

# Aula 8

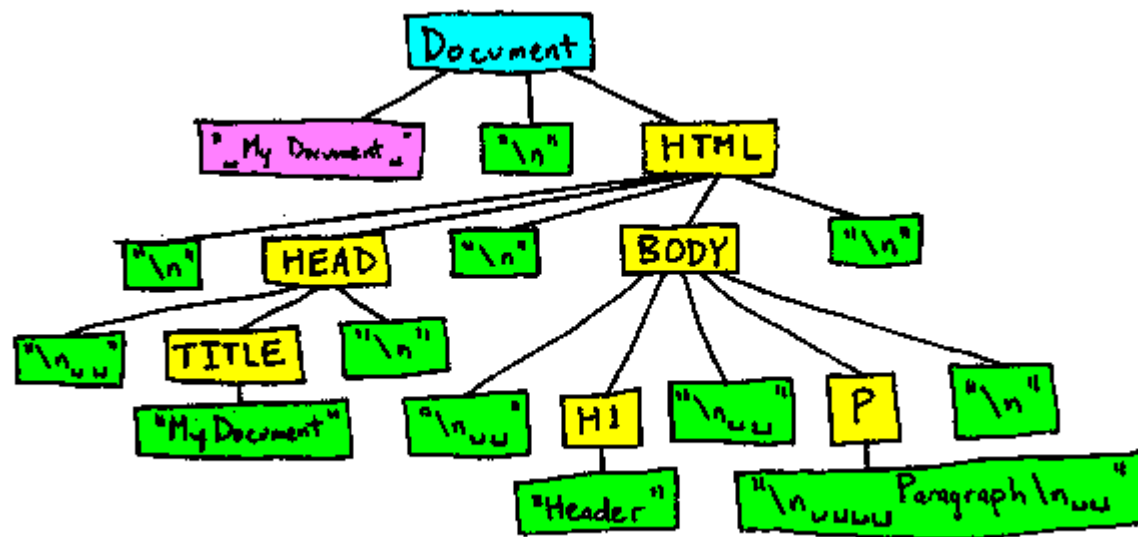
Document Object Model / Eventos  
Programação IV

Prof. Sandino Jardim  
CC-UFMT-CUA



# Document Object Model (DOM)

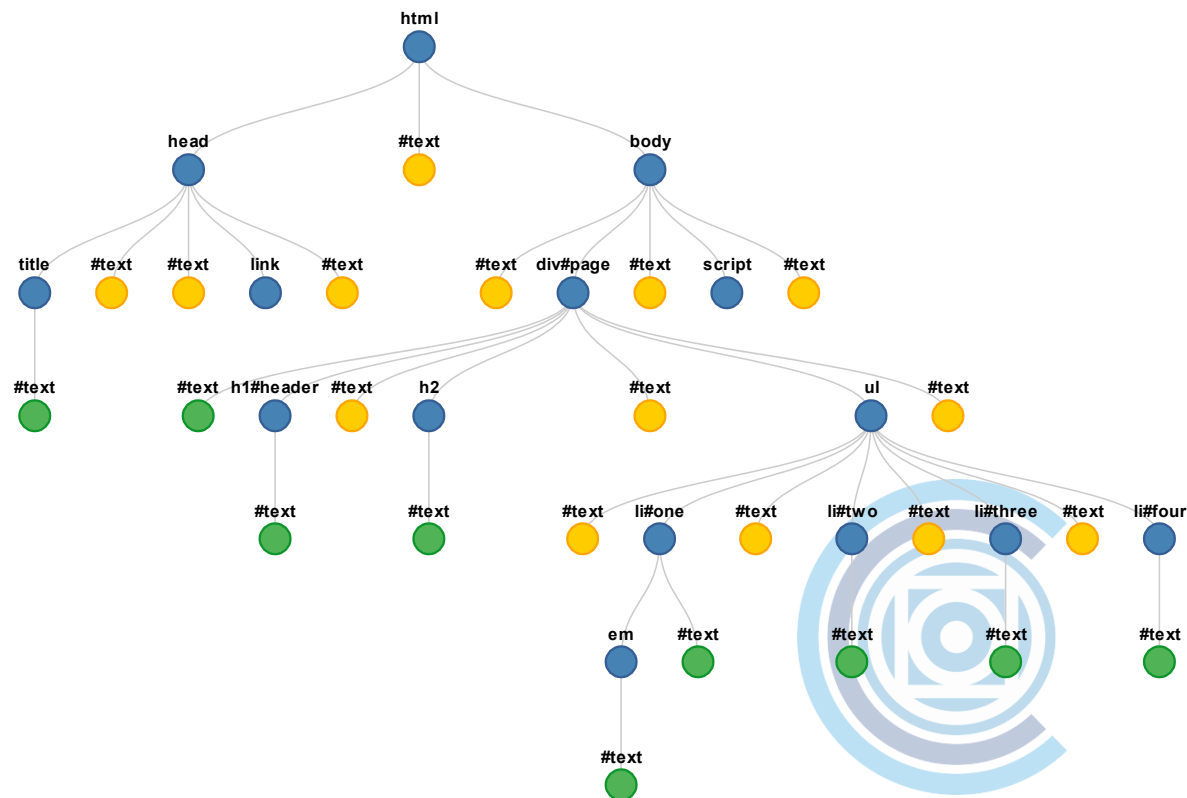
- Especifica como os browsers devem criar um modelo de página HTML
- E como JS pode acessar e atualizar o conteúdo de uma página web



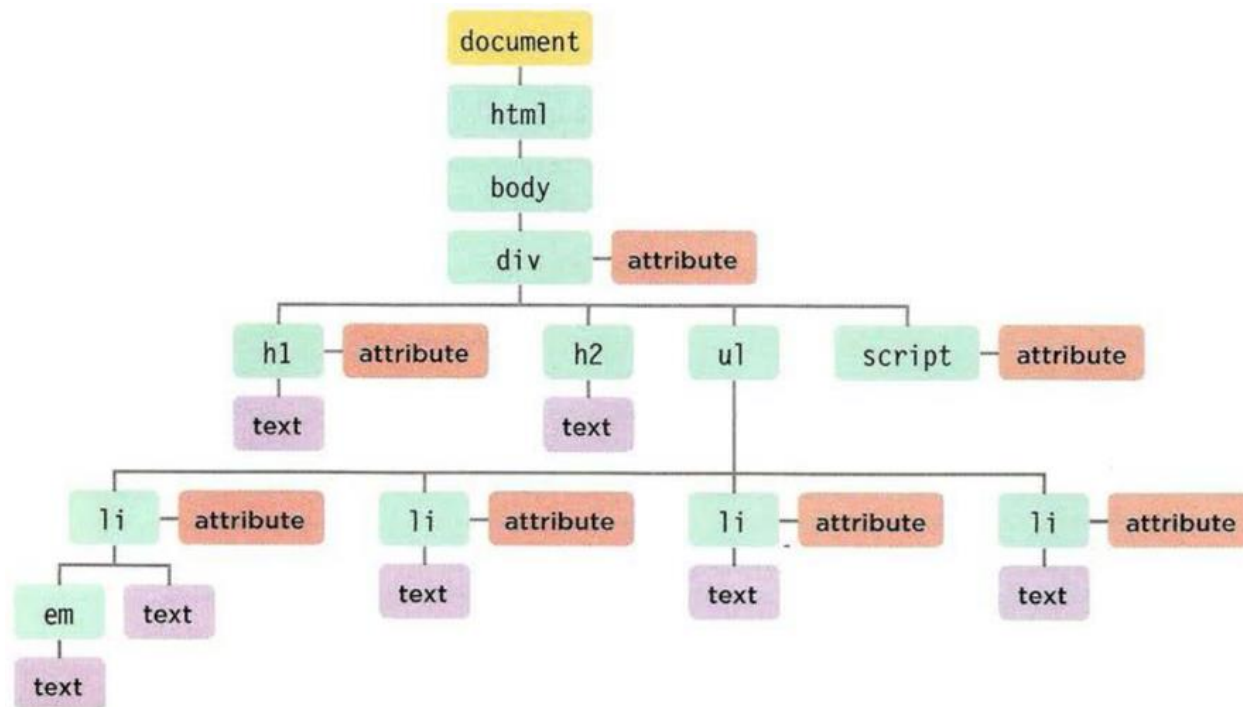
# Árvore DOM

- Modelo que representa a estrutura HTML de uma página carregada

```
<html>
<head>
  <title>Document Object Model - Example</title>
  <link rel="stylesheet" href="css/c05.css">
</head>
<body>
  <div id="page">
    <h1 id="header">List</h1>
    <h2>Buy groceries</h2>
    <ul>
      <li id="one"><em>fresh</em> figs</li>
      <li id="two">pine nuts</li>
      <li id="three">honey</li>
      <li id="four">balsamic vinegar</li></ul>
  </div>
  <script src="js/example.js"></script>
</body>
</html>
```



# Árvore DOM



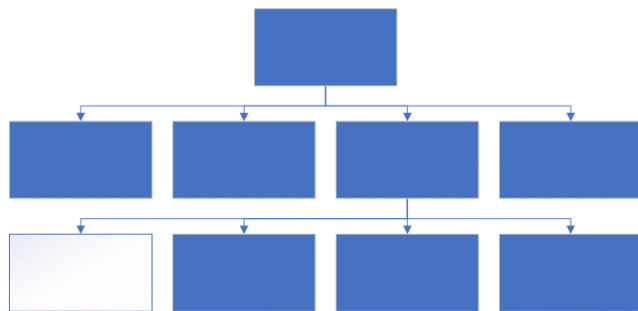
- Documento
- Elemento
- Atributo
- Texto



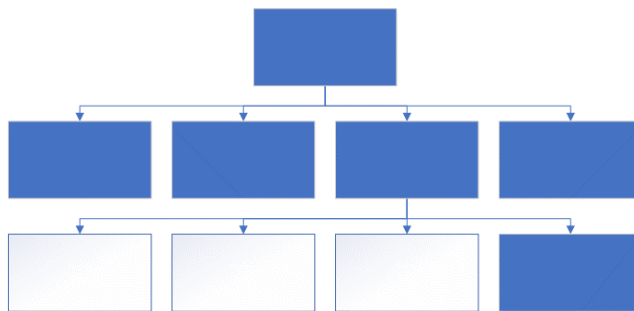
# Manipulando Árvore DOM

- Passo 1: Acesse os elementos

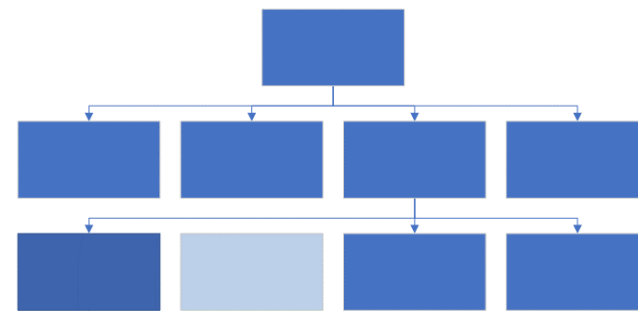
Selecione um elemento individualmente



Selecione múltiplos elementos



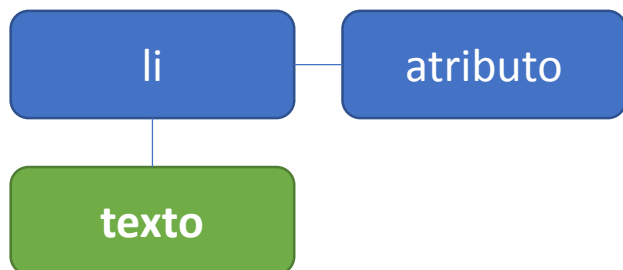
Navegue entre elementos



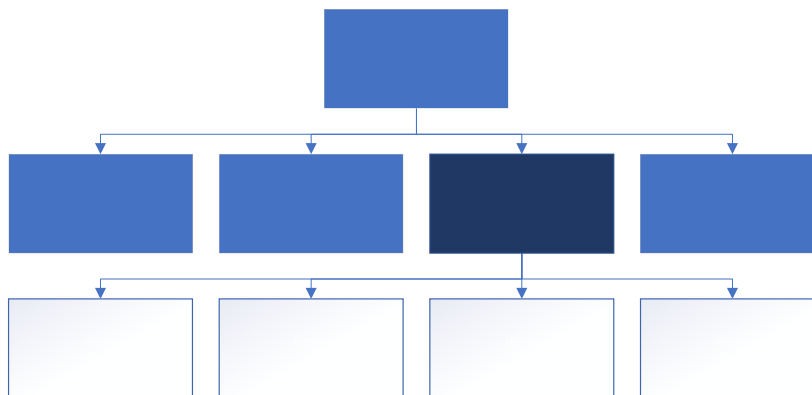
# Manipulando Árvore DOM

- Passo 2: Trabalhe com os elementos

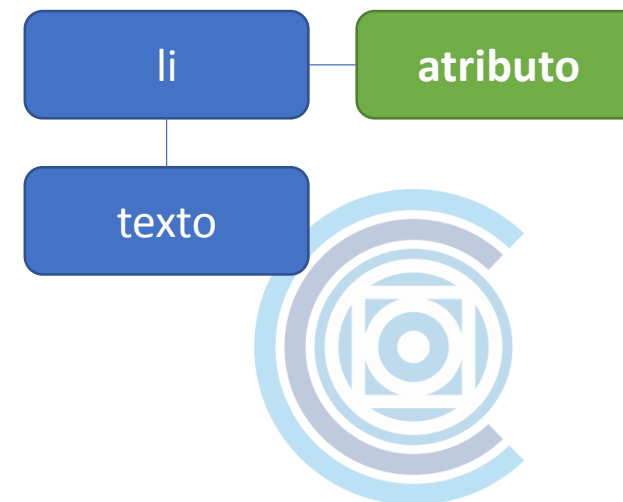
Acesse/atualize textos



Trabalhe com o conteúdo HTML



Acesse/atualize atributos



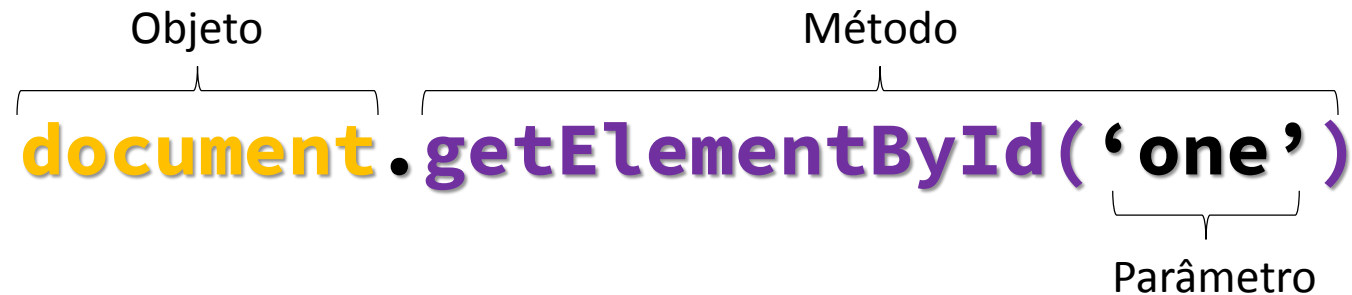
# Acessando elementos

- Elemento da árvore podem ser armazenados em variáveis
- 'DOM queries' podem retornar um elemento ou uma NodeList deles

Objeto                      Método

**document**.getElementById(**'one'**)

Parâmetro



# Acessando elementos individualmente

## getElementById()

Forma mais rápida e eficiente de acessar elementos de id única

## querySelector()

Forma mais flexível de acessar elementos, pois parâmetro aceita seletores CSS

```
<h1 id="header">List King</h1>
<h2>Buy groceries</h2>
<ul>
  <li id="one" class="hot"><em>fresh</em> figs</li>
  <li id="two" class="hot">pine nuts</li>
  <li id="three" class="hot">honey</li>
  <li id="four">apple vinegar</li>
</ul>
```

HTML

```
var el = document.getElementById('one');
el.className = 'cool';
```

get-element-by-id.js

JAVASCRIPT



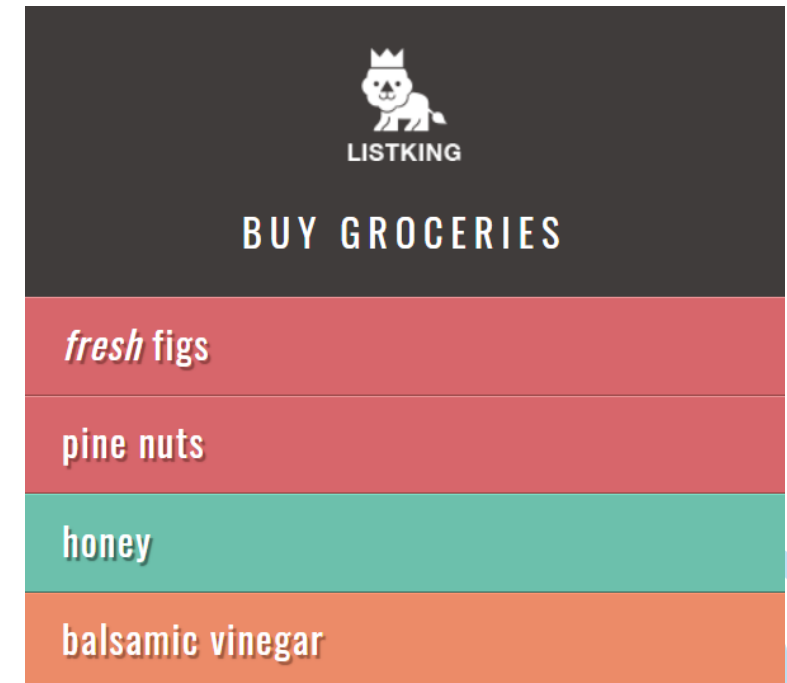


# Acessando elementos por classe

## JAVASCRIPT

```
var elements =  
document.getElementsByClassName('hot');  
  
if (elements.length > 2)  
{  
  var el = elements[2];  
  el.className = 'cool';  
}
```

get-element-by-class-name.js

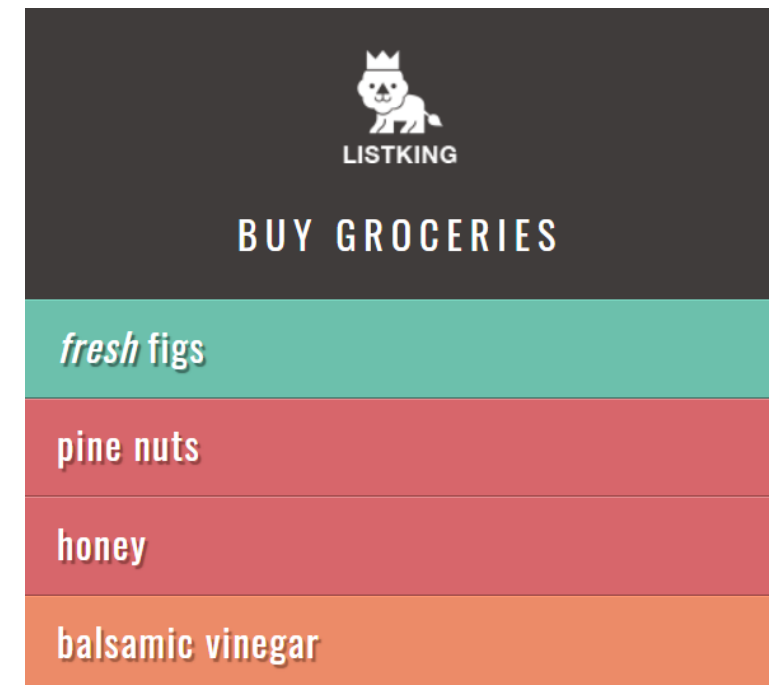


# Acessando elementos por tag

```
var elements = document.getElementsByTagName('li');  
  
if (elements.length > 0) {  
  var el = elements[0];  
  el.className = 'cool';  
}
```

get-element-by-tag-name.js

JAVASCRIPT

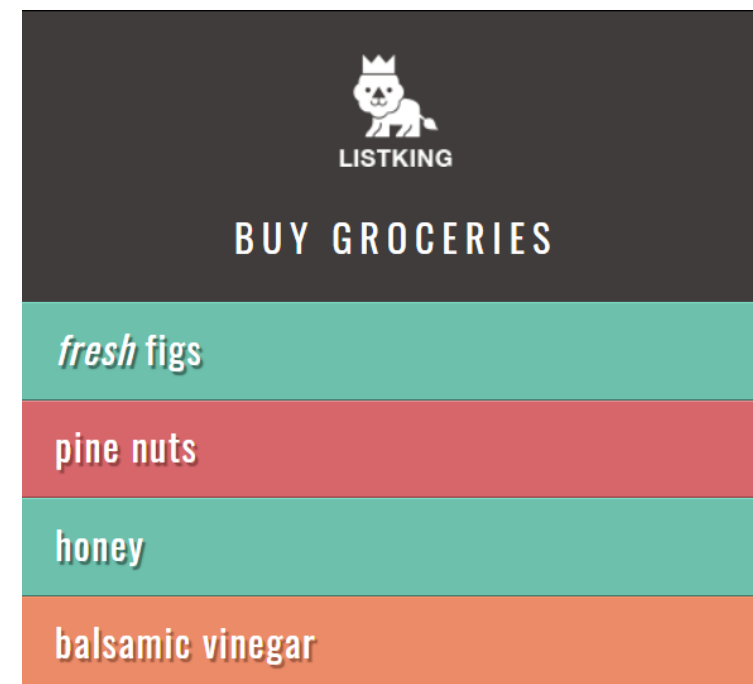


# Acessando elementos por seletor CSS

```
var el = document.querySelector('li.hot');  
el.className = 'cool';  
  
var els = document.querySelectorAll('li.hot');  
els[1].className = 'cool';
```

query-selector.js

JAVASCRIPT

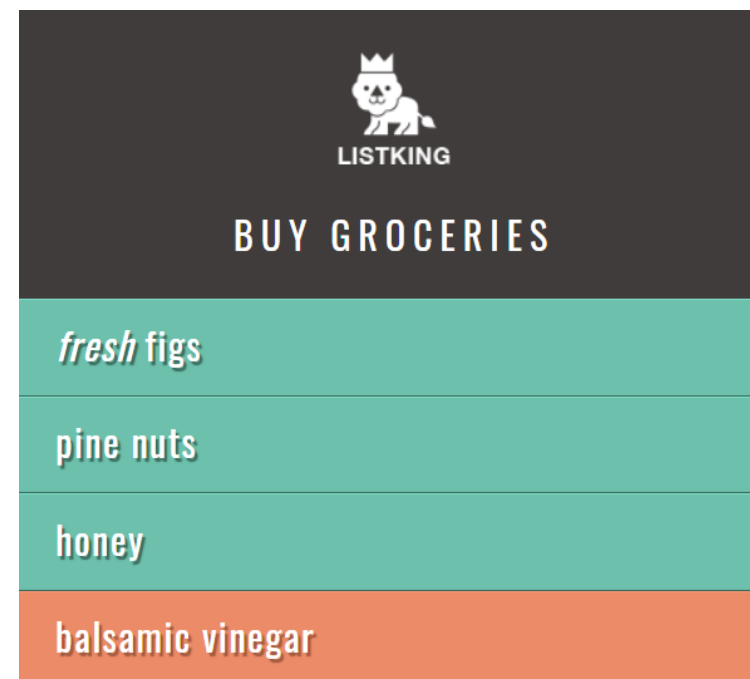


# Iterando sobre uma NodeList

```
var hotItems = document.querySelectorAll('li.hot');  
if (hotItems.length > 0) {  
  
    for (var i = 0; i < hotItems.length; i++) {  
        hotItems[i].className = 'cool';  
    }  
  
}
```

node-list.js

JAVASCRIPT



# Navegando entre elementos DOM

```
var startItem = document.getElementById('two');  
var prevItem = startItem.previousSibling;  
var nextItem = startItem.nextSibling;
```

```
prevItem.className = 'complete';  
nextItem.className = 'cool';
```

sibling.js

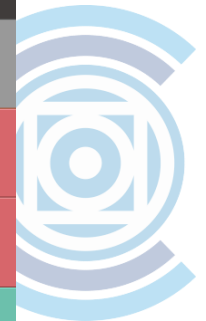
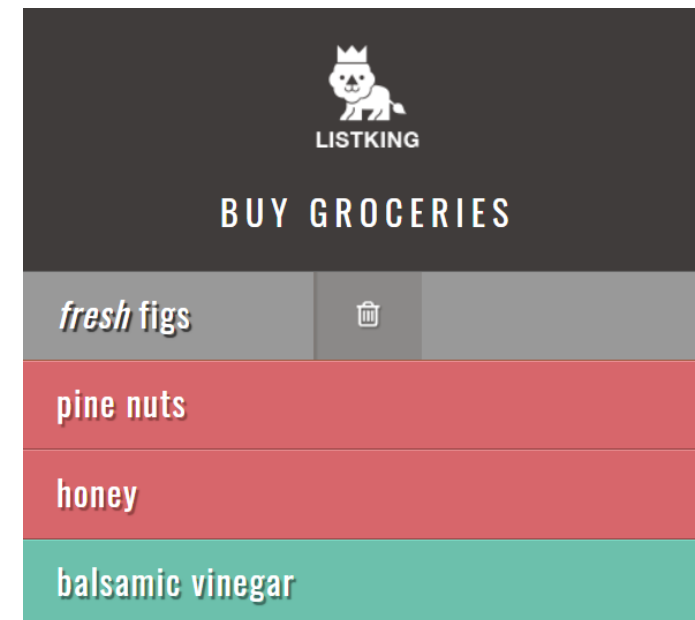
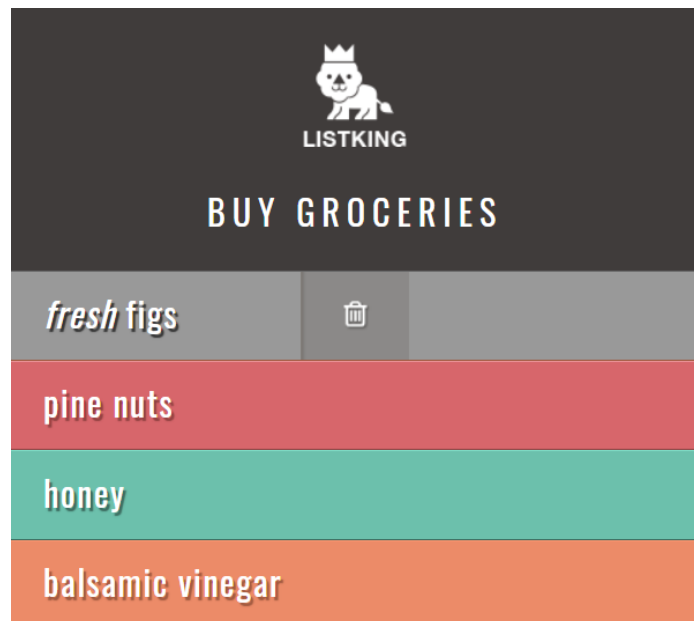
JAVASCRIPT

```
var startItem = document.getElementsByTagName('ul')[0];  
var firstItem = startItem.firstChild;  
var lastItem = startItem.lastChild;
```

```
firstItem.className = 'complete';  
lastItem.className = 'cool';
```

child.js

JAVASCRIPT

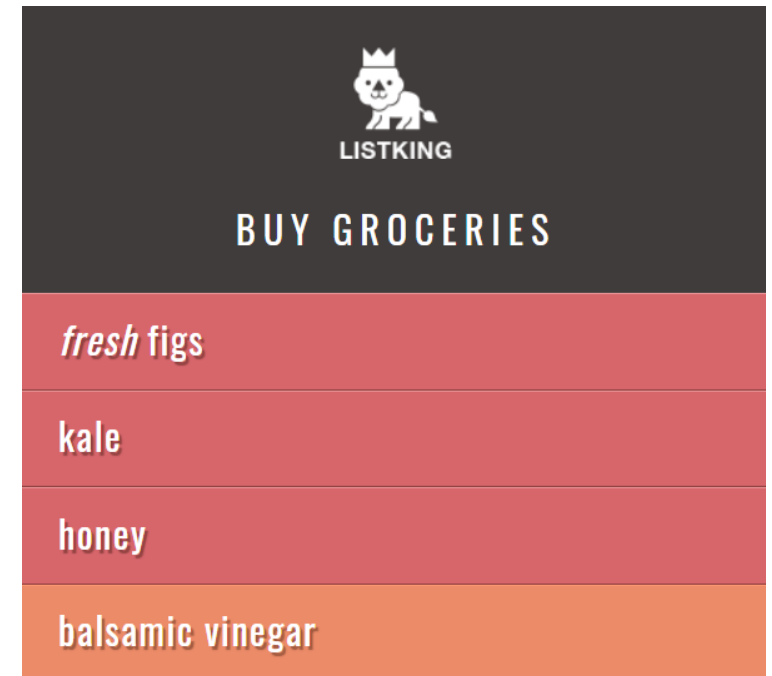


# Acessando e alterando texto

```
var itemTwo = document.getElementById('two');  
  
var elText  = itemTwo.firstChild.nodeValue;  
  
elText = elText.replace('pine nuts', 'kale');  
  
itemTwo.firstChild.nodeValue = elText;
```

node-value.js

JAVASCRIPT

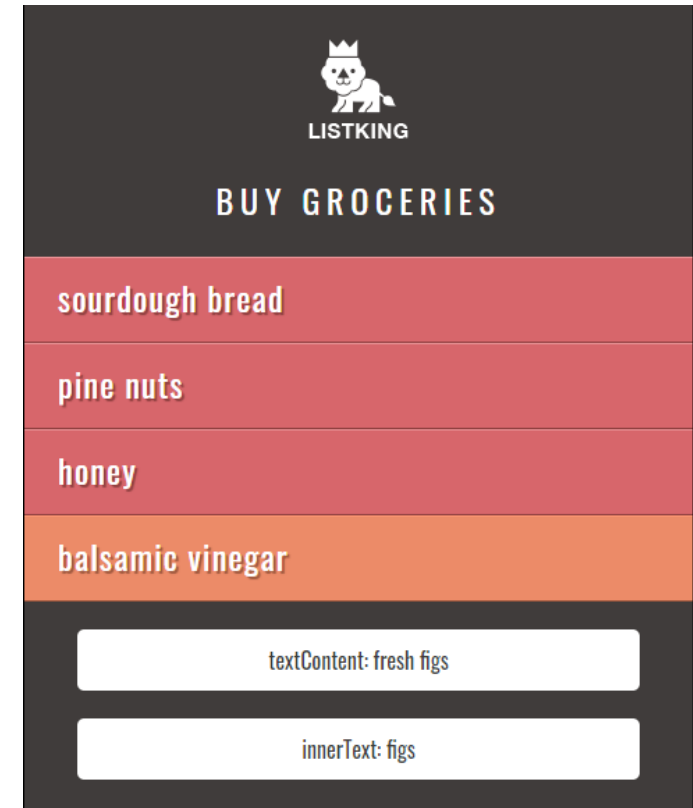


# Acessando somente texto

```
var firstItem = document.getElementById('one');  
var showTextContent = firstItem.textContent;  
var showInnerText = firstItem.innerText;  
  
var msg = '<p>textContent: ' + showTextContent + '</p>';  
      msg += '<p>innerText: ' + showInnerText + '</p>';  
var el = document.getElementById('scriptResults');  
el.innerHTML = msg;  
  
firstItem.textContent = 'sourdough bread';
```

inner-text-and-text-content.js

JAVASCRIPT



# Atualizando texto e marcação

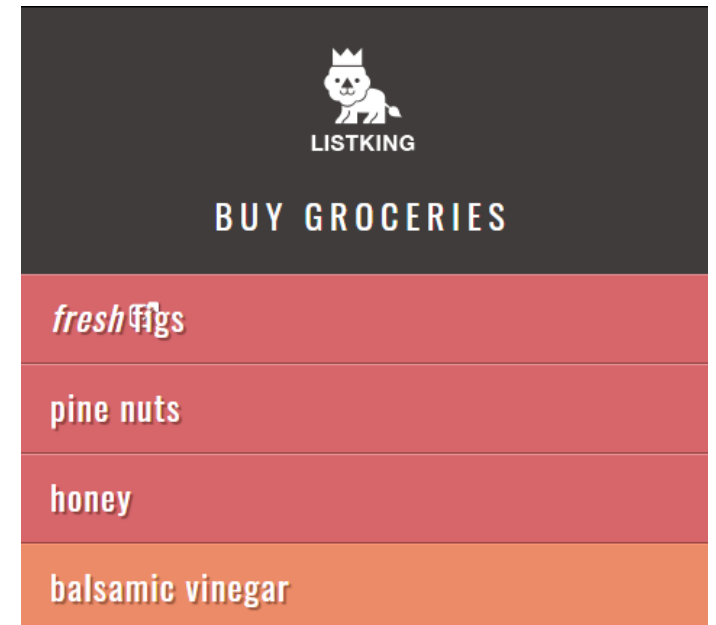
```
// Store the first list item in a variable
var firstItem = document.getElementById('one');

// Get the content of the first list item
var itemContent = firstItem.innerHTML;

// Update the content of the first list item so it is a
link
firstItem.innerHTML = '<a href=\"http://example.org\">'
+ itemContent + '</a>';
```

inner-html.js

JAVASCRIPT



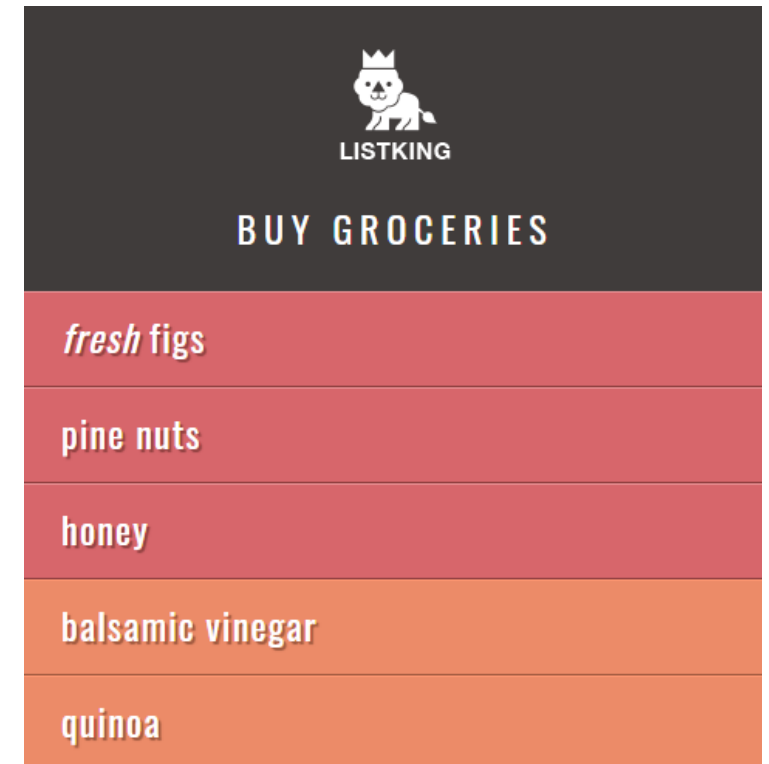


# Adicionando elemento

```
var newEl = document.createElement('li');  
  
var newText = document.createTextNode('quinoa');  
  
newEl.appendChild(newText);  
  
var position = document.getElementsByTagName('ul')[0];  
  
position.appendChild(newEl);
```

add-element.js

JAVASCRIPT



# Alterando atributos

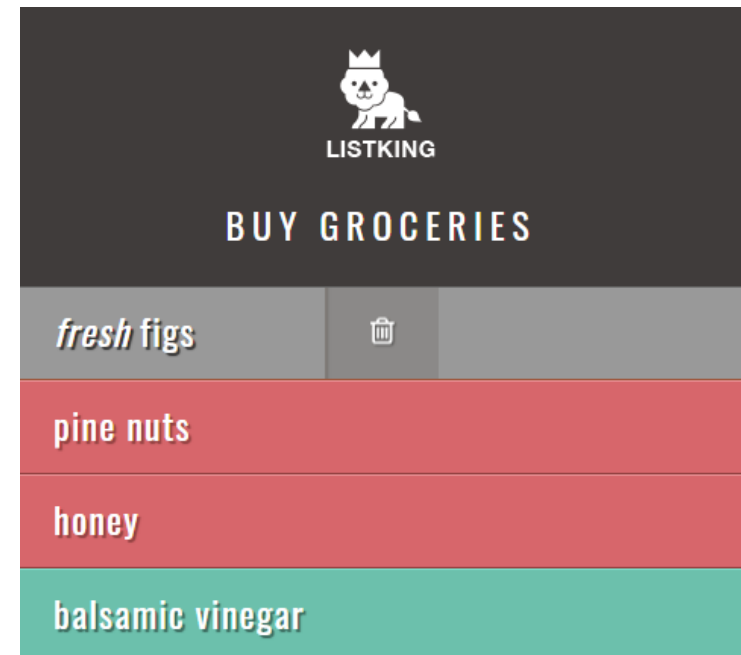
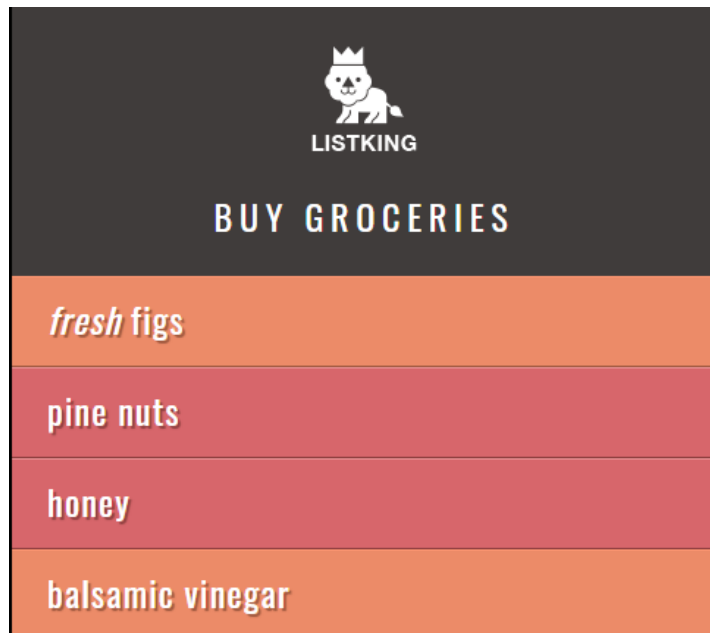
```
var firstItem = document.getElementById('one');  
if (firstItem.hasAttribute('class')) {  
    firstItem.removeAttribute('class');  
}
```

JAVASCRIPT

```
var firstItem = document.getElementById('one');  
firstItem.className = 'complete';
```

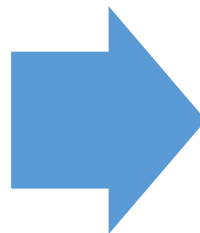
```
var fourthItem = document.getElementsByTagName('li').item(3);  
fourthItem.setAttribute('class', 'cool');
```

JAVASCRIPT



# Lidando com Eventos

Enquanto você navega,  
seu browser registra  
diferentes tipos de  
eventos



Scripts geralmente  
respondem a eles  
atualizando o conteúdo da  
página (via DOM),  
tornando-a mais interativa



# Diferentes tipos de eventos

Eventos de UI	Eventos de teclado	Eventos de mouse
load	keydown	click
error	keyup	dblclick
resize	keypress	mousedown
scroll		mouseup
Eventos de Foco	Eventos de formulário	Eventos de mutação
focus	input	DOMSubtreeModified
focusin	change	DOMNodeInserted
blur	submit	DOMNodeRemoved
focusout	select	



# Dinâmica do tratamento de eventos

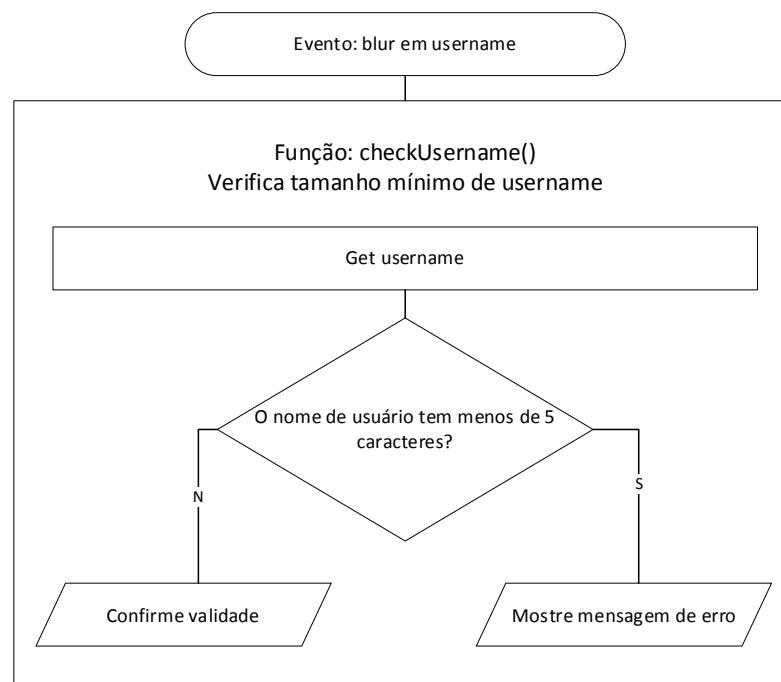
Selecione o nó DOM  
que deseja  
monitorar



Indique qual evento  
sobre o nó ativar o  
script



Defina o código que  
deseja ser executado  
após o evento



# Formas de lidar com eventos

## Manipuladores de Eventos HTML

- Prática não recomendada, mas muito vista em códigos mais antigos
- Mistura scripts JS com código HTML

## Manipuladores tradicionais de eventos DOM

- Apresentados na especificação original do DOM
- Suportados na maioria dos browsers, limitado a uma função por evento

## ‘Event listeners’ DOM nível 2

- Permite que um evento acione múltiplas funções
- Diminui a probabilidade de conflitos



# Manipuladores de Eventos DOM

- Sintaxe:

**element**.**onevent** = *functionName*;

Elemento DOM alvo    Evento associado ao elemento    Nome da função a ser invocada (sem parênteses)



# Manipuladores de Eventos DOM

```
<div id="page">
  <h1>List King</h1>
  <h2>New Account</h2>
  <form method="post" action="http://www.example.org/register">

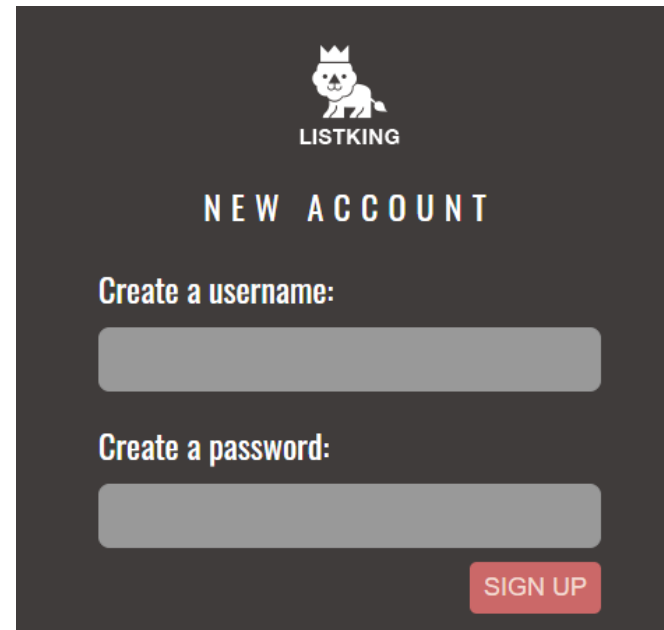
    <label for="username">Create a username: </label>
    <input type="text" id="username" />
    <div id="feedback"></div>

    <label for="password">Create a password: </label>
    <input type="password" id="password" />

    <input type="submit" value="sign up" />

  </form>
</div>
<script src="js/event-handler.js"></script>
```

HTML



```
function checkUsername() {
  var elMsg = document.getElementById('feedback');
  if (this.value.length < 5) {
    elMsg.textContent = 'Username must be 5 characters or more';
  } else { elMsg.textContent = ''; }
}
var elUsername = document.getElementById('username');
elUsername.onblur = checkUsername;
```

JAVASCRIPT

event-handler.js





# Event Listeners

- Sintaxe:

**element.addEventListener('event', function [, Boolean]);**

Diagram illustrating the syntax of the `addEventListener` method:

- element**: Elemento DOM alvo
- add**: Método
- EventListener**: Método
- ('event')**: Evento esperado
- function**: Função
- [, Boolean]**: Fluxo de evento

```
function checkUsername() {  
    var elMsg = document.getElementById('feedback');  
    if (this.value.length < 5) {  
        elMsg.textContent = 'Username must be 5 characters or more';  
    } else {  
        elMsg.textContent = '';  
    }  
}  
  
var elUsername = document.getElementById('username');  
  
elUsername.addEventListener('blur', checkUsername, false);
```

JAVASCRIPT

event-listener.js



# Event Listeners

- Exemplo de uso

JAVASCRIPT

```
var elUsername = document.getElementById('username');
var elMsg       = document.getElementById('feedback');

function checkUsername(minLength) {
  if (elUsername.value.length < minLength) {
    elMsg.innerHTML = 'Username must be ' + minLength + ' characters or more';
  } else {
    elMsg.innerHTML = '';
  }
}

elUsername.addEventListener('blur', function() {
  checkUsername(5);
}, false);
```



event-listener-with-parameters.js

# Event listener com objeto 'event'

JAVASCRIPT

```
function checkLength(e, minLength) {  
  var el, elMsg;  
  if (!e) {  
    e = window.event;  
  }  
  el = e.target || e.srcElement;  
  elMsg = el.nextSibling;  
  
  if (el.value.length < minLength) {  
    elMsg.innerHTML = 'Username must be ' + minLength + ' characters or more';  
  } else {  
    elMsg.innerHTML = '';  
  }  
}
```

event-listener-with-event-object.js



# Event listener com objeto 'event' (cont.)

```
var elUsername = document.getElementById('username');  
if (elUsername.addEventListener) {  
    elUsername.addEventListener('blur', function(e) {  
        checkLength(e, 5);  
    }, false);  
} else {  
    elUsername.attachEvent('onblur', function(e) {  
        checkLength(e, 5);  
    });  
}
```



JAVASCRIPT

# Delegando eventos

- Ideal quando elementos são manipulados de maneira semelhante
- Ao invés de atribuir um manipulador para cada um, coloca-se um único manipulador no ancestral em comum destes

event-delegation.js



# Exemplos de diferentes tipos de eventos

- Focus/blur
- Form
- Keypress
- Load
- Mutation
- Position
- Click

