Programming 1

Javascript
Lecture #3: HTML and the document object model
(or “everything you’ve learned about Javascript is a lie”)

CLICKER CHANNEL 82

The story so far

- `console.log` to print stuff out
- `prompt` to get input from the user

The story so far

- ~~`console.log` to print stuff out~~
- ~~`prompt` to get input from the user~~

it's a LIE!

Javascript and user input/output

- It is very, very rare for real-world Javascript code to ever use `console.log` or prompt
  - `prompt` doesn't always work in all browsers
  - `console.log` doesn't actually get *seen* by the end-user unless they go into the developer console in their browser
- Most real-world Javascript works in conjunction with HTML pages
- The Javascript will manipulate elements in the HTML to change the page that the user sees in the browser
- It will also read elements in the HTML (such as form fields) to get user input from the browser

Caveat emptor ("buyer, beware")

- Only Team Solo get taught HTML on this module
- The rest of you, we hope that you will either already know HTML or will be able to pick it up as we go during this unit
  - …you are degree level students doing a computing degree, after all… ☺
- However, you won’t be expected to compose original HTML as part of this unit - we'll give you some "potted" HTML pages
- Your Javascript code will read and manipulate these "potted" pages in order to interact with the user

Introducing the DOM

- The *document object model* or DOM is a model that describes the structure of an HTML page
- It describes where each HTML element is in relation to others, and the overall page itself
- You can think of it as being like a tree

Navigating the DOM

```
<html>
  <head>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
    <h1>Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <h1>Good here, isn't it?</h1>
  </body>
</html>
```

Navigating the DOM

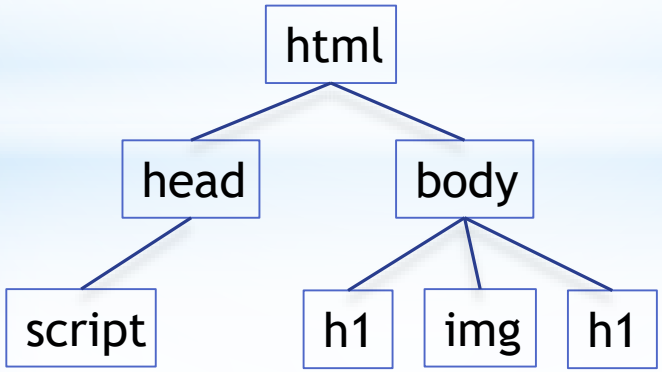
```
<html>
  <head>
    <script typ                                  t.js"></script>
  </head>
  <body>
    <h1>Hello A
    <img src="h                                  rRingo.jpg"/>
    <h1>Good he
  </body>
</html>
```

Hello Again

Good here, isn't it?

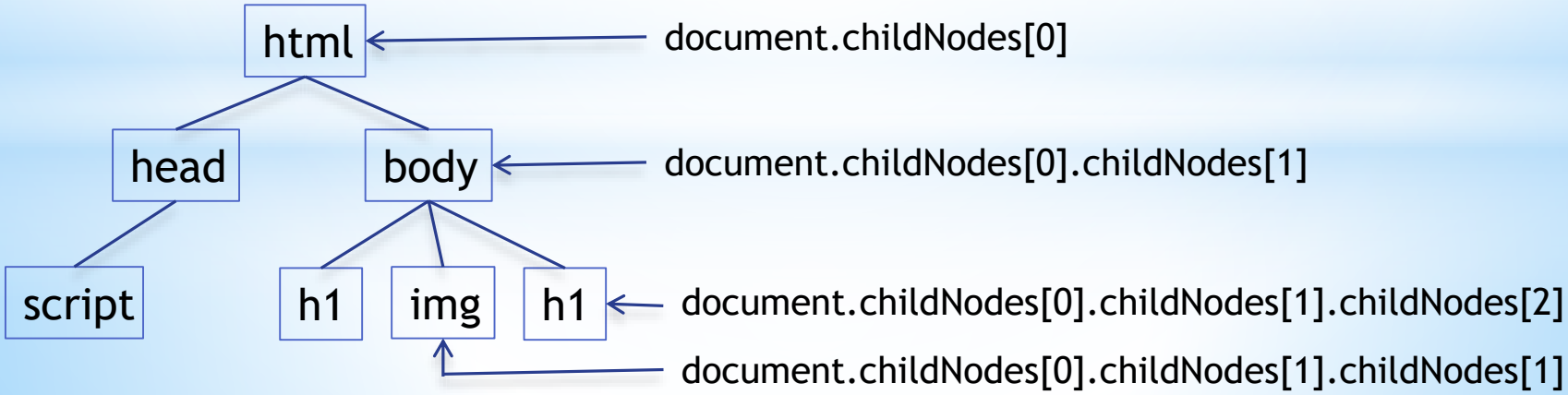
Navigating the DOM

```
<html>
  <head>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
    <h1>Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <h1>Good here, isn't it?</h1>
  </body>
</html>
```



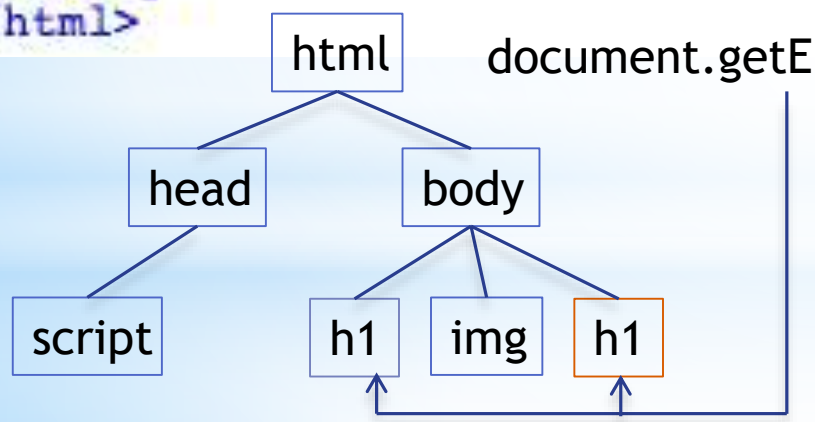
Navigating the DOM

```
<html>
  <head>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
    <h1>Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <h1>Good here, isn't it?</h1>
  </body>
</html>
```



Navigating the DOM

```
<html>
  <head>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
    <h1>Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <h1>Good here, isn't it?</h1>
  </body>
</html>
```



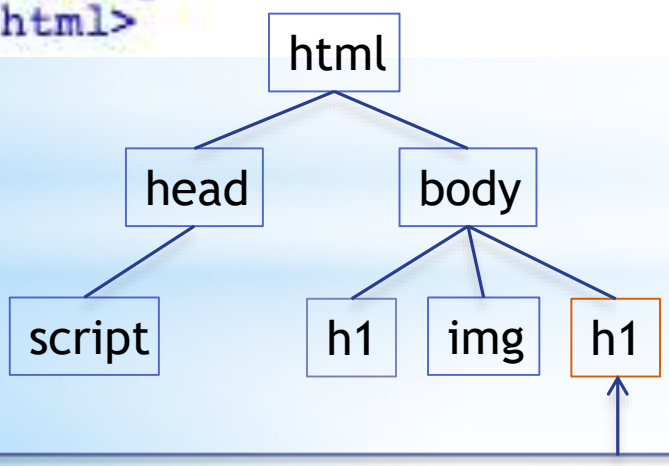
document.getElementsByTagName("h1") → h1 h1

document.getElementsByTagName("h1")[0]

document.getElementsByTagName("h1")[1]

Making changes in the DOM

```
<html>
  <head>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
    <h1>Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <h1>Good here, isn't it?</h1>
  </body>
</html>
```




```
document.getElementsByTagName("h1")[1].innerHTML = "Boring here, isn't it?"
```

Making changes in the DOM

```
<html>
  <head>
    <script t
  </head>
  <body>
    <h1>Hello
    <img src=
    <h1>Good
  </body>
</html>
```

```
.js"></script>

Ringo.jpg"/>
```

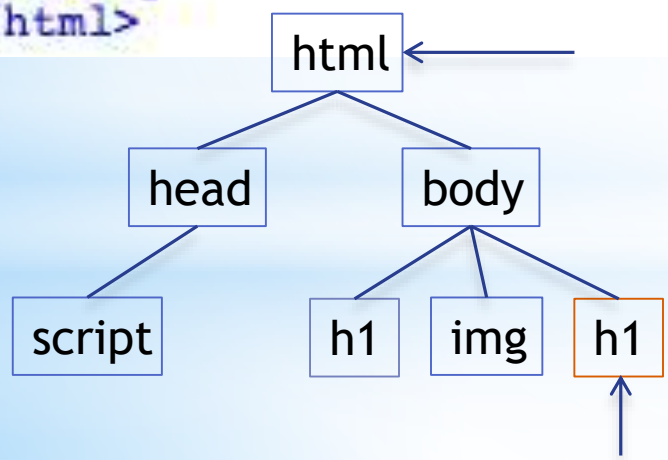
head

script

document.getElementsby tagName("h1")[1].innerHTML = "Boring here, isn't it?"

Making changes in the DOM

```
<html>
  <head>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
    <h1>Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <h1>Good here, isn't it?</h1>
  </body>
</html>
```




```
var myTargetElement = document.getElementsByTagName("h1")[1];
myTargetElement.innerHTML = "Boring here, isn't it?"
```

The ID element

- Elements in HTML can be labeled with an ID attribute
- You can then use document.getElementById to access them specifically, by this label

```
<html>
  <head>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
    <h1>Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <h1 id="myTarget">Good here, isn't it?</h1>
  </body>
</html>
```

The id attribute

- Elements in HTML may be labeled with an ID attribute
- You can then use document.getElementById to access them specifically, by this label

```
<html>
  <head>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
    <h1>Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <h1 id="myTarget">Good here, isn't it?</h1>
  </body>
</html>
```

The id attribute

```
var oneToChange = document.getElementById("myTarget");
oneToChange.innerHTML = "Boring here, isn't it?";
```

or

```
document.getElementById("myTarget").innerHTML = "Boring here isn't it?"
```

```
<html>
  <head>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
    <h1>Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <h1 id="myTarget">Good here, isn't it?</h1>
  </body>
</html>
```

The id attribute

```
var oneToChange = document.getElementById("myTarget");
oneToChange.innerHTML = "Boring here, isn't it?";
```

or

```
document.getElementById("myTarget").innerHTML = "Boring here isn't it?"
```

```
<html>
  <head>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
    <h1>Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <h1 id="myTarget">Good here, isn't it?</h1>
  </body>
</html>
```

Boring here, isn't it?

The class attribute

- Each element can also have a class
- Unlike IDs, the class attribute can be shared among more than one element, e.g. as in this HTML excerpt…

```
<body>
    <h1>List of students</h1>
    <ul>
        <li class="firstYear">Bill Smith</li>
        <li class="firstYear">Fred Bloggs</li>
        <li class="SecondYear">Katie Hole</li>
        <li class="SecondYear">Abdul Karim</li>
        <li class="firstYear">Carly Simmons</li>
        <li class="thirdYear">Jay Patel</li>
        <li class="thirdYear">John Thomas</li>
        <li class="thirdYear">Mandy Saunders</li>
        <li class="firstYear">Joe Soap</li>
        <li class="SecondYear">Neil Harris</li>
    </ul>
</body>
```

getElementsByClass

- We can target all elements of a specific class with **`getElementsByClass`**

```
var firstYears = document.getElementsByClass("firstYear");
```

- This would target the elements highlighted and ONLY these elements:

```
<body>
    <h1>List of students</h1>
    <ul>
        <li class="firstYear">Bill Smith</li>
        <li class="firstYear">Fred Bloggs</li>
        <li class="SecondYear">Katie Hole</li>
        <li class="SecondYear">Abdul Karim</li>
        <li class="firstYear">Carly Simmons</li>
        <li class="thirdYear">Jay Patel</li>
        <li class="thirdYear">John Thomas</li>
        <li class="thirdYear">Mandy Saunders</li>
        <li class="firstYear">Joe Soap</li>
        <li class="SecondYear">Neil Harris</li>
    </ul>
</body>
```

getElementsByClass

- getElementsByClass returns an array* – not a single element
- So, the result will not be singular – you must either know which element you're looking for, or iterate through to find the correct one

```
var firstYears = document.getElementsByClass("firstYear");
console.log(firstYears[2].innerHTML);
```

```
<body>
    <h1>List of students</h1>
    <ul>
        <li class="firstYear">Bill Smith</li>
        <li class="firstYear">Fred Bloggs</li>
        <li class="SecondYear">Katie Hole</li>
        <li class="SecondYear">Abdul Karim</li>
        <li class="firstYear">Carly Simmons</li>
        <li class="thirdYear">Jay Patel</li>
        <li class="thirdYear">John Thomas</li>
        <li class="thirdYear">Mandy Saunders</li>
        <li class="firstYear">Joe Soap</li>
        <li class="SecondYear">Neil Harris</li>
    </ul>
</body>
```

* strictly speaking, it returns a collection of elements – but you can treat it like an array…

getElementsByClass

- getElementsByClass returns an array* – not a single element
- So, the result will not be singular – you must either know which element you're looking for, or iterate through to find the correct one

```
var firstYears = document.getElementsByClass("firstYear");
for (var i = 0; i < firstYears.length; i++)
{
  var current = firstYears[i];
  if (current.innerHTML.indexOf("Joe") != -1)
  {
    console.log(current.innerHTML);
  }
}
```

```
<body>
    <h1>List of students</h1>
    <ul>
        <li class="firstYear">Bill Smith</li>
        <li class="firstYear">Fred Bloggs</li>
        <li class="SecondYear">Katie Hole</li>
        <li class="SecondYear">Abdul Karim</li>
        <li class="firstYear">Carly Simmons</li>
        <li class="thirdYear">Jay Patel</li>
        <li class="thirdYear">John Thomas</li>
        <li class="thirdYear">Mandy Saunders</li>
        <li class="firstYear">Joe Soap</li>
        <li class="SecondYear">Neil Harris</li>
    </ul>
</body>
```

* strictly speaking, it returns a collection of elements – but you can treat it like an array…

getElementsByTagName

- Will give you all elements of a certain tag type
  - document.getElementsByTagName("p")
    - gives you all the paragraphs
  - document.getElementsByTagName("h1")
    - gives you all the top level headings
  - document.getElementsByTagName("img")
    - gives you all the images
    - …and so on…!

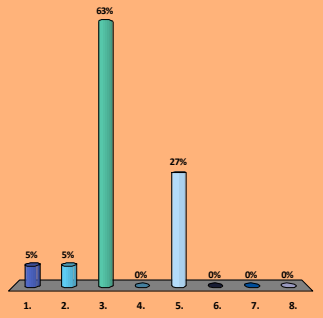
Remember the grammar

- It doesn't matter how you "get" an element
  - You can get an element with getElementById
  - You can get a group of elements with getElementsByClass or getElementsByTagName, and then extract a single one from the collection
- If you have a "block" in your code that is a single HTML element, you can read its innerHTML or any other attribute that's valid for the given element

What would be returned by this function if it was called on the following HTML fragment?

```
function getBit()
{
   var result = document.getElementsByTagName("b")[2];
   return result;
}

<html>
  …head element assumed here…
  <body>
     <p>The thing about <b>Black Pudding</b> that
        makes it <b>really</b> interesting is that
        when it's really good, even the white bits
        in it are <b>black</b>.</p>
  </body>
</html>
```

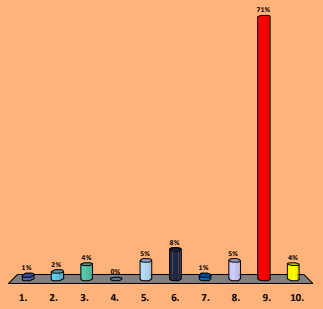


1. It would return all of the bold text elements
2. It would return the element containing the text *really*
3. It would return the element containing the text *black*
4. It would return the string *really*
5. It would return the string *black*
6. It would print the text *really*
7. It would print the text *black*
8. Something else

What would be returned by this function if it was called on the following HTML fragment?

```
function getBit() {
   var result = document.getElementsByTagName("p")[1];
   result = result.childNodes[3];
   return result.innerHTML;
}

<html>
  …head element assumed here…
  <body>
     <p>It's complicated because <b>Fred</b> is
        married to <b>Jane</b> but having an affair
        with <b>Susan</b>.</p>
     <p>What <b>Fred</b> didn't know was that <b>Jane</b> was
        <b>also</b> having an affair with
        <b>Susan</b>!</p>
  </body>
</html>
```



1. The first element containing the text *Fred* (in the first paragraph)
2. The second element containing the text *Fred* (in the second paragraph)
3. The string value *Fred*
4. The first element containing the text *Jane* (in the first paragraph)
5. The second element containing the text *Jane* (in the second paragraph)
6. The string value *Jane*
7. The first element containing the text *Susan* (in the first paragraph)
8. The second element containing the text *Susan* (in the second paragraph)
9. The string value *Susan*
10. Something else

Text blocks are elements too!

```
function getBit() {
   var result = document.getElementsByTagName("p")[1];
   result = result.getChildNodes[3];
   return result.innerHTML;
}

<html>
  …head element assumed here…
  <body>
     <p> It's complicated because <b>Fred</b> is
        married to <b>Jane</b> but having an affair
        with <b>Susan</b>. </p>
     <p> What <b>Fred</b> didn't know was that <b>Jane</b> was
        <b>also</b> having an affair with
        <b>Susan</b> ! </p>
  </body>
</html>
```

Text blocks are elements too!

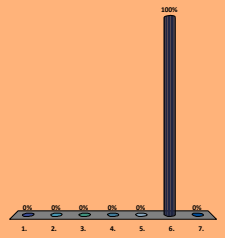
```
function getBit() {
   var result = document.getElementsByTagName("p")[1];
   result = result.getChildNodes[3];
   return result.innerHTML;
}

<html>
  …head element assumed here…
  <body>
     <p> It's complicated because <b>Fred</b> is
         married to <b>Jane</b> but having an affair
         with <b>Susan</b>. </p>
     <p> What <b>Fred</b> didn't know was that <b>Jane</b> was
         <b>also</b> having an affair with
         <b>Susan</b> ! </p>
  </body>
</html>
```

What would be the effect of running this function?

```
function getBit() {
   var target = document.getElementsByClass("content");
   target[1].childNodes[3].innerHTML = "repeatedly";
}

<html>
  …head element assumed here…
  <body>
    <h1>Instructions</h1>
    <p class="content"><b>First of all</b>, get a <b>hammer</b>.</p>
    <p class="note">(make <b>sure</b> it's a </b>big</b> one!)</p>
    <p class="content">Then, apply it <b>softly</b> and
      <b>swiftly</b> to the student's head.</p>
  </body>
</html>
```



1. The text *First of all* would change to *repeatedly*
2. The word *hammer* would change to *repeatedly*
3. The word *sure* would change to *repeatedly*
4. The word *big* would change to *repeatedly*
5. The word *softly* would change to repeatedly
6. The word *swiftly* would change to *repeatedly*
7. Something else

Using CSS selectors to target DOM elements

- In HTML5, some new Javascript commands came in that allow us to target DOM elements using CSS selectors

- **`document.querySelector`**
  - gets a **single element** based on the CSS selector you supply

- **`document.querySelectorAll`**
  - gets a **collection of elements** based on the CSS selector you supply
    - …think of it like giving you an array of elements
    - you could iterate through this "array" using a for loop and **`.length`** to find out how many were returned

CSS selectors

- CSS selectors are a powerful way to navigate the DOM
- You can select a specific type of element by specifying it
  - e.g. specifying **`p`** would give you all paragraphs, specifying **`img`** would give you all images…
- You can specify elements with a certain class by using a full stop in front of the class name
  - e.g. **`.content`** would give you all elements with a class of *content*
    - …note that it wouldn't matter what element type these were! If you had a paragraph, a div and an image with a class of *content* then this selector would give you them all!
- You can specify elements with a certain ID by using a hash sign in front of the ID name
  - e.g. #headline would give you the element with the ID *headline*
    - …the element type would be irrelevant
    - …remember that IDs are unique. So selecting by ID will only ever give a single result

CSS selectors

- You can also nest CSS selectors
  - p.content
    - …would give you all paragraphs with a class of *content*
  - div#footer
    - …would give you the single div with an ID of *footer*
      - (why only a single div?)
  - div p
    - …would give you all paragraphs that are inside a div (but not any other paragraphs)
  - div>p
    - …would give you all paragraphs that are direct descendents of a div (but not any other paragraphs)
  - div#footer p.contact a
    - …would give you all links inside any paragraph element with a class of contact that is inside a div of ID footer ☺

.querySelector example

HTML

(we'll take the structural stuff as read)

```
<body>
  <div id="foo">Hello</div>
  <div class="bar">There</div>
  <p class="bar">CSS</p>
  <div class="bar">
    <p class="woo">selectors</p>
    <p id="yay">are</p>
    <p>great</p>
  </div>
</body>
```

Javascript

```
var p = document.querySelector("div#foo");
console.log(p.innerHTML);

var x = document.querySelector("p#yay");
console.log(x.innerHTML);

var y = document.querySelector("div.bar p.woo");
console.log(y.innerHTML);

var z = document.querySelector("p.woo");
console.log(z.innerHTML);

p.innerHTML = "Goodbye";
```

.querySelectorAll example

HTML

(we'll take the structural stuff as read)

```
<body>
  <div id="foo">Hello</div>
  <div class="bar">There</div>
  <p class="bar">CSS</p>
  <div class="bar">
     <p class="woo">selectors</p>
     <p id="yay">are</p>
     <p>great</p>
  </div>
</body>
```

Javascript

```
var p = document.querySelectorAll("p");
for (var i = 0; i < p.length; i++)
{
   console.log(p[i].innerHTML);
}

p[3].innerHTML = "rubbish";

var x = document.querySelectorAll(".bar");
console.log(x[1].innerHTML);
```

Changing image elements

- You can change the src attribute on an image element to change what picture gets displayed

```
<html>
  <head>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
    <h1>Hello Again</h1>
    <img id="cat" src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <h1>Good here, isn't it?</h1>
  </body>
</html>
```

```
document.getElementById("cat").src =
"http://fetlar.kingston.ac.uk/dog.jpg";
```

Changing image elements

- You can change the src attribute on an image element to change w

```
<html>
  <head>
    <script
  </head>
  <body>
    <h1>Hel
    <img id
    <h1>Goo
  </body>
</html>
```

```
"></script>

otherRingo.jpg"/>
```

```
document.getEle
"http://fetlar.
```

Changing image elements

- You can change the src attribute on an image element to change w

```
<html>
  <head>
    <script
  </head>
  <body>
    <h1>Hel
    <img id
    <h1>Goo
  </body>
</html>
```

```
document.getEle
"http://fetlar.
```

```
"></script>

otherRingo.jpg"/>
```

Change styles of elements

```
<html>
  <head>
    <style type="text/css">
        .label { color : blue; }
        .error { color : red; }
    </style>
  </head>
  <body>
    <h1 class="label">Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <p class="label">Good here, isn't it?</p>
  </body>
</html>
```

- CSS styles can be applied to all HTML elements with a specific class attribute
- So in this example, the page when viewed in the browser would look like…

Hello Again

Good here, isn't it?

```
<html>
  <hea
    <s

    </s
  </hea
  <body
    <h
    <i
    <p
  </bo
</html>
```

```
g" />
```

- CSS styl specific class at
- So in th yser would look lik

Change styles of elements

```
<html>
  <head>
    <style type="text/css">
        .label { color : blue; }
        .error { color : red; }
    </style>
  </head>
  <body>
    <h1 class="label">Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <p class="label">Good here, isn't it?</p>
  </body>
</html>
```

- You can modify the class in your Javascript code…

```
document.getElementsByTagName("p")[0].className = "error";
```

(Hello Again ents

```
<html>
  <hea
    <s

    </
  </he
  <bod
    <h
    <i
    <p
  </bo
</html
```

```
g" />
```

- You car

```
docume
or";
```

Good here, isn't it?

Change styles of elements

```
<html>
  <head>
    <style type="text/css">
        .label { color : blue; }
        .error { color : red; }
    </style>
  </head>
  <body>
    <h1 class="label">Hello Again</h1>
    <img src="http://fetlar.kingston.ac.uk/anotherRingo.jpg"/>
    <p class="label">Good here, isn't it?</p>
  </body>
</html>
```

- If there's not a class that fits the style you want, you can specify styles directly, e.g.

```
document.getElementsByTagName("p")[0].style.color = "green";
```

- W3schools has a good list of all of these:
  - http://www.w3schools.com/jsref/dom_obj_style.asp

Input elements

```
<html>
  <body>
    <div>Given Name <input id="givenname" size="30"/></div>
    <div>Family Name <input id="familyname" size="30"/></div>
  </body>
</html>
```

Given Name

Family Name

- Input elements let the user interact and type data into an HTML page once it's been rendered in the browser
- You can read what the user types in Javascript with the `.value` property on an input element

Input elements

```
<html>
  <body>
    <div>Given Name <input id="givenname" size="30"/></div>
    <div>Family Name <input id="familyname" size="30"/></div>
  </body>
</html>
```

Given Name Paul

Family Name Neve

```
var firstname = document.getElementById("givenname").value;
var surname = document.getElementById("familyname").value;
```

Input elements

```
<html>
  <body>
    <div>Given Name <input id="givenname" size="30"/></div>
    <div>Family Name <input id="familyname" size="30"/></div>
  </body>
</html>
```

Given Name Paul

Family Name Neve

```
var firstname = document.getElementById("givenname").value;
```

Input elements

```
<html>
  <body>
    <div>Given Name <input id="givenname" size="30"/></div>
    <div>Family Name <input id="familyname" size="30"/></div>
  </body>
</html>
```

Given Name Paul

Family Name Neve

Paul

```
var firstname = document.getElementById("givenname").value;
```

Input elements

```
<html>
  <body>
    <div>Given Name <input id="givenname" size="30"/></div>
    <div>Family Name <input id="familyname" size="30"/></div>
  </body>
</html>
```

Given Name Paul

Family Name Neve

- Can also *set* an input element's value, too:

```
document.getElementById("familyname").value = "Smith";
```

Input elements

```
<html>
  <body>
    <div>Given Name <input id="givenname" size="30"/></div>
    <div>Family Name <input id="familyname" size="30"/></div>
  </body>
</html>
```

Given Name Paul

Family Name ~~Neve~~ Smith

- Can also *set* an input element's value, too:

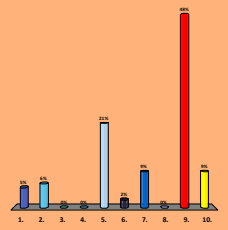
```
document.getElementById("familyname").value = "Smith";
```

- Note that value ONLY works on input elements – i.e. things that the end user can interact with and change
- For other elements (e.g. <P>, <DIV>, <H1> etc) use innerHTML to change them from code

What would be the effect of running this function?

```
function doStuff() {
   var target = document.getElementById("kno");
   target.innerHTML = "k12345";
   target.value = "k54321";
   var target2 = document.querySelectorAll("p")[1];
   target2.innerHTML = "Kaboom";
}

<html>
  …head element assumed here…
  <body>
    <p>K-Number: <input id="kno" size="10"/></p>
    <p>Name: <input id="name" size="50"/></p>
  </body>
</html>
```



(select all options that are valid!)

1. The first input element would be filled in with the text **k12345**
2. The second input element would be filled in with the text **k12345**
3. The first input element would be filled in with the text **k12345**
4. The second input element would be filled in with the text **k12345**
5. The first input element would be filled in with the text **k54321**
6. The second input element would be filled in with the text **k54321**
7. The first input element would be filled in with the text **k54321**
8. The second input element would be filled in with the text **k12345**
9. The second paragraph would be wiped out and replaced with the text **Kaboom**
10. Something else

Summary

- Javascript is primarily used as a means of adding interactivity to web pages
- In this context, Javascript programs do not “print” output to a “screen”, or get input (directly) from a user
  - They interact with web pages
  - They can read and modify HTML elements on these web pages
  - More accurately, they can read and modify elements in the document object model or DOM
- The DOM is the model of the page that the web browser builds up from the HTML
- Javascript programs “display” output by modifying elements in the DOM
- Javascript programs get “input” from a user by reading elements from the DOM – usually form elements like <input> or <textarea>