Javascript

Programa de atração de formação de talentos
Bradesco / Visionaire

Aula 05/06 – DOM

Mark Joselli

mark.Joselli@pucpr.br

Placar

- Crie um contador de placar para partidas entre dois times:
  - Pode escolher quem fizer a pontuação máxima ganha (3,4..., 21);
  - Tem a opção de recomeçar o jogo

Guerra dos navegadores

- Até o 2015, havia muita diferença na forma que os navegadores renderizavam a página
  - A entrada de novos players, como o Chrome e o Firefox impulsionaram a indústria em busca de padrões
  - Isso forçou a a MS a adotar uma postura bem mais séria no navegador Edge, além de várias outras plataformas (como o C++);
  - O padrão é aberto e pode ser acesso em: *https://github.com/whatwg/html*
  - A incompatibilidade justificava a existência de APIs como o JQuery
- APIs como o AngularJS e React.JS usam Javascript intensamente. De fato, renderizam todo o site em uma única página!

Tag script

A tag script permite fazer a inclusão de scripts em uma página

```
<h1>Testando o alerta</h1>
<script>alert("Olá, Mundo!");</script>
```

O navegador executa a tag script assim que a encontra.

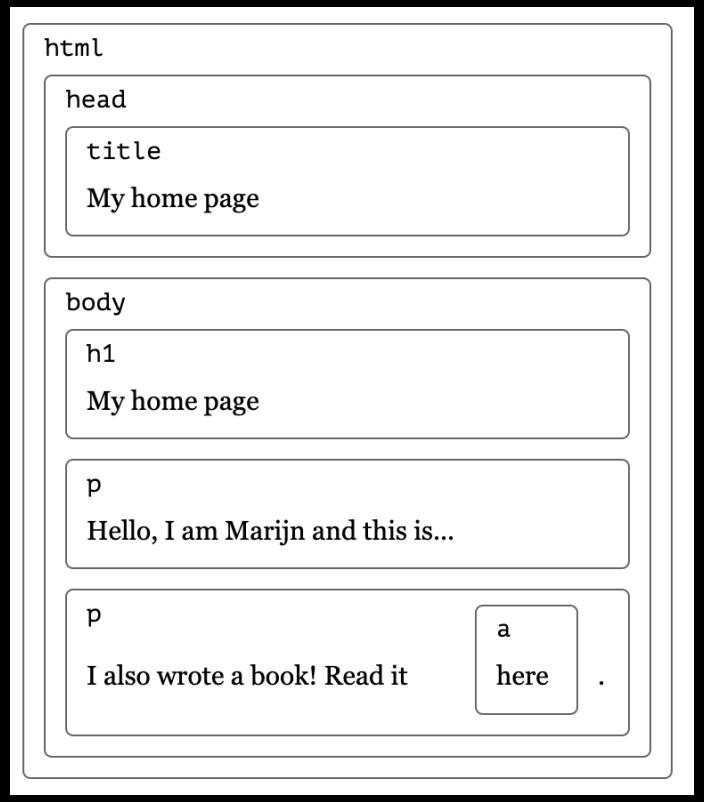
Obviamente, não é prático carregar todo script no html. Nesse caso, o atributo src também pode ser usado:

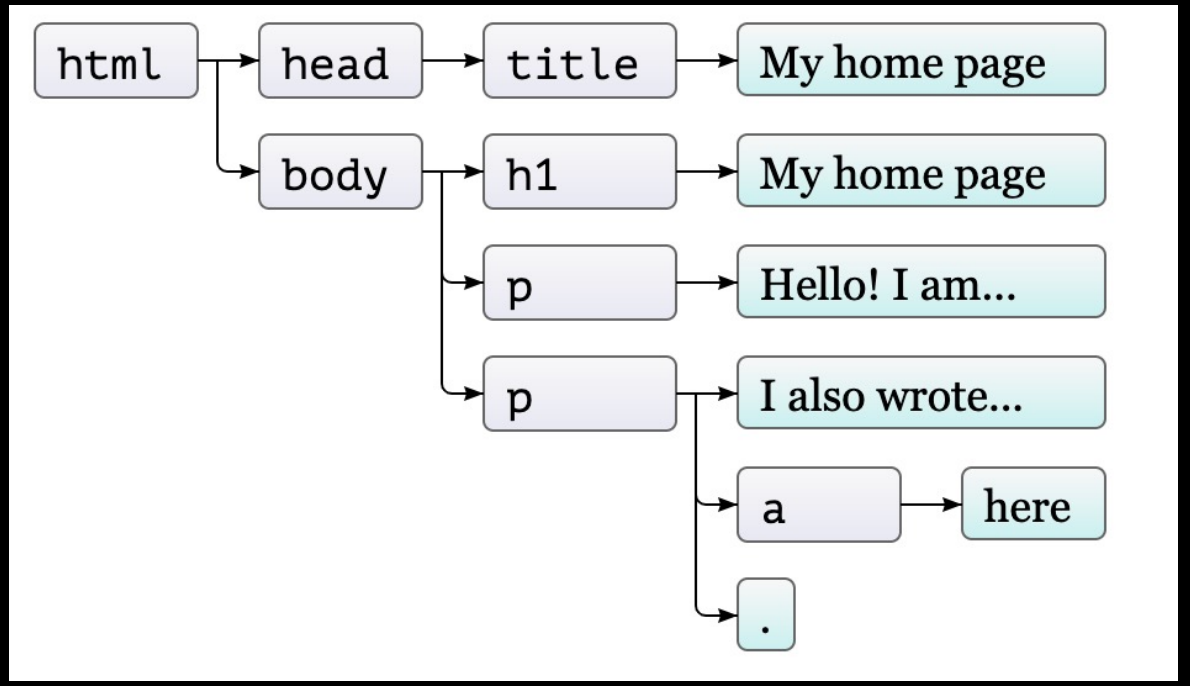
```
<script src="code/hello.js"></script>
```

Existe também a tag <noscript> usada para especificar o texto caso o javascript esteja desabilitado.

DOM

- Toda página HTML é representada na forma de uma árvore.
- Essa árvore é conhecida como Document Object Model (DOM)
- Cada item do HTML representa um nó da árvore, com nós filhos. O nó raiz é chamado de document





Document Object Model: Node

- Caso alteremos um nó, o navegador irá se redesenhar para exibir a mudança
- Podemos localizar nós por uma série de método get tais como: getElementById, getElementsByName, getElementsByClassName, getElementsByTagName
- Porém, hoje em dia, é mais comum utilizarmos o método querySelector pelo primeiro elemento que se encaixa em um seletor, ou querySelectorAll, para todos os elementos.

Alterando nós

- Existem 3 propriedades interessantes para alterar o conteúdo no interior de um nó:
  - textContent: Altera o conteúdo do nó, tratando-o como texto simples
  - innerText: Altera somente o conteúdo de texto visível
  - innerHtml: Altera o conteúdo do nó, tratando-o como html.

- Em todos os casos, qualquer conteúdo html no interior do nó será destruído.

NodeList

- Métodos que retornam grupos de nós retornam um objeto do tipo NodeList, que apesar de parecer, não é um array
  - Não use o comando for in sobre ele, porém o for...of pode ser usado
  - Em navegadores mais modernos, já é possível utilizar o comando forEach, entries(), values() e keys() ou converte-los em um array com o comando Array.from, mas nem todos os navegadores suportam.

Exemplo

```
<!doctype html>
<html>
<title></title>
<body>
  <p id="relogio”></p>
</body>
</html>
```

```
function atualizaRelogio() {
    document
        .querySelector("#relogio")
        .textContent = new Date().toLocaleTimeString();
}

setInterval(atualizaRelogio, 1000);
```

Eventos

- **oninput**: quando um elemento input tem seu valor modificado
- **onclick, ondblclick**: quando ocorre o click ou duplo clique com o mouse
- **onmousemove**: quando o mouse se movimenta
- **onmousedown e onmouseup**: pressionar do botão do mouse
- **onkeypress, onkeydown e onkeyup**: pressionar de teclas
- **onblur e onfocus**: Perda e ganho de foco
- **onchange**: quando um input, select ou textarea tem seu valor alterado
- **onload e onunload**: quando a página é carregada / fechada
- **onsubmit**: disparado antes de submeter o formulário (útil para realizar validações)

Inputs

- Representam objetos criados para interação com usuário
- Possuem uma série de tipos, sendo os mais comuns:
  - **Botão**: button, submit
  - **Seleção**: checkbox, radio
  - **Texto**: text, email, password, url e search
- Possuem propriedades convenientes como *value*
- São geralmente locais adequados para se registrar eventos

Recursividade

Por se tratar de uma árvore, muitas vezes precisamos escrever funções recursivas para lidar com o DOM

Por exemplo, esta função busca um valor em um nó de textos qualquer

```
function contemOTexto(no, valor) {
    //Se for um nó de texto, basta o valor
    if (no.nodeType === Node.TEXT_NODE) {
        return no.nodeValue.includes(valor);
    }
    //Se for outra coisa, busca nos elementos filhos
    for (let filho of no.childNodes) {
        if (no.nodeType === Node.ELEMENT_NODE &&
            contemOTexto(filho, valor))
        {
            return true;
        }
    }
    return false;
}

console.log(contemOTexto(document.body, "três"));
```

Inserindo elementos no DOM

- É possível remover qualquer nó da árvore chamando o método remove de seu pai
- Também é possível adicionar um filho em um nó com os métodos:
  - appendChild: Para inserção ao final
  - insertsBefore: Para inserção antes de um nó específico.
- Um nó só pode existir em uma única posição da árvore. Chamar os métodos acima em um nó existente o removerá da sua posição atual na árvore.

Exemplo

- Teste o seguinte programa no CodePen

Html

```
<p>Um</p>
<p>Dois</p>
<p>Três</p>

<input value="Mudar posições"
       type="button"
       onClick="mudarPosicoes()"
/>
```

JS

```
function mudarPosicoes() {
   let ps = document.body
      .getElementsByTagName("p");
   document.body.insertBefore(ps[2], ps[0]);
}
```

Criando um nó de textos

- Criar um nó textual é bastante simples, basta chamar a função createTextNode
- Por exemplo, vamos escrever um programa para substituir todas de uma página por seu texto alternativo

Atributos

- Alguns atributos padrão (como o alt) existem em qualquer nó.
- Entretanto, você pode testar por outros atributos em um nó através do método getAttribute.
- Isso permitirá que você inclua no elemento atributos com qualquer nome, inclusive criados por você.
- Por exemplo, altere o exemplo anterior trocando o atributo alt pelo atributo texto. Em seguida, use o getAttribute para fazer a substituição por esse valor.

Criando nós mais complexos

- Você pode criar nós mais complexos através do comando document.createElement(type)
- O parâmetro type permite escolher o tipo do elemento.
- Em seguida, você pode chamar os métodos:
  - setAttributes para adicionar atributos adicionais ou
  - appendChild para incluir elementos filhos.
- Não há uma forma fácil de fazer isso em “uma só tacada”. Mas nada que um pouco de Javascript não resolva. ☺

Trabalho: Quiz em grupos 2/3

- Realize um página de quiz, que tenha:
  - Um contexto;
  - Pelo menos 5 perguntas com opção de ter uma imagem e pelo menos 4 opções;
  - Mostre a opção certa e tenha um sistema de pontuação
  - A ordem das perguntas e das opções devem ser randomizados
  - Uso de arrays, objects, inclusão/exclusão de itens no DOM
  - Data de entrega: Antes da aula do dia 28/10
- Desafios extras:
  - Interface bonita – CSS;
  - Um placar com os melhores pontuações;
  - Pelo menos uma pergunta com uso de áudio;
  - Timer de tempo máximo para responder;

Calendário

- Aula 7 - Trabalho QUIZ
- Aula 8 – Apresentação QUIZ
- Aula 9 – POO
- Aula 10 – Async 1
- Aula 11 – Async 2
- Aula 12 – TDD – Trabalho 2
- Aula 13 – Expressões regulares
- Aula 14 – Apresentação trabalho 2