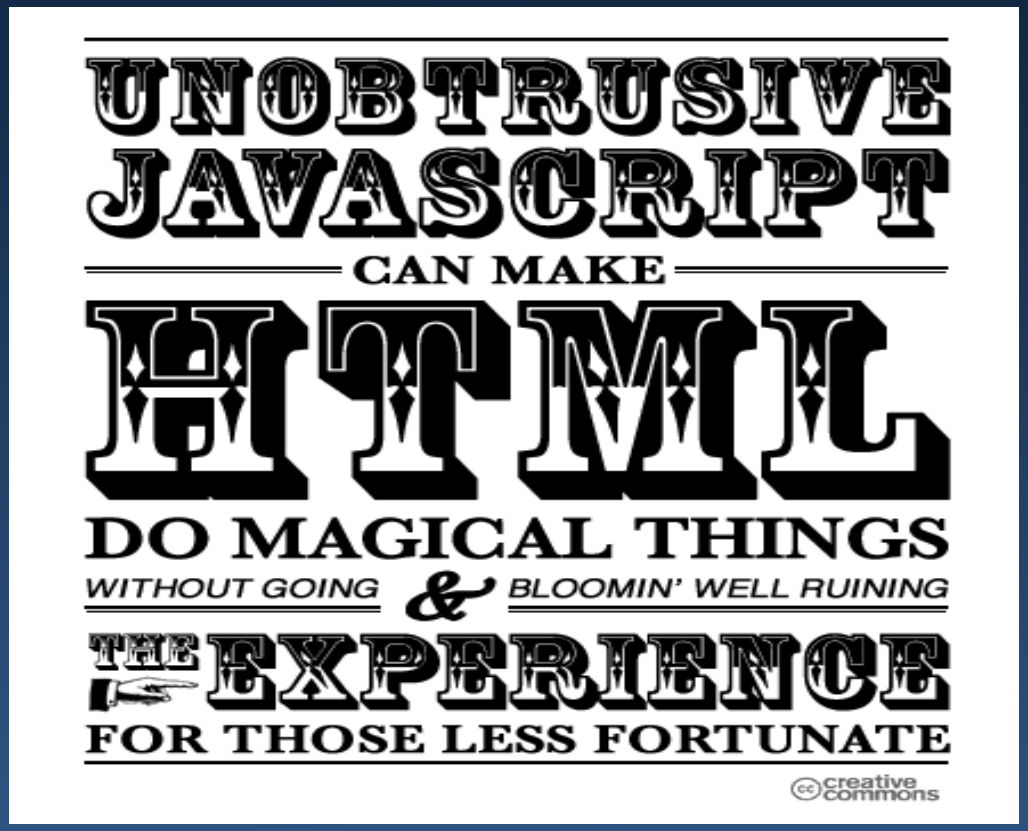

UNOBTRUSIVE
JAVASCRIPT
CAN MAKE
HTML
DO MAGICAL THINGS
WITHOUT GOING & BLOOMIN' WELL RUINING
THE EXPERIENCE
FOR THOSE LESS FORTUNATE
creative commons

Web Technology

JavaScript Advanced

computing@aberdeen

Last Week

- Comparison operators: == === != > < >= <=
- Logical operators: && || !
- Conditional Statement: if … else
- Loops: for while
- Functions: function fName(var1 …) { … }
- Popup Boxes: alert("sometext");
- Objects: String, Date, Array

Today – More JavaScript

Document Object Model

Magically manipulating web page content on the fly

Validating forms

Ensuring the users fill it in correctly!

HTML DOM

A W3C standard

- A standard object model for HTML
- A standard programming interface for HTML
- Platform- and language-independent

The HTML DOM defines the **objects and properties** of all HTML elements, and the **methods** (interface) to access them.

DOM Nodes

Everything in an HTML document is a node.

- The entire document is a document node
- Every HTML element is an element node
- The text in the HTML elements are text nodes
- Every HTML attribute is an attribute node
- Comments are comment nodes

DOM Example

```
<html>
  <head>
    <title>My Web Page</title>
  </head>
  <body>
    <h1>Welcome!</h1>
    <p>Hello world!</p>
  </body>
</html>
```

<html> is the root node

All other nodes in the document are contained within the root.

<html> node has two child nodes: <head> and <body>.

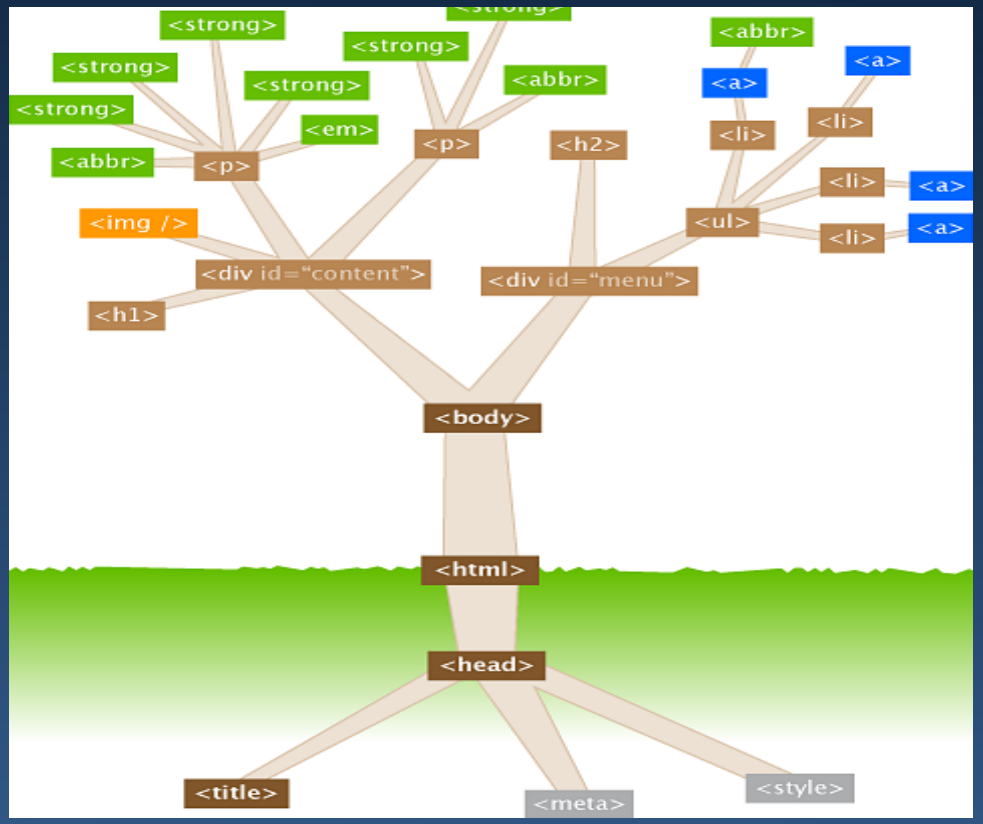
<head> node holds a <title> node.

<body> node holds a <h1> and <p> node.

The HTML Node Tree

The HTML *DOM views a HTML document as a tree-structure* called a ***node-tree.***

All nodes can be accessed through the tree. Their contents can be modified or deleted, and new elements can be created.



8

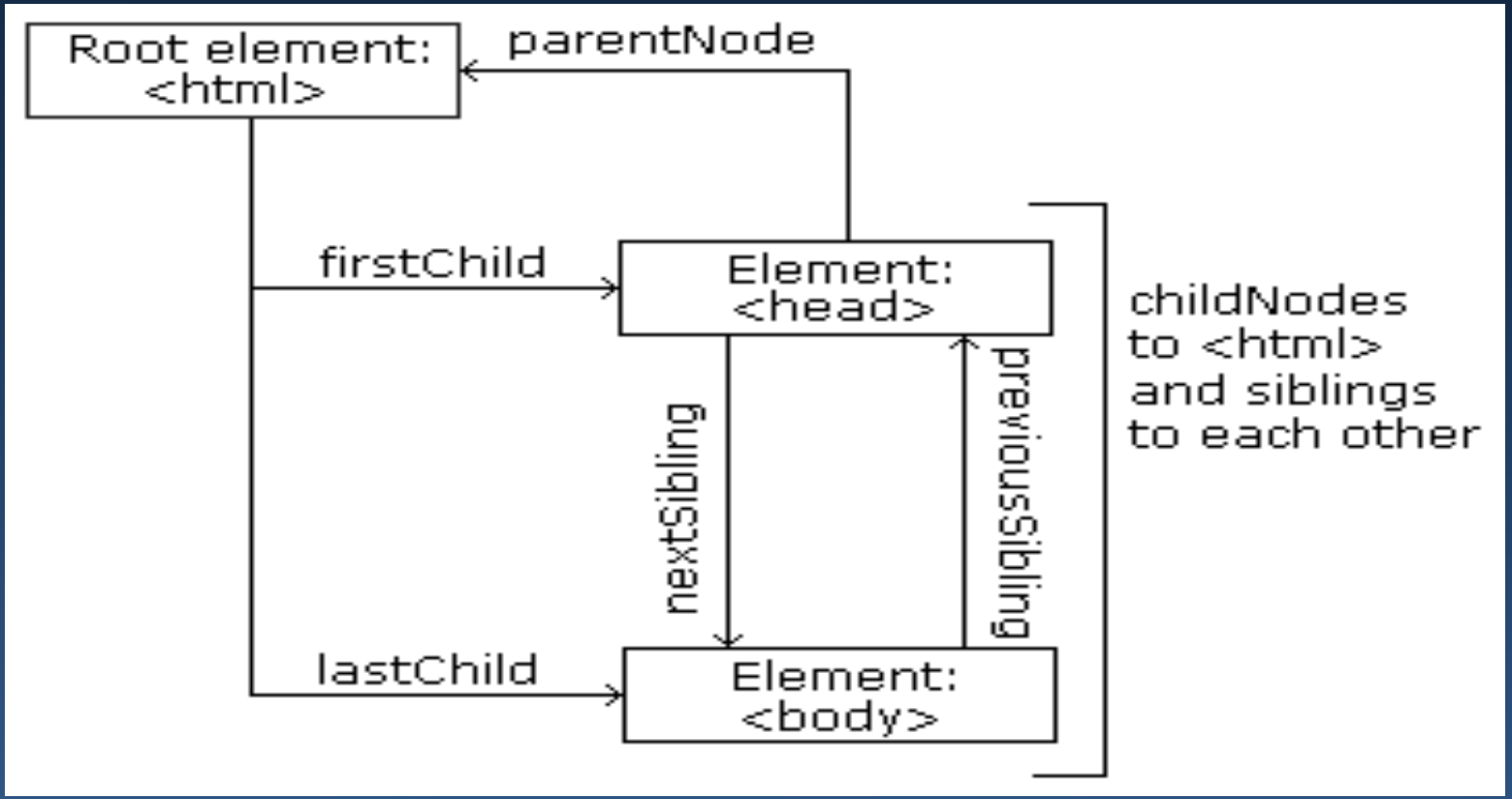
Node Parents, Children, and Siblings

The nodes in the node tree have a hierarchical relationship to each other.

Parent, *child*, and *sibling* terms describe the relationships.

- In a node tree, the top node is called the **root**
- Every node, except the root, has exactly **one parent** node
- A node can have any number of **children**
- A **leaf** is a node with no children
- **Siblings** are nodes with the same parent

Parent, Children, and Siblings Relationships



DOM Programming Interface

Node objects can be accessed with JavaScript or other programming languages.

- We'll will use JavaScript here

The programming interface of the DOM is defined by standard properties and methods.

- **Properties** are often referred to as something that is (i.e. the name of a node).
- **Methods** are often referred to as something that is done (i.e. remove a node).

Example HTML DOM Properties

x.innerHTML - the text value of x

x.nodeName - the name of x

x.nodeValue - the value of x

x.parentNode - the parent node of x

x.childNodes - the child nodes of x

x.attributes - the attributes nodes of x

Example HTML DOM Methods

x.getElementById(*id*) - get the element with a specified id

x.getElementsByTagName(*name*) - get array containing all elements with a specified tag name

x.appendChild(*node*) - insert a child node to x

x.removeChild(*node*) - remove a child node from x

The innerHTML Property

The easiest way to get or modify the content of an element

Not part of the W3C DOM specification

But is supported by all major browsers

Useful for

- returning or replacing the content of HTML elements (including <html> and <body>),
- viewing the source of a page that has been dynamically modified.

The Inner HTML Property - Example

Get the innerHTML (text) from the <p> element with id="intro":

```
<html>
<body>

<p id="intro">Hello World!</p>

<script type="text/javascript">
txt=document.getElementById("intro").innerHTML;
document.write("<p>The text from the intro paragraph: " + txt +
"</p>");
</script>

</body>
</html>
```

Hello World!

The text from the intro paragraph: Hello World!

childNodes and nodeValue

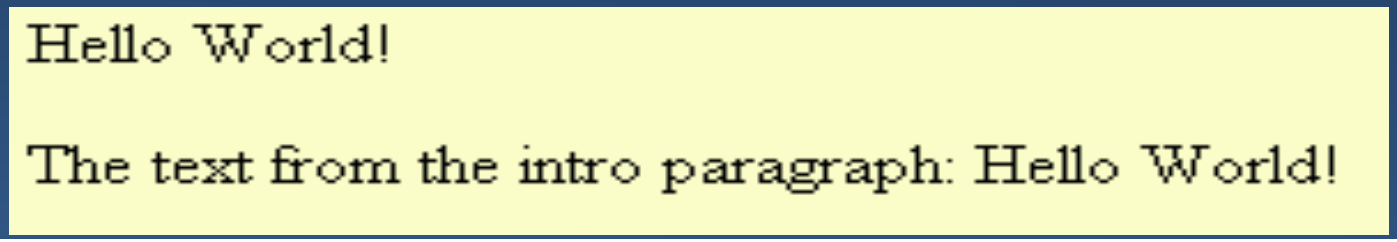
- We can also use the childNodes and nodeValue properties to get the content of an element.
- Get the value of the <p> element with id="intro":

```
<html>
<body>

<p id="intro">Hello World!</p>

<script type="text/javascript">
txt=document.getElementById("intro").childNodes[0].nodeValue;
document.write("<p>The  text from the intro paragraph: " + txt + "</p>");
</script>

</body>
</html>
```



Accessing Nodes: Three ways

By using the getElementById() method

```
document.getElementById("intro");
```

By using the getElementsByTagName() method

```
document.getElementsByTagName("p");
```

By navigating the node tree, using the node relationships

```
x = document.getElementById(“intro");
document.write(x.firstChild.nodeValue);
```

Question

- Which of the following will determine the number of anchor tags (<a>) on the page

1. document.getElementById('a').children.length;
2. document.['a‘].size;
3. document.getElementsByTagName('a').length;
4. document.getElementsByTagName('body')
   .children.size;

JavaScript Form Validation

JavaScript can be used to validate data in HTML forms before sending off the content to a server.

Typically checks:

- has the user left required fields empty?
- has the user entered a valid e-mail address?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Example

Form with name* email*, and comments

```
<form name="myForm" onsubmit="return
    validateForm();" method="post“
    action=“form.php”>
Name*:
    <input type="text" name="name">
Email*:
    <input type="text" name="email">
Comments:
    <textarea rows="10" cols="30"></textarea>
<input type="submit" value="Submit">
</form>
```

JavaScript

```
function validateForm(){
if (false == validate_required(
    document.forms["myForm"]["name"], "Name
    must be supplied")){
    return false;}
if (false == validate_email(
    document.forms["myForm"]["email"], “Valid
    email must be supplied")){
    return false;}
return true;}
```

Required Fields – Check for Value

```
function validate_required(field,alerttxt)
{

  if (field.value==null||field.value=="")
    {
    alert(alerttxt); return false;
    }
  else
    {
    return true;
    }
}
```

Minimum Length

```
function validate_min_length(field, minLength, alerttxt){
 if (field.value==null)  {
        alert(alerttxt); return false;
  }
  else {
        // remove all whitespace from the field value
        var value = field.value.replace(/\s/g, "");
        if (value.length < minLength) {
              alert(alerttxt); return false;
        } else  {
        return true;
       }
  }
```

Is a number provided

```
function validate_is_number(field,alerttxt){
  if (field.value==null)   {
        alert(alerttxt); return false;
    }
   else {
   // remove all whitespace from the field value
   var value = field.value.replace(/\s/g, "");
   return isNaN(value);
   }
}
```

Email Validation – Check Syntax

```
function validate_email(field,alerttxt)
{
  var apos=field. value.indexOf("@");
  var dotpos=field.value.lastIndexOf(".");
  if (apos<1||dotpos-apos<2)
    {   alert(alerttxt);
        return false;
    } else {return true;}
}
```

Question

Which of the following is NOT true about Javascript form validation

1. It provides quick feedback to the user
2. It can check if a valid email has been entered
3. It can be bypassed
4. It can check the correct type of input has been provided

Summary

- Document Object Model
- Form Validation
- Thursday
  - AJAX and jQuery
- Practical
  - Creating and using a design

26