COMP30680
Web Application Development

JavaScript part 1

David Coyle
d.coyle@ucd.ie

"JavaScript is easy to learn."
www.w3schools.com

Learning JavaScript

- JavaScript is a very rich programming language.
- There are many different ways of doing the same thing!
- The aim in this module is to introduce the core features of JavaScript. We won't be able to cover everything, but it will provide a starting point.
- Please also look at online materials, e.g. w3schools.com JavaScript tutorials: http://www.w3schools.com/js/default.asp
- **View Source.** A good way to learn JavaScript is to look at scripts other people have written. JavaScript, just like HTML, can be viewed by selecting view source on your browser.

Origin of JavaScript

- Was originally developed by Netscape, as *Mocha*, then *LiveScript*
- Became a joint venture of Netscape and Sun in 1995, renamed *JavaScript*
- Was standardized by the International Standards Association (ISO)
- Modern browsers contain JavaScript interpreters and so can execute JavaScript scripts
- ECMA-262 is the official name. ECMAScript 6 (released in June 2015) is the latest official version of JavaScript.

JavaScript is not Java!

- JavaScript and Java are completely different languages, both in concept and design. They are only related through syntax – both are C-like
- JavaScript be embedded in many different things, but its primary use is embedded in Web documents using the <script> tag.

JavaScript can be divided into three categories:

We will focus here

1. Core
   - The heart of the language, including its operators, expressions, statements, etc.
2. Client-side
   - The collection of objects that support control of the browser and interactions with the user
3. Server-side **
   - The collection of objects that make the language useful on the webserver

** We will focus on client-side JavaScript. For a nice article on server-side JavaScript see:
http://www.webreference.com/programming/javascript/rg37/index.html

HTML and JavaScript

HTML is largely static:

- plain pages
- text, images, links

JavaScript allows interaction:

- uses code to add functionality to web pages

JavaScript is embedded in HTML as scripts.

1. ***Automatic* Scripts** – these are executed by the browser when the page is loaded, without the visitor having to do anything.
2. ***Triggered* Scripts** – these react to *intrinsic* events (i.e., to something the visitor does)

Embedding JS in a HTML page

Where does it go?

- JavaScript can be placed in the <body> and the <head> sections of an HTML page.
- In HTML, JavaScript code must be inserted between <script> and </script> tags.

```
<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

Note: Older examples may use a type attribute: <script type="text/javascript">.
This type attribute is not required. JavaScript is the default scripting language in HTML.

The HTML DOM

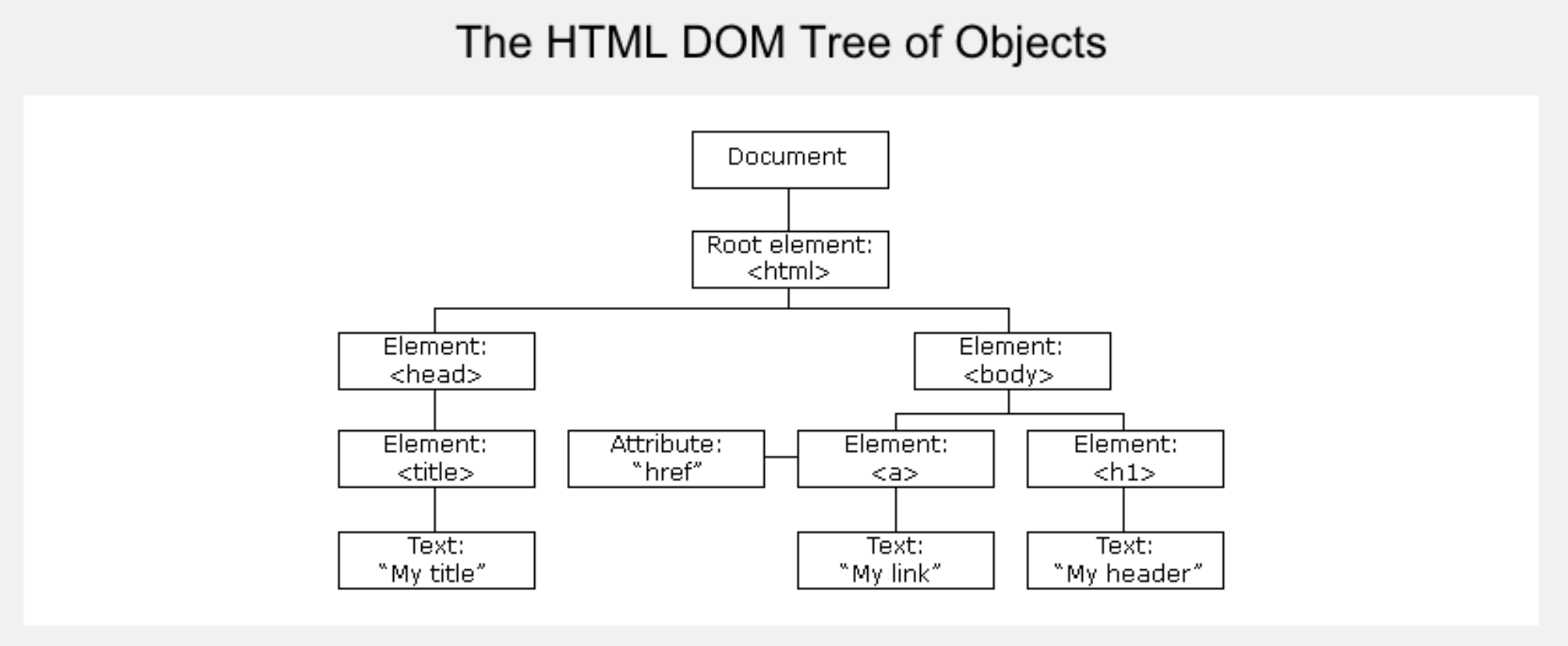
When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel (DOM) of the page.

The HTML DOM defines:

- The HTML elements as **objects**
- The **methods** to access all HTML elements
- The **properties** of all HTML elements
- The **events** for all HTML elements

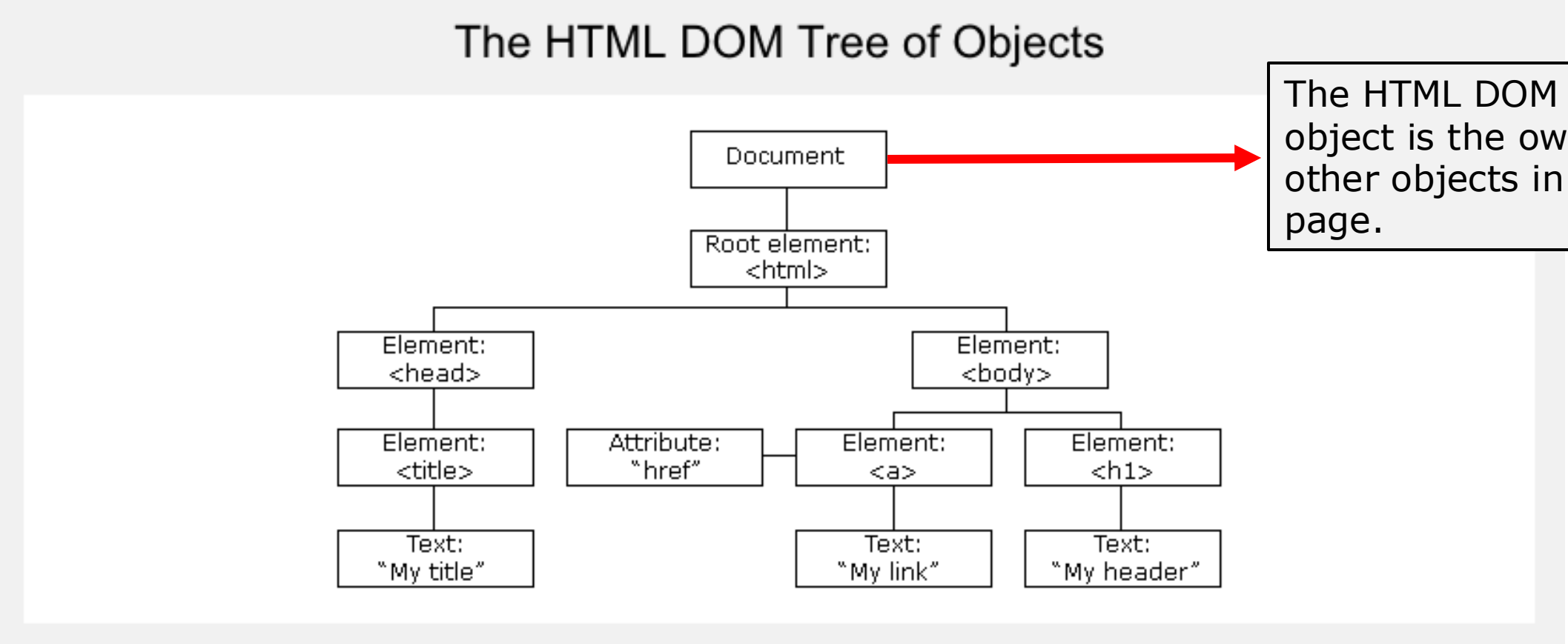
DOM objects

The **HTML DOM** model is constructed as a tree of **Objects**:



DOM objects

The **HTML DOM** model is constructed as a tree of **Objects**:



The HTML DOM document object is the owner of all other objects in your web page.

HTML DOM programming interface

The programming interface is the methods and properties of each object.

A **method** is an action you can perform on an object - like get, add or deleting an HTML element.

A **property** is a value that you can get or set - like changing the content or style of an HTML element.

```
<html>
<body>

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

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

</body>
</html>
```

getElementById is a **method**,

innerHTML is a **property**

Note: the script above is an example of an automatic script. It runs automatically when the page is loaded

DOM methods

The HTML DOM provides many different methods through with JavaScript can find and manipulate HTML elements.

Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

See http://www.w3schools.com/js/js_htmldom_elements.asp for a more complete list of methods to access elements.

DOM methods

Changing HTML Elements

Method	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element

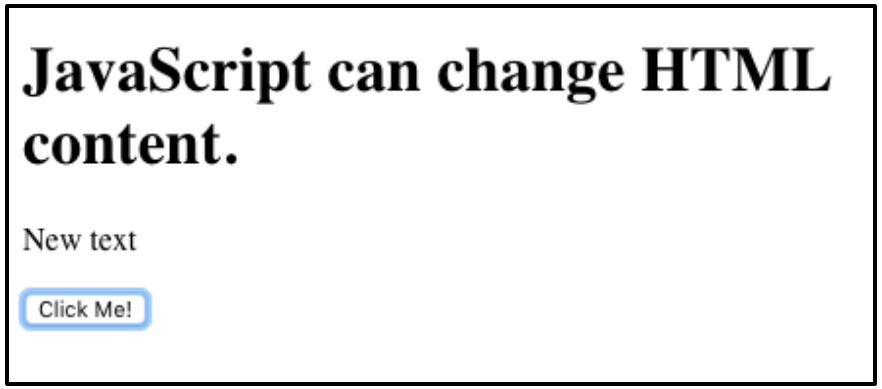
See http://www.w3schools.com/js/js_htmldom_html.asp for more examples of writing HTML and changing HTML elements.

A first example:

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <h1>JavaScript can change HTML content.</h1>
6
7  <p id="demo">Click the button to change this text content.</p>
8
9  <button type="button" onclick="changeText()">
10 Click Me!</button>
11
12 <script>
13
14 function changeText(){
15     document.getElementById('demo').innerHTML = 'New text!';
16 }
17
18 </script>
19
20
21 </body>
22 </html>
```

See: **js_example1.html**

When the button is pressed it triggers an onclick event, which calls the function changeText(). This function includes a statement that changes the text content of the the element with id=demo.



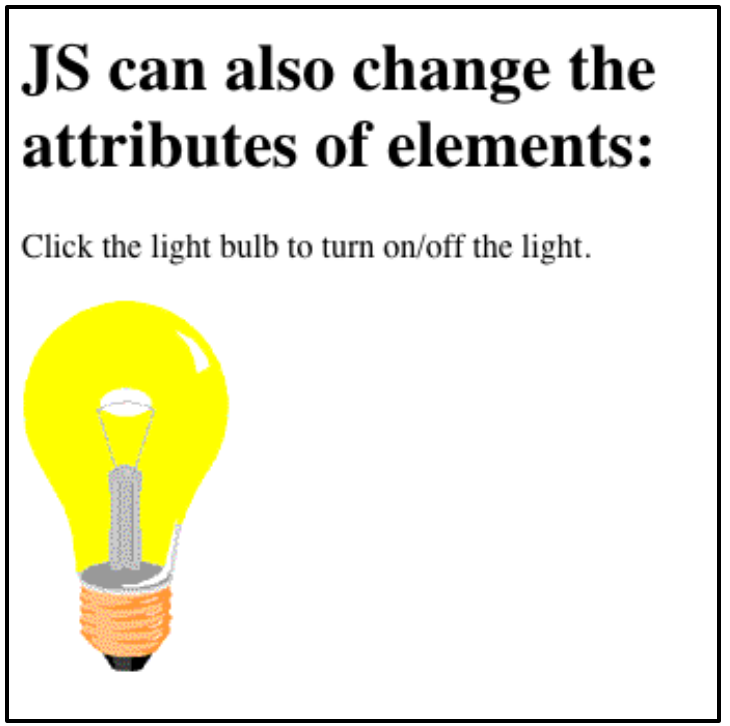
Slightly more complex

```
1   <!DOCTYPE html>
2   <html>
3   <body>
4
5   <h1>JS can also change the attributes of elements:</h1>
6
7   <p>Click the light bulb to turn on/off the light.</p>
8
9   <img id="myImage" onclick="changeImage()"
    src="images/pic_bulboff.gif" width="100" height="180">
10
11  <script>
12
13  function changeImage() {
14      var image = document.getElementById('myImage');
15      if (image.src.match("bulbon")) {
16          image.src = "images/pic_bulboff.gif";
17      } else {
18          image.src = "images/pic_bulbon.gif";
19      }
20  }
21
22  </script>
23
24
25  </body>
26  </html>
```

See: **js_example2.html**

When the image (lightbulb) is clicked it triggers an onclick event, which calls the function changeImage().

This function includes statements that create an variable ‘image’ and then uses an if/else statement to change the src attribute of the image.



DOM methods

Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

For a complete list of methods and properties see: http://www.w3schools.com/jsref/dom_obj_all.asp

DOM events

The HTML DOM allows you to execute code when an event occurs.

Events are generated by the browser when "things happen" to HTML elements.

We saw examples of **onclick** events in js_example1.html and js_example2.html.

Further examples of events include:

- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted
- When a user strokes a key

For a complete list of events see: http://www.w3schools.com/jsref/dom_obj_event.asp

Back to our previous example

```
1   <!DOCTYPE html>
2   <html>
3   <body>
4
5   <h1>JS can also change the attributes of elements:</h1>
6
7   <p>Click the light bulb to turn on/off the light.</p>
8
9   <img id="myImage" onclick="changeImage()"
    src="images/pic_bulboff.gif" width="100" height="180">
10
11  <script>
12
13  function changeImage() {
14      var image = document.getElementById('myImage');
15      if (image.src.match("bulbon")) {
16          image.src = "images/pic_bulboff.gif";
17      } else {
18          image.src = "images/pic_bulbon.gif";
19      }
20  }
21
22  </script>
23
24
25  </body>
26  </html>
```

See: **js_example2.html**

Within the <script> tag you see some of the key syntax and features of the JavaScript language.

- We can assign variables.
- Scripts are built up using statements.
- Every statement ends with a semi-colon.
- We can create conditional statements.
- We can create functions.

Embedding JS in a HTML - continued

We have said:

- JavaScript can be placed in the <body> and the <head> sections of an HTML page.
- In HTML, JavaScript code must be inserted between <script> and </script> tags.

There is also a third option:

The JavaScript code is put into an external file, which is attached to HTML, by reference:

```
<script src="/path/to/script.js"></script>
```

The /path/to/script.js is a relative path.

If you have a specific location including the full URL that is the absolute path. Relative paths are relative to your current location on the site.

Which is better?

Placing JavaScripts in external files has some advantages:

- It separates HTML and code.
- It makes HTML and JavaScript easier to read and maintain.
- Cached JavaScript files can speed up page loads.

You can place an external script reference in <head> or <body> as you like.
The script will behave as if it was located exactly where the <script> tag is located.

myScript.js

```
function myFunction() {
   document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

```
<!DOCTYPE html>
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

Automatic scripts

***Automatic* Scripts** – these are executed by the browser when the page is loaded, without the visitor having to do anything.

```
<html>
<body>
  <h1>Counting rabbits</h1>

  <script>
    for(var i=1; i<=3; i++) {
      alert("Rabbit "+i+" out of the hat!")
    }
  </script>

  <h1>...Finished counting</h1>

</body>
</html>
```

Triggered scripts

Triggered **Scripts** – these react to *intrinsic* events (i.e., to something the visitor does)

```
<html>
  <head>
    <script>
      function count_rabbits() {
        for(var i=1; i<=3; i++) {
          alert("Rabbit "+i+" out of the hat!");
        }
      }
    </script>
  </head>

  <body>

  <h2>Press the button to start</h2>

    <input type="button" onclick="count_rabbits()" value="Count rabbits!"/>

  </body>

</html>
```

Questions, Suggestions?

Next class:

JavaScript part 2 – Variables, functions and conditional statements.