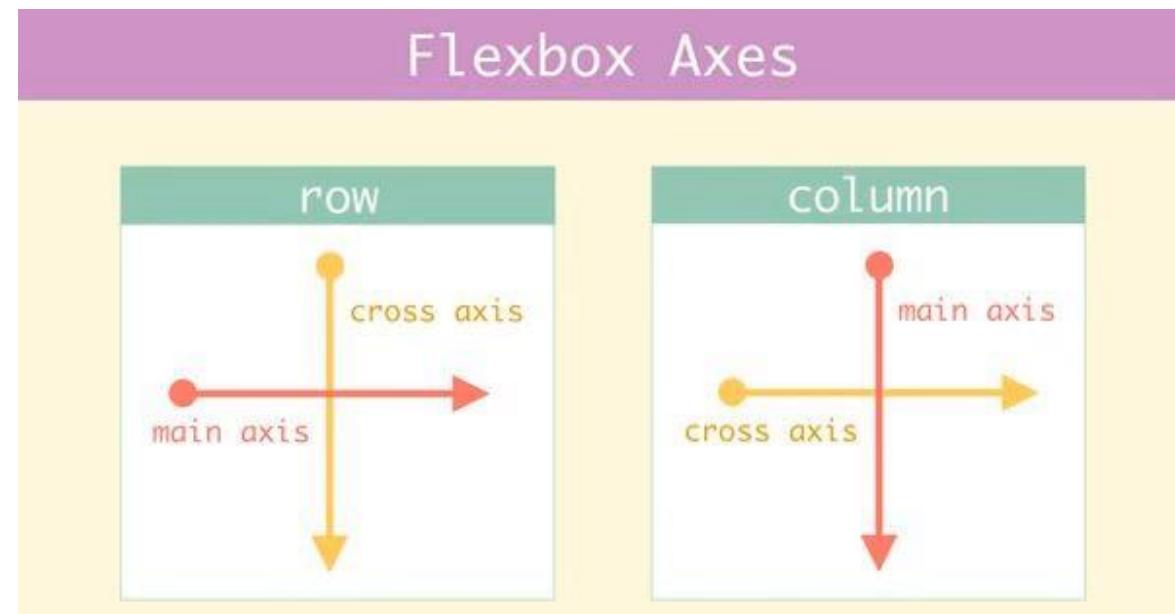
- Padding e border **passam** a serem incluídos dentro das dimensões totais definidas, e não serão mais somados a elas, isso torna o layout muito mais previsível e fácil de gerenciar, evitando que elementos ultrapassem seus containers.

# Flexbox



# Flexbox: Introdução

- Flexbox é um modelo de layout unidimensional;
- Permite alinhar e distribuir espaço entre elementos;
- Facilita layouts flexíveis sem depender de medidas fixas.





# Por que usar Flexbox?

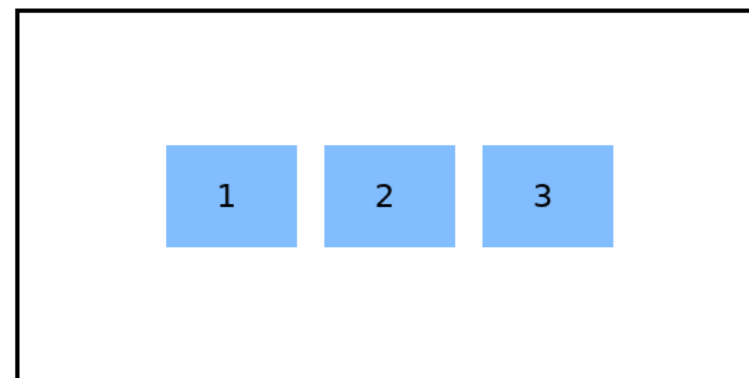
- Problemas que o Flexbox resolve:
  - Alinhamento vertical:
    - Elimina a necessidade de utilização de hacks como **position**, **top**, **margin** ou **transform**;
  - Distribuição de espaço:
    - Elimina a necessidade de utilização de hacks como **float**, **inline-block** e **margin**;
  - Elementos com tamanhos diferentes:
    - Elimina a necessidade de utilização de **JavaScript** para preenchimento de um espaço de forma organizada em layouts complexos;
  - Layout adaptáveis:
    - Permite a flexibilização para novas telas, como **dispositivos móveis** e **TVs de alta resolução**.

# Flexbox: A hierarquia pai-filho

- O elemento (pai): Define o fluxo (direção, alinhamento e distribuição);
- Os elementos (filhos): Definem a flexibilidade (tamanho individual e ordem);
- Regra de Ouro: Propriedades do elemento pai afetam o grupo; propriedades do elemento filho afetam a unidade.

## Flexbox: Hierarquia Pai-Filho

Flex Container (pai)



O container define o fluxo (flex-direction: row)

# Flexbox e a responsividade

- O Flexbox é uns dos elementos responsáveis por facilitar a vida do desenvolvedor para lidar com a **responsividade de uma tela**;
- Responsividade de tela é a capacidade de um site ou aplicação adaptar seu layout e conteúdo automaticamente a diferentes tamanhos de tela (computador, tablet, celular), garantindo uma boa experiência de uso e visualização sem distorções, redimensionando imagens e reorganizando elementos de forma fluida através de técnicas como **unidades de medida relativas** (% , vw, vh) e **Media Queries** no CSS (Próxima aula);

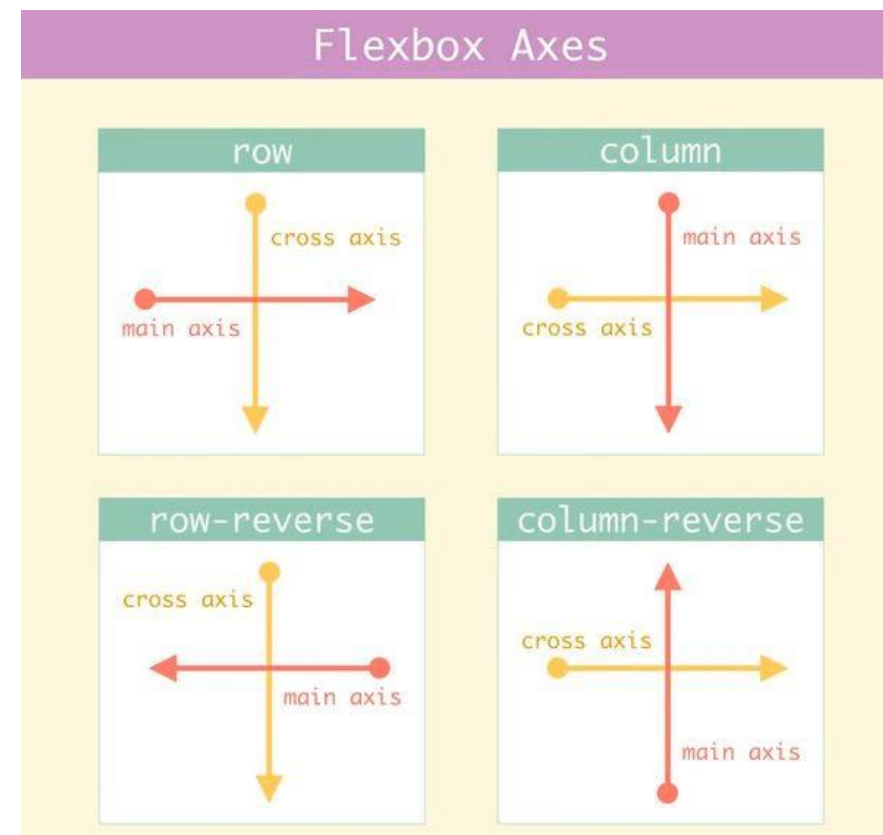
# Flexbox: Estrutura básica

```
.container {  
  display: flex; /* os elementos alinham-se em linha por defeito */  
}
```

```
.container {  
  display: flex;  
  flex-direction: column; /* os elementos alinham-se em coluna */  
}
```

# Eixos do flexbox

- Main axis → eixo principal;
- Cross axis → eixo secundário (cruzado);
- A direção dos eixos muda conforme o valor de flex-direction.

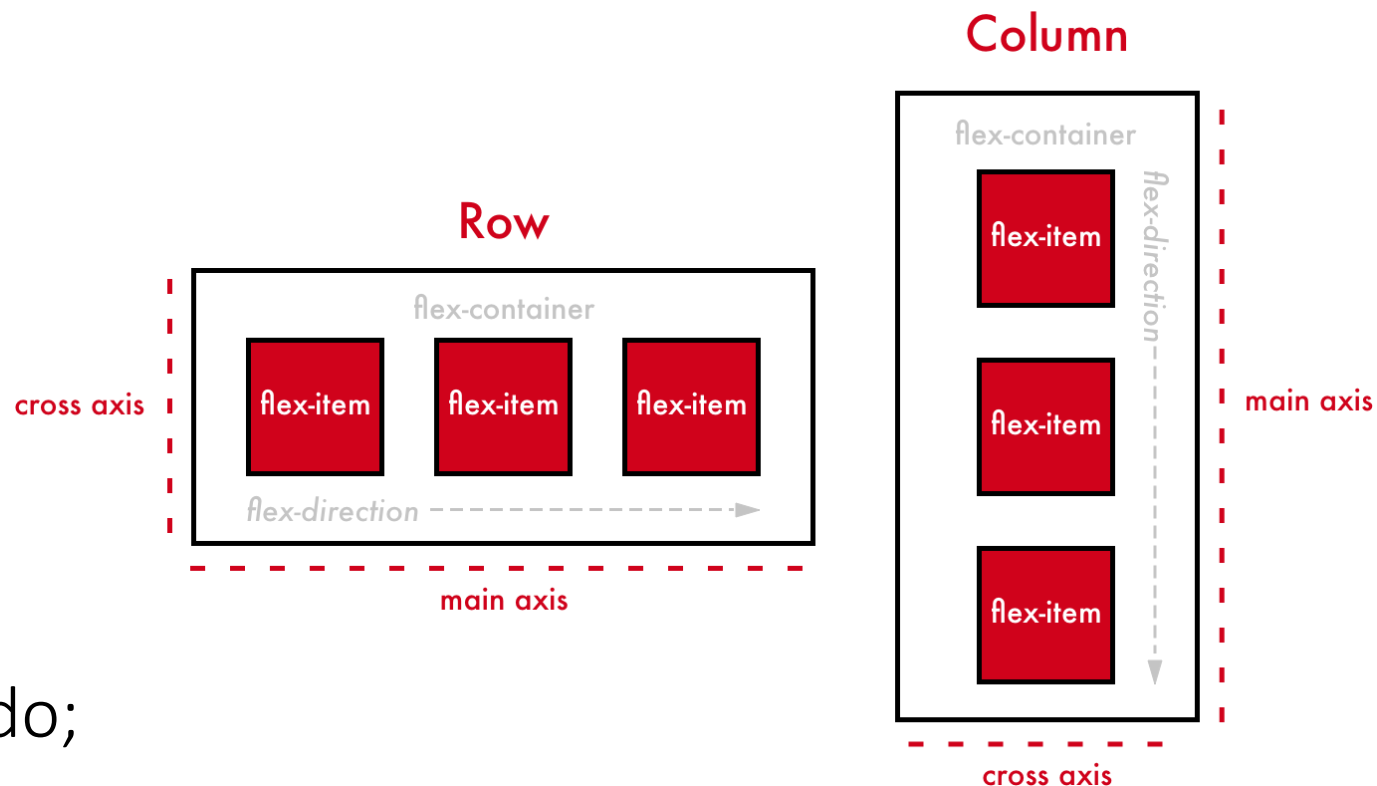


# Justify-content vs align-items

justify-content: center;

align-items: center;

- justify-content atua no eixo principal;
- align-items atua no eixo cruzado;



# Unidades

# O que são as unidades no CSS?

- Unidades CSS são medidas usadas para definir dimensões em uma folha de estilo. Elas dividem-se em **absolutas**, como pixels (px), que possuem valores fixos, e **relativas**, que se adaptam ao contexto: o **em** e o **rem** baseiam-se no tamanho da fonte, enquanto o **%**, **vw** e **vh** baseiam-se nas dimensões do elemento pai ou da janela do navegador. Essa distinção é vital para criar layouts que respondem corretamente tanto a diferentes telas quanto a preferências de acessibilidade do usuário.



# Unidades absolutas

- Possuem valores fixos:
  - px, cm, mm, in, pt
  - Recomendado o seu uso para controle preciso de elementos que não precisam escalar muito, como bordas ou ícones pequenos
- Não se adaptam ao ecrã (escalam mal em diferentes dispositivos), sendo um problema para a responsividade;

# Unidades relativas

- Unidades que se adaptam:
  - % → relativo ao elemento pai [ótimos para larguras e alturas que devem se ajustar ao container];
  - em → relativo ao tamanho da fonte do elemento\*;
  - rem → relativo ao tamanho da fonte raiz (html)\*.
    - \*[ótimos para fontes, margens e paddings, garantindo escalabilidade com o tamanho da fonte].
  - vh/vw: São unidades de medida relativas ao tamanho da área visível do navegador (a viewport). Ao contrário das percentagens (%), que dependem do elemento pai, estas dependem sempre da janela total do utilizador. [ótimos para elementos que devem preencher a tela (banners, secções de altura total)].
- Se adaptam ao ecrã (escalam bem em diferentes dispositivos);

# Cuidados ao utilizar o vh e vw

- vh → viewport height (altura da janela);
- vw → viewport width (largura da janela)
- Em mobile, a altura do viewport pode variar (barra do navegador)
  - 100vh pode causar overflow
- Use quando:
  - O layout depende do tamanho do ecrã;
  - Precisamos de secções full-screen.
- Evite o uso quando:
  - O conteúdo cresce dinamicamente;
  - A altura precisa de ser fluída.

# Unidades de Viewport Avançadas (svh, lvh, dvh)

- svh → small viewport height: Representa a menor altura possível do viewport:
  - Considera o navegador com barras visíveis;
- lvh → large viewport height: Representa a maior altura possível do viewport:
  - Considera o navegador com barras ocultas;
- dvh → dynamic viewport height: Representa a altura dinâmica real do viewport:
  - Ajusta-se automaticamente quando barras aparecem/desaparecem;

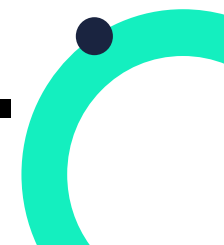
# Qual devo usar?

- Desktop → vh;
- Mobile seguro → svh;
- Layout moderno → dvh;
- Evitar height fixa quando o conteúdo cresce.



# Síntese

- Cascata e Especificidade do CSS;
- Tipografia e Cores;
- Box Model;
- Flexbox;
- Unidades no CSS.



# Conclusão

# Programação em JavaScript

ServiceNow - Deloitte

Rodrigo Costa