Interface

homem-máquina

Inatel | Engenharia de Software

S205 - Interface Homem Máquina

Prof. Raphael C. M. Pereira

Módulos da Disciplina

Habilidades desenvolvidas em laboratório.

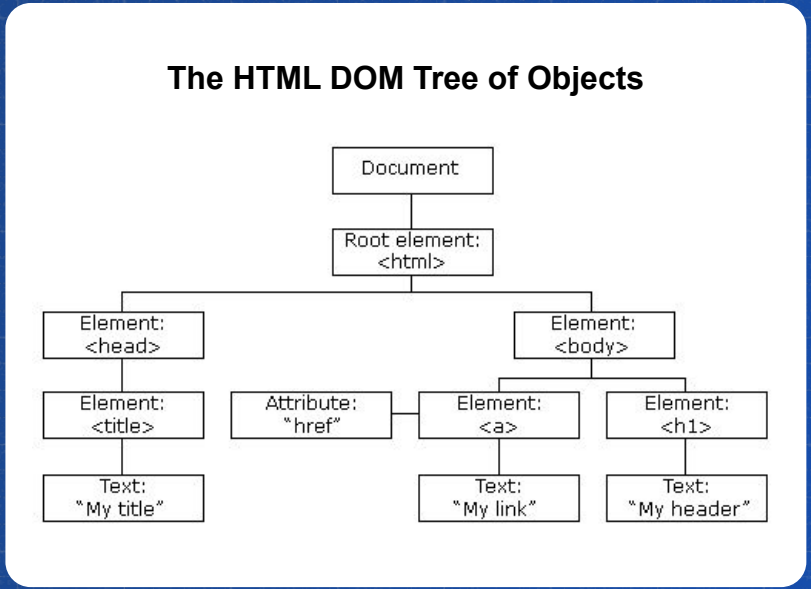
Roteiro da Aula

Principios de Design

- ~~✓ HTML, CSS & JS~~
- ~~✓ Grid e Responsividade~~
- ~~✓ Cores, Tipografia e Ícones~~
- ~~✓ Imagens e Vídeos~~
- ✓ Interação e Animação
- ✓ Componentes
- ✓ Formulários e Visualização de Dados
- ✓ Testes de Usabilidade

Interação e Animação.

Linha de Pensamento



JavaScript HTML DOM

No DOM (Document Object Model), todos os elementos HTML são definidos como objetos, e representados como uma árvore de nós, permitindo que scripts, como JavaScript, alterem o conteúdo, estrutura e estilo da página de forma dinâmica.

Exemplos:

```
Encontre um elemento por ID de elemento
document.getElementById(id)

Alterar o valor do atributo de um elemento HTML
element.attribute = new value

Crie um elemento HTML
document.createElement(element)

Adicionando código do manipulador de eventos a um evento onclick
document.getElementById(id).onclick = function(){code}
```

Document

O objeto **document** é o proprietário de todos os outros objetos em uma página web. O JavaScript pode buscar, alterar, remover, adicionar e criar vários elementos ou atributos HTML, e de estilos CSS de uma página. Assim como, reagir a todos os eventos HTML existentes na página.

No exemplo abaixo, **getElementById** é um método, enquanto **innerHTML** é uma propriedade.

```
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML =
"Hello World!";
</script>

</body>
</html>
```

Methods

A interface de programação são os métodos e propriedades de cada objeto. Um método é uma ação que você pode realizar (como adicionar ou excluir um elemento HTML). Uma propriedade é um valor que você pode obter ou definir (como alterar o conteúdo de um elemento HTML).

Métodos disponíveis no objeto documento:

document.**getElementById**(id): Retorna o elemento com o id especificado.

document.**getElementsByClassName**(class): Retorna uma coleção de todos os elementos com a classe especificada.

document.**getElementsByTagName**(tag): Retorna uma coleção de todos os elementos com o nome da tag especificada.

document.**querySelector**(selector): Retorna o primeiro elemento que corresponde ao seletor CSS especificado.

document.**querySelectorAll**(selector): Retorna uma lista de todos os elementos que correspondem ao seletor CSS especificado.

document.**createElement**(tag): Cria um novo elemento com o nome da tag especificada.

Continuação… Métodos disponíveis no documento:

document.**createTextNode**(text): Cria um novo nó de texto com o texto especificado.

document.**appendChild**(node): Adiciona um nó como o último filho de um nó pai.

document.**removeChild**(node): Remove um nó filho de um nó pai.

document.**replaceChild**(newNode, oldNode): Substitui um nó filho por um novo nó.

document.**write**(text): Escreve texto HTML diretamente no documento.

document.**open**(): Abre um documento stream para escrever.

document.**close**(): Fecha um documento stream aberto por document.open().

```
nome de tag
document.getElementsByTagName("p");

document.getElementById("main");
x.getElementsByTagName("p");

por nome de classe
document.getElementsByClassName("intro");

seletores CSS
document.querySelectorAll("p.intro");

Lista de objetos DOM Nível 1 e 3.
```

Buscar Elementos

É possível encontrar um elemento HTML pela id, tag, classe, seletores CSS, e por coleções de objetos HTML. Os seguintes elementos também estão acessíveis: anchors; body; embeds; forms; head; images; links; scripts; e title.

```
<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML =
"New text!";
</script>

<img id="myImage" src="smiley.gif">

<script>
document.getElementById("myImage").src =
"landscape.jpg";
</script>
```

Alterar o HTML

A maneira mais fácil de modificar o conteúdo de um elemento HTML é usando a propriedade innerHTML. Mas também é possível alterar um atributo, ou setar um conteúdo dinâmico.

```
Alterando o estilo
<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color
= "blue";
</script>

Usando eventos
<h1 id="id1">My Heading 1</h1>

<button type="button"
onclick="document.getElementById('id1').
style.color = 'red'">
Click Me!</button>
```

Alterar o CSS

O HTML DOM permite que o JavaScript altere o estilo dos elementos HTML. E também permite executar código quando ocorre um evento. Os eventos são gerados pelo navegador quando “coisas acontecem” aos elementos HTML: um elemento é clicado; a página foi carregada; os campos de entrada são alterados.

```
Exemplo
<h1 onclick="changeText(this)">Click on this
text!</h1>

<script>
function changeText(id) {
  id.innerHTML = "Ooops!";
}
</script>
```

Events

Um JavaScript pode ser executado quando ocorre um eventos HTML, como: quando um usuário clica com o mouse; uma página da web é carregada; uma imagem foi carregada; o mouse passa sobre um elemento; um campo de entrada é alterado; um formulário HTML é enviado; um usuário pressiona uma tecla.

Lista de Eventos

Mouse Events

onclick: Um usuário clica em um elemento.

oncontextmenu: Um usuário clica com o botão direito em um elemento.

ondblclick: Um usuário clica duas vezes em um elemento.

onmousedown: Um botão do mouse é pressionado sobre um elemento.

onmouseenter: O ponteiro do mouse se move para um elemento.

onmouseleave: O ponteiro do mouse sai de um elemento.

onmousemove: O ponteiro do mouse se move sobre um elemento.

onmouseout: O ponteiro do mouse sai de um elemento.

onmouseover: O ponteiro do mouse se move para um elemento.

onmouseup: Um botão do mouse é liberado sobre um elemento.

Interação e Animação

Touch Events

ontouchcancel: Um toque é interrompido.

ontouchend: Um dedo é removido de uma tela sensível ao toque.

ontouchmove: Um dedo é arrastado pela tela.

ontouchstart: Um dedo é colocado em uma tela sensível ao toque.

Others

resize: A visualização do documento é redimensionada.

reset: Um formulário é redefinido.

scroll: Uma barra de rolagem está sendo rolada.

Exemplos

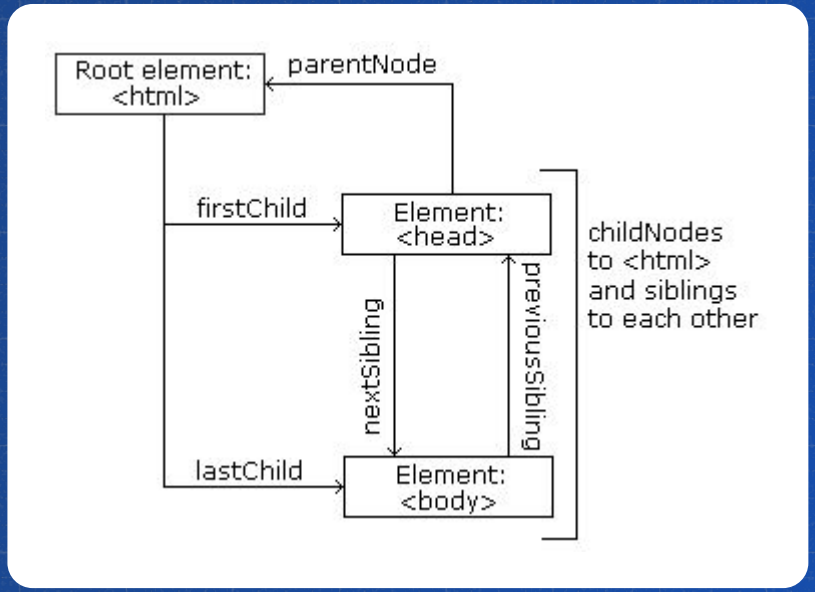
```
document.getElementById("myBtn").addEventLis
tener("click", displayDate);

element.addEventListener("click",
function(){ alert("Hello World!"); });
```

Exemplo no CodePen

Event Listener

O método addEventListener anexa um manipulador de eventos a um elemento sem substituir os manipuladores de eventos existentes, possibilitando o uso de muitos manipuladores no mesmo elemento.

O primeiro parâmetro é o tipo do evento (como " click" ou " mousedown" ou qualquer outro evento HTML DOM). O segundo parâmetro é a função que queremos chamar quando o evento ocorrer. O terceiro parâmetro é opcional, é um valor booleano que especifica se deve ser usado o evento bubbling ou a captura de eventos.



Navegação, nós, coleções e listas de nós.

Tudo em um documento HTML é um nó, e é possível navegar na árvore de nós usando relacionamentos de nós.

Um HTMLCollection é uma coleção de elementos de documento, e um NodeList é uma coleção de nós de documentos (nós de elementos, nós de atributos e nós de texto). E ambos são coleções (listas) semelhantes a arrays de nós (elementos) extraídos de um documento, e podem ser acessados por números de índice.

Interação e Animação

Troca de Tema Claro/Escuro:

```
<button id="themeBtn">Tema Escuro</button>

var body = document.body;
var themeBtn = document.getElementById('themeBtn');
themeBtn.addEventListener('click', function() {
    body.classList.toggle('dark-theme');
});
```

Interação e Animação

Casos de uso disponíveis no CodePen ou ChatGPT:

Galeria de Imagens com Controles.

Menu de Navegação Responsivo.

Barra de Progresso de Carregamento.

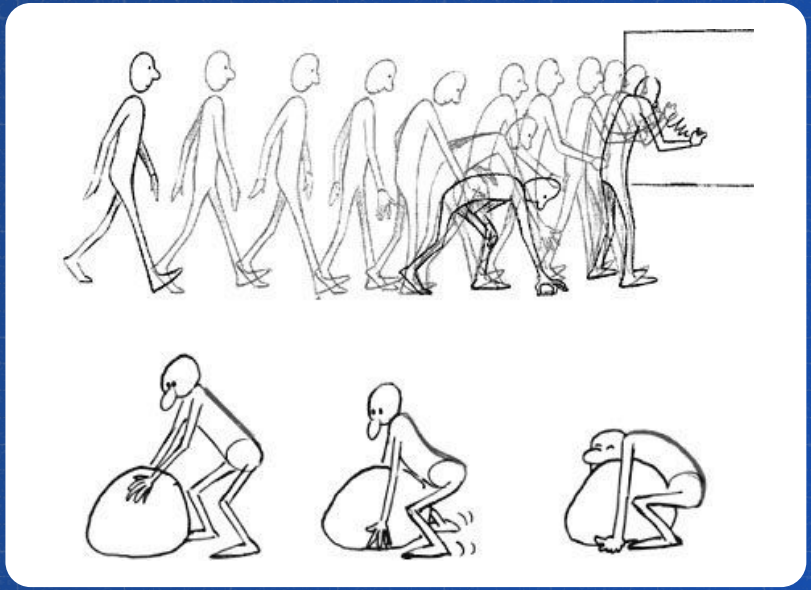
Pop-up Modal.

Contador de Cliques.

Autocomplete de Pesquisa.

Drag and Drop de Elementos.

...



Animação

A animação clássica envolve a criação de movimento por meio de uma sequência de imagens estáticas, chamadas **frames**. Fundamentos incluem a **taxa de frames** (frames por segundo, FPS), que determina a fluidez, e os **keyframes**, que são frames principais que definem as posições ou estados críticos. Técnicas como a **interpolação** preenchem os frames intermediários, garantindo transições suaves e naturais entre keyframes.

```
function myMove() {
  let id = null;
  const elem =
document.getElementById("animate");
  let pos = 0;
  clearInterval(id);
  id = setInterval(frame, 5);
  function frame() {
    if (pos == 350) {
      clearInterval(id);
    } else {
      pos++;
      elem.style.top = pos + 'px';
      elem.style.left = pos + 'px';
    }
  }
}
```

Animação no JS

As animações JavaScript são feitas programando mudanças graduais no estilo de um elemento. As mudanças são chamadas por um timer, e quando o intervalo do temporizador é pequeno, a animação parece contínua.

Exemplo no CodePen

```
@keyframes
animation-name
animation-duration
animation-delay
animation-iteration-count
animation-direction
animation-timing-function
animation-fill-mode
animation


transition
transition-delay
transition-duration
transition-property
transition-timing-function
```

Animação e Transição no CSS

Animação e Transição no CSS permitem criar efeitos visuais suaves e dinâmicos. As propriedades transition e animation facilitam mudanças de estilo e movimentos complexos. Compatíveis com todos os navegadores modernos, essas funcionalidades melhoram a experiência do usuário, tornando interfaces web mais atraentes e interativas, enquanto garantem um desempenho eficiente e uma implementação simplificada.

Exemplo no CodePen.

Obrigado!