Introduction to JavaScript – Part Three

Website Development 2

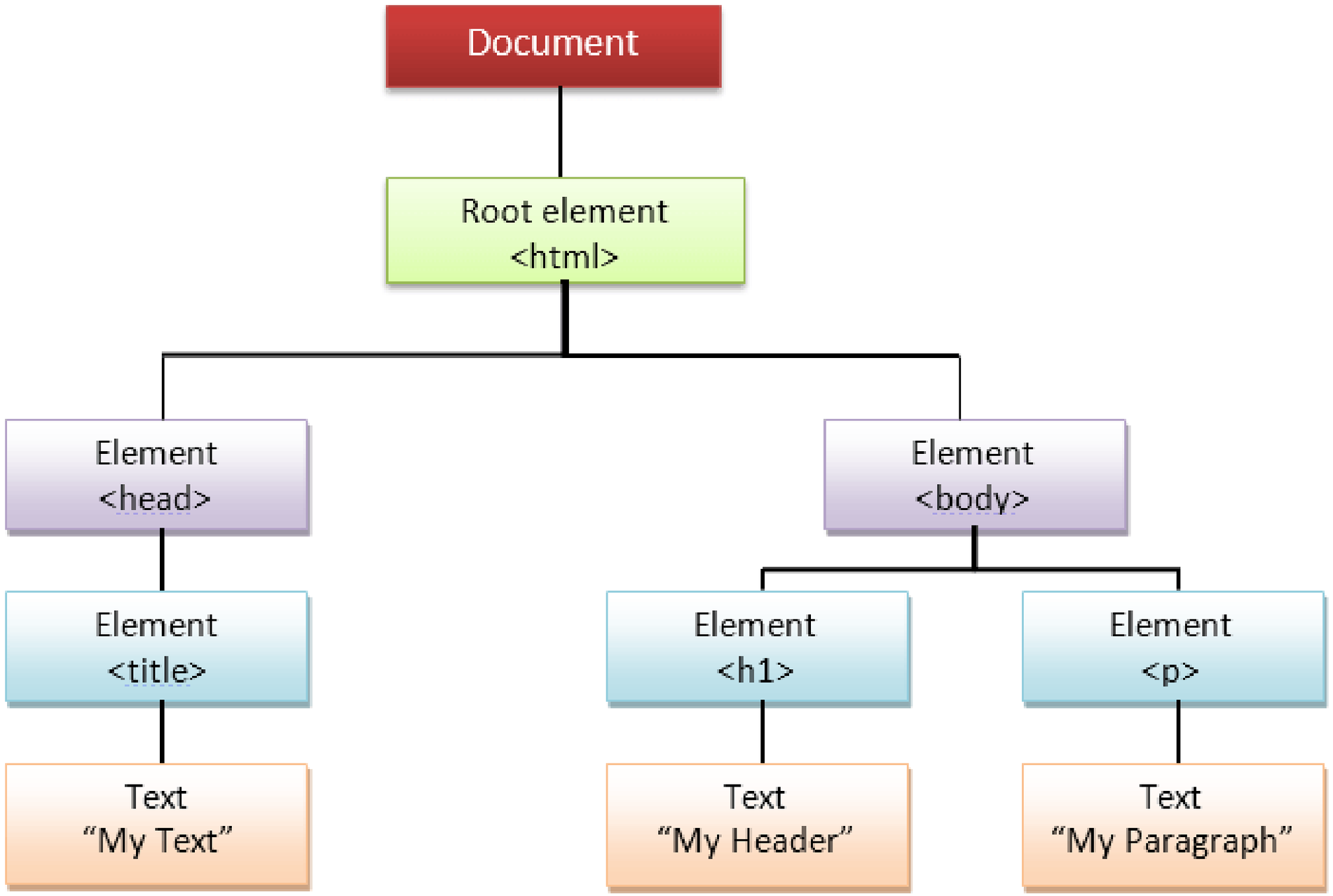
Lecture Outline

- Document Object Model
- Accessing Elements
- Accessing Form Elements
- Events

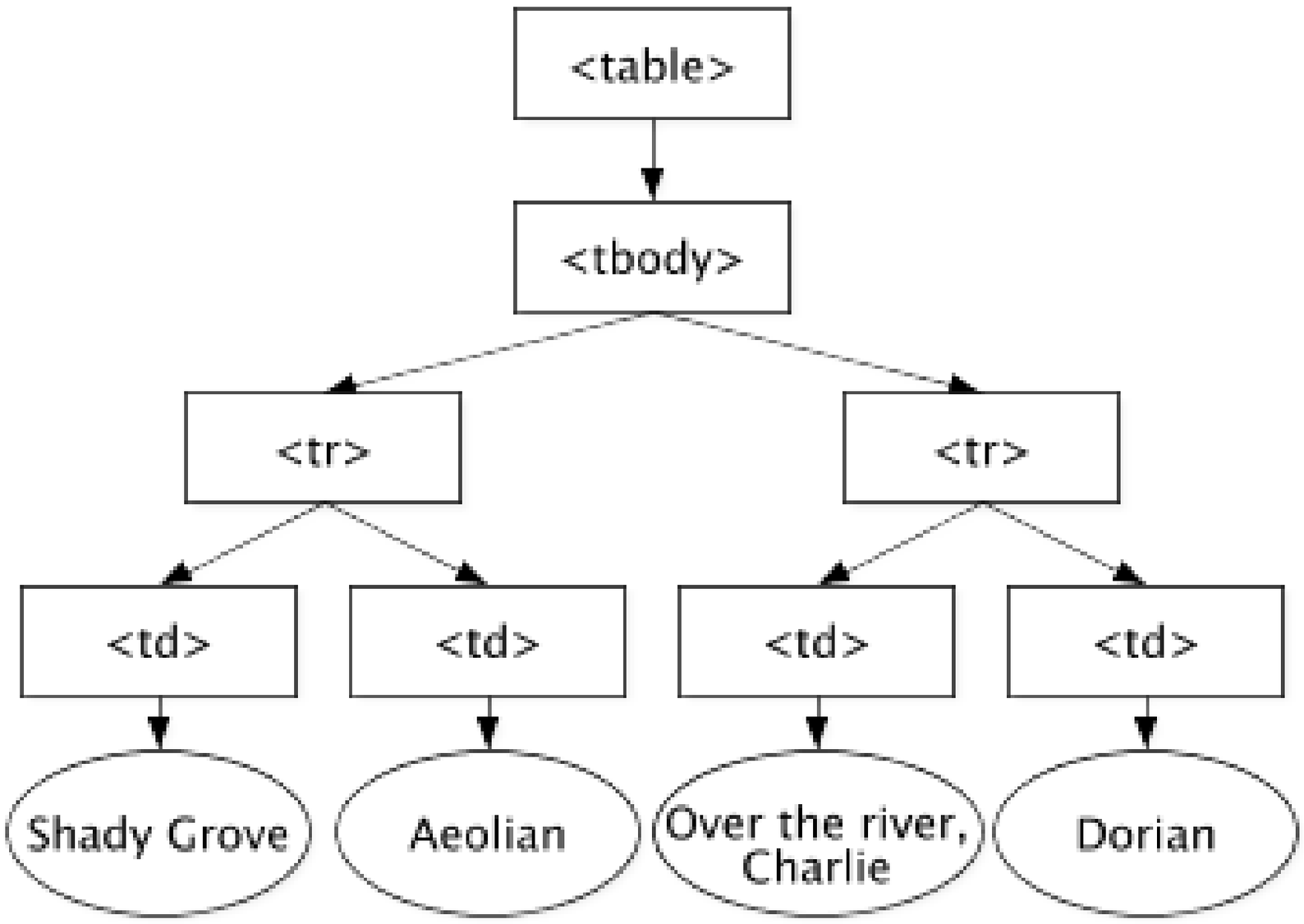
Document Object Model

- JavaScript can access all the elements in a webpage making use of Document Object Model (DOM). In fact, the web browser creates a DOM of the webpage when the page is loaded. The DOM model is created as a tree of objects.
- The DOM specifies how browsers should create a model of an HTML page and how JavaScript can access and update the contents of a web page while it is in the browser window.

Document Object Model



Document Object Model (Table)



Document Object Model

The document node

- Every element, attribute, and piece of text in the HTML is represented by its own **DOM node**.
- At the top of the tree a **document node** is added; it represents the entire page.

Element nodes

- HTML elements describe the structure of an HTML page (For example, the <h1> elements describe what parts are Large heading data; while <td> elements describe what parts are table data).

Document Object Model

Attribute nodes

- The opening tags of HTML elements can carry attributes and these are represented by attribute nodes in the DOM tree. For example the <a> tag has the *href* attribute.
- Attributes nodes are not *children* of the element that carries them; they are *part* of that element.

Text nodes

- Once you have accessed an element node, you can then reach the text within that element. This is stored in its own text node.

Lecture Outline

- Document Object Model
- Accessing Elements
- Accessing Form Elements
- Events

Accessing Elements

- DOM queries may return one element, or they may return a **NodeList**, which is a collection of nodes.

Return a Single Element Node

- Sometimes you will want to access one individual element (or a fragment of the page that is stored within that one element).

Return one or more Elements

- You may also want to select a group of elements, for example, every <h1> element in the page or every <li> element within a particular list.

Accessing Elements

```
<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">balsamic vinegar</li>
</ul>
```

List King

Buy groceries

- *fresh* figs
- pine nuts
- honey
- balsamic vinegar

Accessing Elements

Return a Single Element Node

getElementById()

- Selects an individual element given the value of its attribute. The HTML must have an **id** attribute in order for it to be selectable.

```
let el = document.getElementById('one');
el.style.color = "red";
```

Accessing Elements

Return a Single Element Node

querySelector()

- Uses CSS selector syntax that would select one or more elements. This method returns **only** the first of the matching elements.

```
let el = document.querySelector('li.hot');
el.style.color='green';
```

Accessing Elements

Return one or more Elements

getElementsByTagName()

- Select all elements on the page with the specified tag name.

```
let elements =  document.getElementsByTagName('li');
for (i=0; i < elements.length; i++) {
    elements[i].style.color='blue';
}
```

Accessing Elements

Return one or more Elements

getElementsByClassName()

- Selects one or more elements given the value of their **class** attribute. The HTML must have a **class** attribute for it to be selectable.

```
let elements =    document.getElementsByClassName('hot');
  for (i=0; i < elements.length; i++) {
     elements[i].style.color='orange';
  }
```

List King

Buy groceries

- *fresh* figs
- pine nuts
- honey
- balsamic vinegar

Accessing Elements

Return one or more Elements

querySelectorAll()

- Uses CSS selector syntax to select one or more elements and returns all of those that match.

```
let elements =  document.querySelectorAll('li.hot');
for (i=0; i < elements.length; i++) {
   elements[i].style.color='cyan';
}
```

List King

Buy groceries

- *fresh* figs
- pine nuts
- honey
- balsamic vinegar

Accessing Elements

- getElementsByTagName(), getElementsByClassName(), and querySelectorAll() return more than one element as a NodeList.
- For example, return the first element from any of the above examples use:

```
elements[0]
```

- If we want to traverse all the elements returned in the NodeList, we will need a loop as seen above.
- Note: There are more methods, which we will meet in further examples in the labs.

Accessing Elements

- To return/set the content (text) of an element, use **innerHTML**.

- To return and output the content of an element:

```
document.write(document.getElementById('two').innerHTML);
```

- To modify the content of an element:

```
document.getElementById('two').innerHTML='roasted nuts';
```

Lecture Outline

- Document Object Model
- Accessing Elements
- Accessing Form Elements
- Events

Accessing Form Elements

Text Box

Returning the value of a text box

- To get the value from a specific text box, you need to know it's **id** in order to use **document.getElementById()**. You also need the **value** attribute as this returns the value in the text box.

```
numdays=document.getElementById("days").value;
```

Setting the value of a text box

- To put a value into a text box, again you need to know it's id. To put the value of 2 into the text box (days):

```
document.getElementById("days").value=2;
```

Accessing Form Elements

Radio/Option buttons

- Because radio/option buttons only allow the selection of one option and they all have the same name, we will need a loop to process all the options. You determine if a radio/option button is selected by testing the **checked** attribute.

```
yourroom=document.getElementsByName("room");
if (yourroom[0].checked)
   {
     string="You require a Single room";
}
else if(yourroom[1].checked)
   {
     string="You require a Double room";
}
```

Accessing Form Elements

Checkbox

- We need to check each checkbox individually and you also need the **checked** attribute.

```
if(document.getElementById("dinner").checked)
{
 meal=parseInt(document.getElementById("dinner").value);
}
```

Accessing Form Elements

Pull down menu/Select

- For a pull down menu, we need to ascertain which option(s) have been selected. You also need the **selectedIndex** and **value** attributes.
- **selectedIndex** records the index of the chosen menu option.

Accessing Form Elements

```
rentals=document.getElementById("rentalPackage");
  if(rentals.selectedIndex===0)
   { rental=0; }
  else if(rentals.selectedIndex===1)
   { rental = parseInt(rentals.value); }
else if(rentals.selectedIndex===2)
 { rental = parseInt(rentals.value);
 }
```

Lecture Outline

- Document Object Model
- Accessing Elements
- Accessing Form Elements
- Events

Events

What is an Event?

- JavaScript's interaction with HTML is handled through **events** that occur when the user or the browser manipulates a page.
- When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

Events

Event	Detected when
focus	Form field gets focus
blur	Form field looses focus
change	Content of a field changes
select	Text is selected
click	Mouse clicks an object
dblclick	A form element or a link is clicked twice
dragdrop	A system file is dragged with a mouse and dropped onto the browser
keydown	A key is pressed but not released
keypress	A key is pressed
keyup	A depressed key is released
load	Page is finished loading
unload	Browser opens new document
mousedown	A mouse button is pressed
mousemove	The mouse is moved
mouseout	The mouse pointer moves off an element
mouseover	The mouse pointer is moved over an element
mouseup	The mouse button is released
move	A window is moved, maximised or restored either by the user or by a script
resize	A window is resized by the user or by a script
submit	Submit button on a form is clicked
reset	Reset button on a form is clicked

Events

- When an event has occurred, it is often described as having **fired** or been **raised**.
- Events are said to **trigger** a function or script.

Events

Event Handling

- When the user interacts with the HTML on a web page, there are three steps involved in getting it to trigger some JavaScript code. Together these steps are known as **Event Handling**.

1. Select the *element* node(s) you want the script to respond to.
2. Indicate which *event* on the selected node(s) will trigger the response.
3. State the *code* you want to run when the event occurs.

Events

Using Event Listeners

- The **addEventListener()** method attaches an event handler to the specified element.

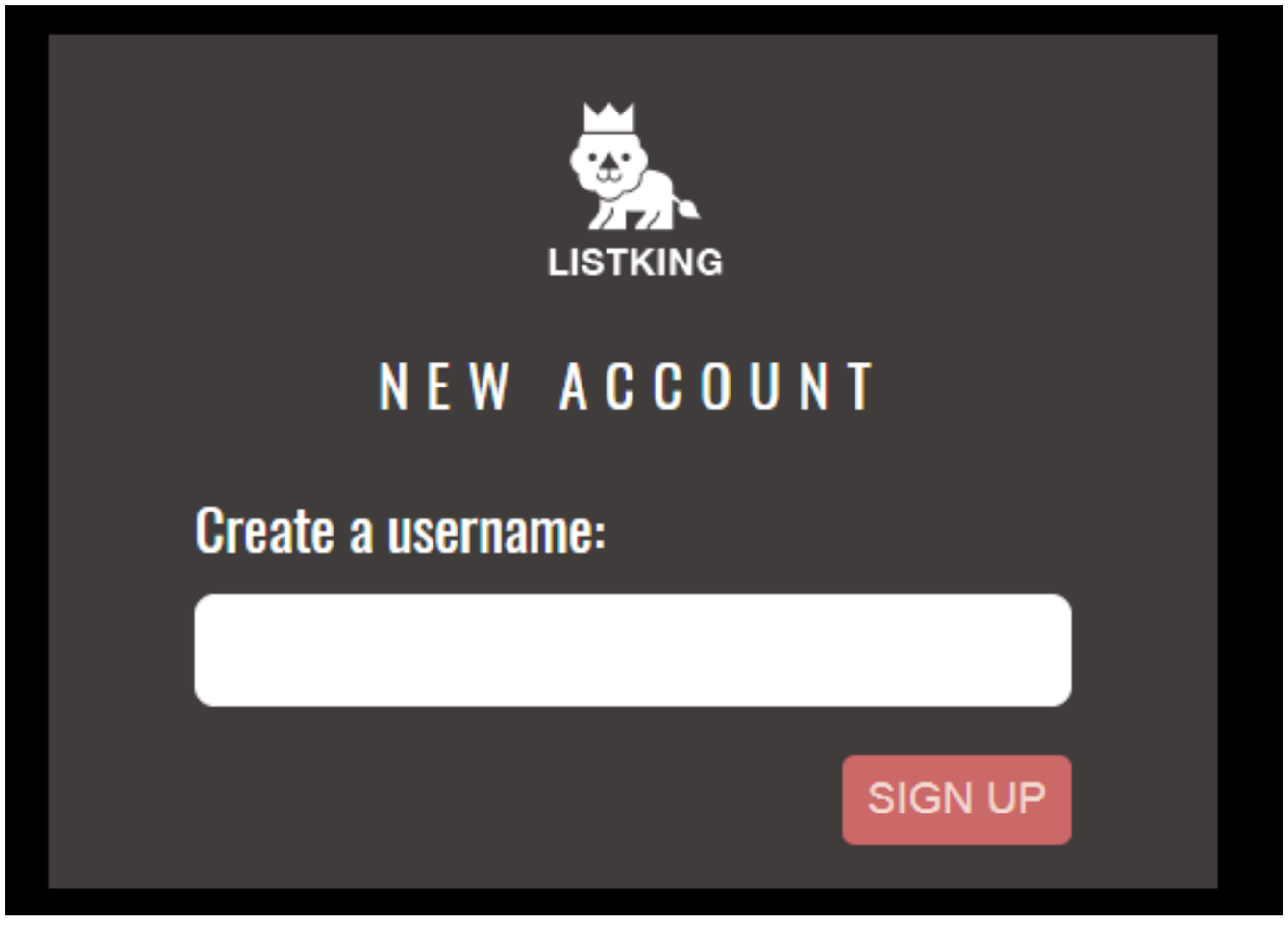
```
var myBtn = document.getElementById("btnClickMe");
myBtn.addEventListener("click",handleClick,false);
```

1. Parameter One is required. This is a String that specifies the name of the event. Do not use the "on" prefix. For example, use "click" instead of "onclick".
2. Parameter Two is required. This specifies the function to run when the event occurs.
3. Parameter Three is optional. This is a Boolean value that specifies whether the event should be executed in the capturing (true) or in the bubbling (false) phase.

Events

```
<form method="post" action="">
 <label for="username">Username: </label>
 <input type="text" id="username" autofocus>
 <div id="feedback"></div>
 <input type="submit" value="sign up">
</form>
```

Events

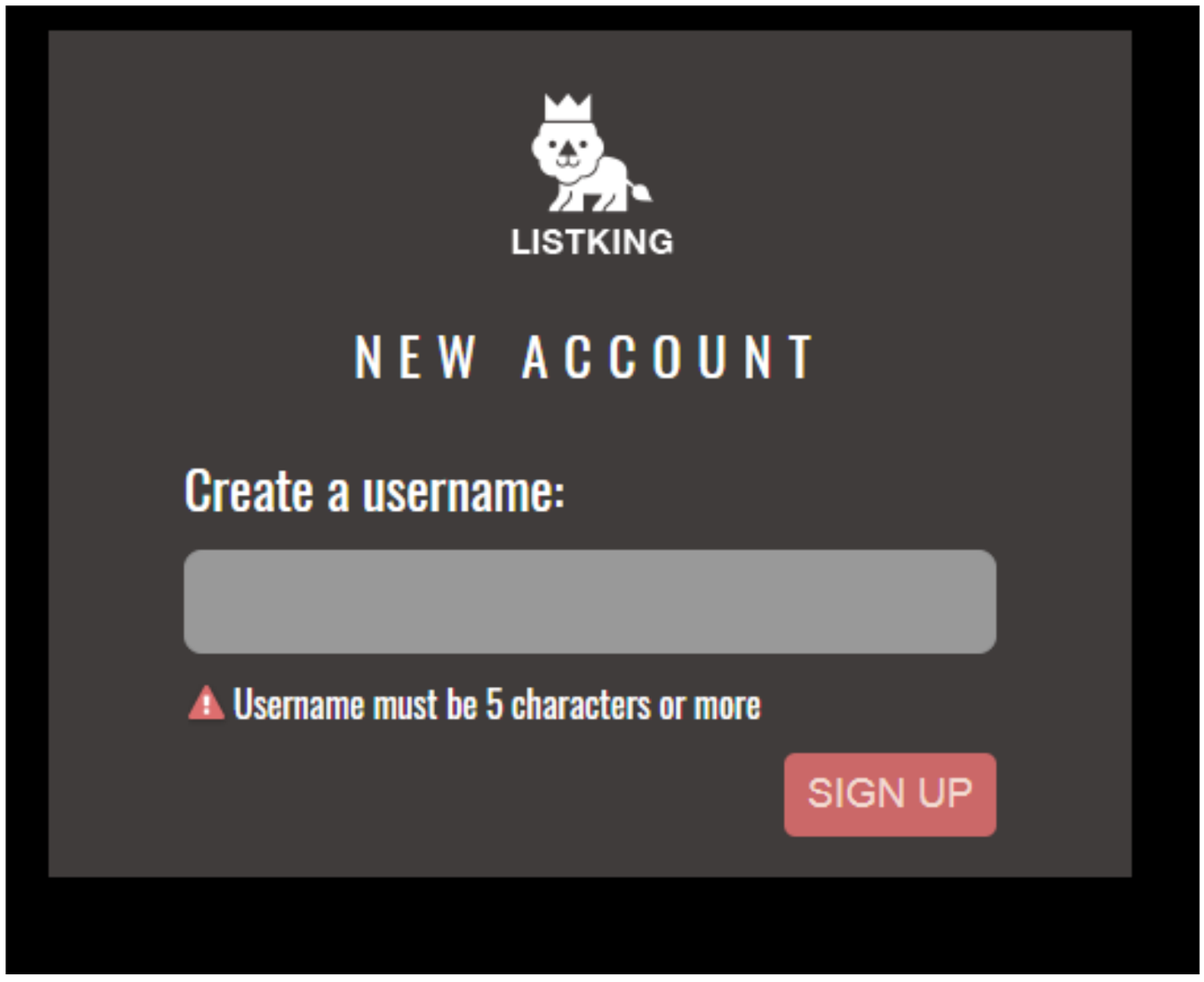


Events

```
var elUser = document.getElementById('username');
elUser.addEventListener('blur', checkUsername, false);

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

Events


LISTKING
NEW ACCOUNT
Create a username:
Username must be 5 characters or more
SIGN UP