College of Information Studies

University of Maryland Hornbake Library Building College Park, MD 20742-4345

Objects and Events

Week 5

INFM 603

Muddiest Points

- Commonly used functions
- .getElementById
- Recursion
- Hashing

Programming in Four Parts

- Structured Programming
- Modular Programming
- Data Structures
- ➢ Object-Oriented Programming

Key Ideas

- Protect the programmer from themselves
  - Model actions and attributes together
- Object
  - Encapsulation of methods and data structures
- Class
  - “Blueprint” for an object
  - Must be “instantiated” using a “constructor”

Objects: Methods

- It’s just a collection of properties!
- Objects can have functions also! (called methods)

```
var fido = {
  name: "Fido",
  weight: 40,
  breed: "Mixed",
  loves: ["walks", "fetching balls"],
  bark: function() {
    alert("Woof woof!");
  }
};
```

Constructor

```
function Dog(name, breed, weight) {
  this.name = name;
  this.breed = breed;
  this.weight = weight;
  this.bark = function() {
    if (this.weight > 25) {
      alert(this.name + " says Woof!");
    } else {
      alert(this.name + " says Yip!");
    }
  };
}
```

Using Constructors

- Invoke constructors using “new”:

```
var fido = new Dog("Fido", "Mixed", 38);
var tiny = new Dog("Tiny", "Chawalla", 8);
var clifford = new Dog("Clifford", "Bloodhound", 65);

fido.bark();
tiny.bark();
clifford.bark();
```

Properties and Methods

- Access object properties using the “dot” notation

```
var w = fido.weight;
fido.breed = "Yellow Lab";
```

- Invoke an object’s method using the dot notation:

```
fido.bark();
```

- this.bark()

Some Conventions

- CapsInitial camel case is used for a a **class**
- lowerCaseInitial camel case is used for a
  - Variable (if not followed by parameters)
  - Method (if followed by parameters)
- An **object** can be assigned to a variable

Object Instantiation

- **var n = new Array(5);**
  - Creates an Array object using the Array class constructor (and specifying 5 elements)
- **var student = new Student(13205, “George”);**
  - Creates a Student object using the Student class constructor (and specifying the student id and name)
  - Note that the variable name need not be different from (or the same as) the class name

Formalizing Object Interfaces

- **status = student.setHeightInches(74);**
  - Invokes the **setHeightInches()** method for the object that is stored in the variable **student** and passes it **74** as a parameter; **status**=true indicates success

- **feet = student.getHeightFeet();**
  - Invokes the **getHeightFeet()** method for the object that is stored in the variable **student** and then sets the variable **feet** to hold that result (in this case, 6); feet<0 indicates failure

Class Definition (private variable)

```
function Student(studentID, name) {
  var totalInches = -1;   // private variable

  // private method
  function inchesToFeet(i) {
    return Math.floor(i/12);
  }

  // public methods
  this.setHeightInches = function(n) {
    if ((n>0) && (n<100)) {
      totalInches = n;
      return true;
    } else {
      return false;
    }
  }
  this.getHeightFeet = function() {
    if (totalInches>0) {
      return inchesToFeet(totalInches);
    } else {
      return -1;
    }
  }
}
```

```
var student = new Student(13205, "George");
alert(student.setHeightInches(74));
alert(student.getHeightFeet());
alert(student.totalInches);
```

Class Definition (public variable)

```
function Student(studentID, name) {
  this.totalInches = -1;  // public variable

  // private method
  function inchesToFeet(i) {
    return Math.floor(i/12);
  }

  // public methods
  this.setHeightInches = function(n) {
    if ((n>0) && (n<100)) {
      this.totalInches = n;
      return true;
    } else {
      return false;
    }
  }
  this.getHeightFeet = function() {
    if (this.totalInches>0) {
      return inchesToFeet(this.totalInches);
    } else {
      return -1;
    }
  }
}
```

```
var student = new Student(13205, "George");
alert(student.setHeightInches(74));
alert(student.getHeightFeet());
alert(student.totalInches);
```

Alternate Method Definition (private variables)

```
var student = new Student(13205, "George");
alert(student.setHeightInches(74));
alert(student.getHeightFeet());
alert(student.feet);
```

```
function Student(studentID, name) {
  var inches = -1;  // private variable
  var feet   = -1;    // private variable

  // private method
  function inchesToFeet(i) {
    return Math.floor(i/12);
  }

  // public methods
  this.setHeightInches = function(n) {
    if ((n>0) && (n<100)) {
      feet = inchesToFeet(n);
      inches = n-(feet*12);
      return true;
    } else {
      return false;
    }
  }
  this.getHeightFeet = function() {
    if ((feet>0) || (inches>0)) {
      return feet;
    } else {
      return -1;
    }
  }
}
```

Alternative Notation

```
<!doctype html>
<html lang="en">
<body>


<script>
var student={name:"Ann", lastname:"Smith", age:28,
    program:["MIM","Data Analytics"]};
var student2={name:”Rob", lastname:“Rogers", age:32,
    program:["MLS",“UX"]};
document.writeln(" “ + student.program[1] + " “ +
    student2.program[1]);
</script>
</body>
</html>
```

Everything is an Object

- var b = new Boolean(true);
- var n = new Number(3.15);
- var n = new Number(3); // same as 3.00
- var a = new Array(5);

String Objects

- (Conceptually) an array of Unicode characters with some interfaces
  - var s = “Mr. Spock”
  - s.toLowerCase is “mr. spock”
  - s.substr(3,4) is “ Spo”
  - s.indexOf(“k”) is 8
  - s.split(“ ”) is [“Mr.”, “Spock”]
  - s.link(http://bit.ly.CUjV) is Mr. Spock
  - s + “Captain Kirk” is “Mr. SpockCaptainKirk”

Some Handy Methods

- document
  - document.writeln(“Test!”);
  - var e=document.getElementById(“goButton”);
  - document.cookie=“message=saveme”;
  - var c=document.cookie.split(“=“)[1];

- window
  - window.prompt(“Input please”);
  - var w=window.open(“”, “New Window”, “”);

Some Math Methods

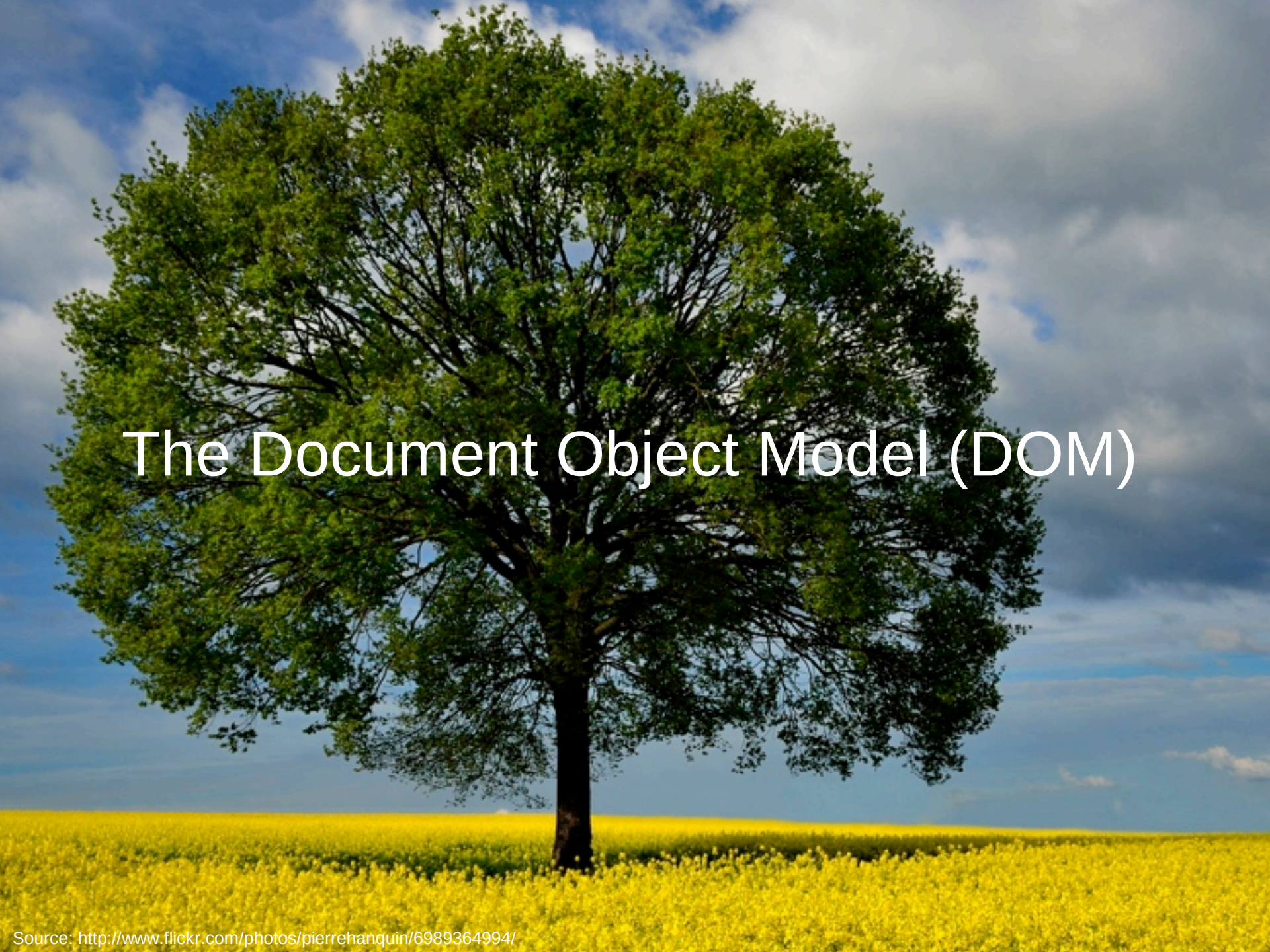
- Math.abs() – Absolute value
  - Example: Math.abs(-10)
- Math.max() – Maximum of two values
  - Example: Math.max(10, 20)
- Math.sqrt() – Square root
  - Example: Math.sqrt(4)
- Math.random() – Random value between 0 and less than 1
  - Example: Math.random()
- Constants
  - Math.PI – Mathematical constant pi

Why Use Objects?

- A way of thinking about programming
  - Objects are nouns, methods are verbs
- A form of defensive programming
  - Hides private variables and methods
  - Allows you to make behaviors explicit
- Represent complex data structures
  - Airplanes have pilots, fuel, and destinations

Design Exercise

- Design a **class** for email messages
- **Private** internal representation should include:
  - Header (date, time, sender, recipients, subject, …)
  - Body
  - Attachments (which may be other emails!)
- **Public** interfaces should include
  - Message creation
  - Access to specific header fields, the body, and specific attachments

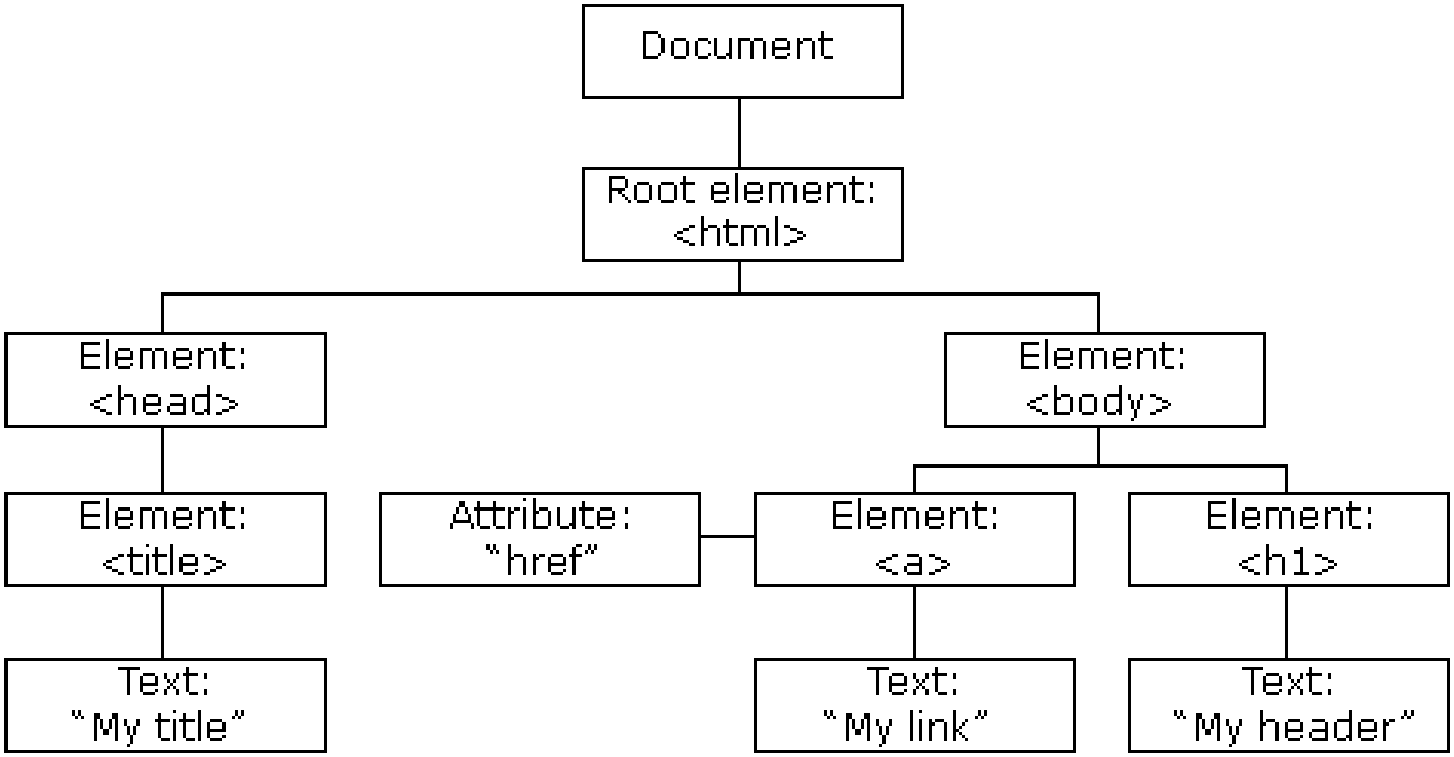


Source: http://www.flickr.com/photos/pierrehanquin/6989364994/

Source: http://www.flickr.com/photos/pierrehanquin/6989364994/

Source: http://www.flickr.com/photos/pierrehanquin/6989364994/

A DOM Tree

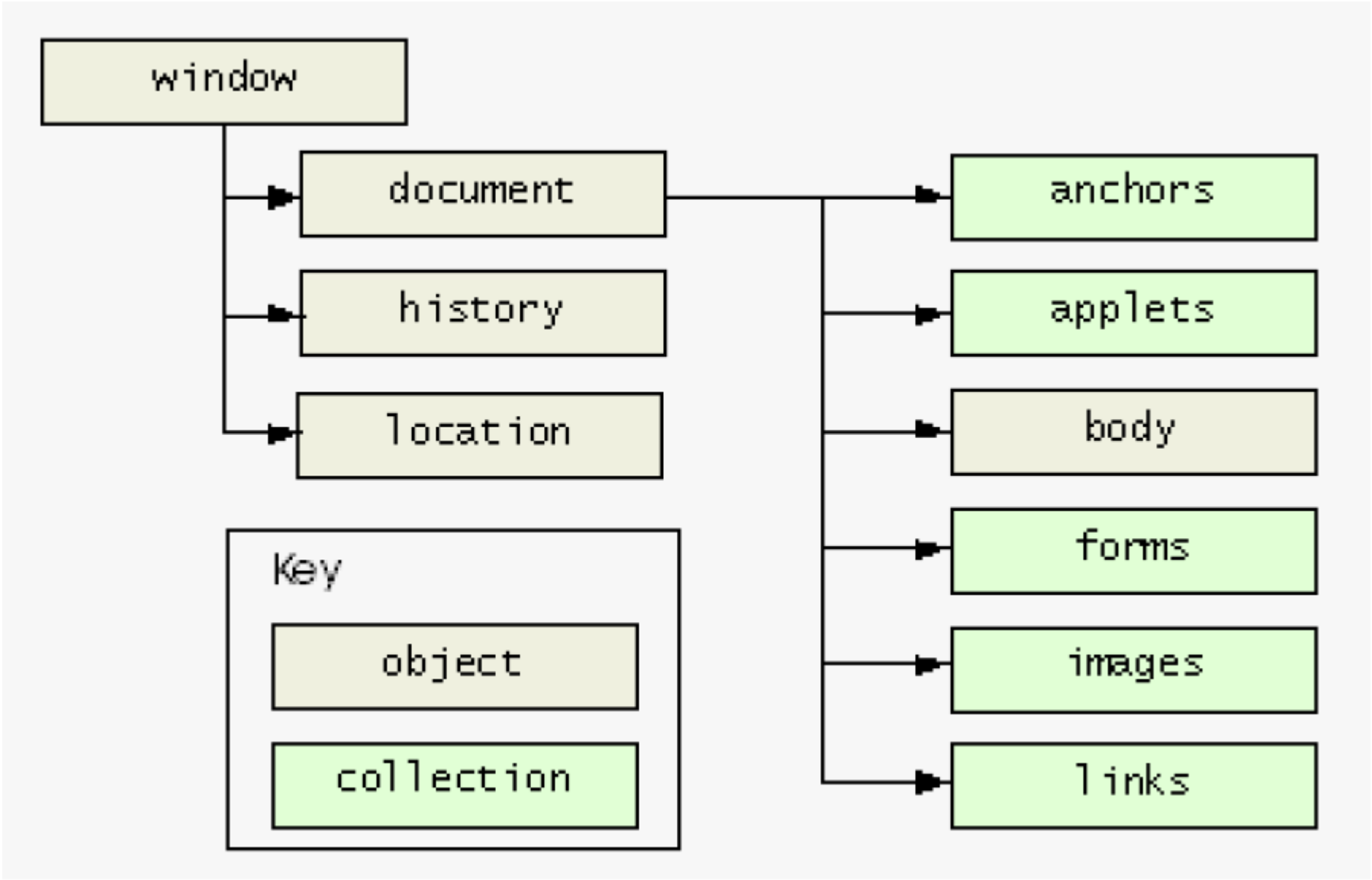

Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
"href"
Element:
<a>
Element:
<h1>
Text:
"My title"
Text:
"My link"
Text:
"My header"

Document Object Model

- The DOM view of an HTML page has:
  - OBJECTS: HTML elements
  - ATRIBUTES of the elements
  - METHODS to access elements
  - EVENTS to control elements

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html lang="en" dir="ltr" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
  <head>
  <body id="homePage">
    <div id="page-wrapper">
      <div id="page">
        <div id="header" class="clearfix">
          <p id="umLogo">
          <p id="iSchoolLogo">
          <div id="headerExtras">
          <div id="menu">
            <div class="region region-sidebar-first column sidebar">
              <div class="section">
                <div id="block-user-1" class="block block-user region-odd odd region-count-1 count-1">
                  <h2 class="title">Navigation</h2>
                  <div class="content">
                    <ul class="menu">
                      <li class="collapsed first">
                        <a title="Prospective Students" href="/content/prospective-students">Prospective Students</a>
                      </li>
                      <li class="collapsed">
                      <li class="collapsed">
                      <li class="collapsed">
                      <li class="collapsed">
                      <li class="collapsed">
                      <li class="collapsed last">
                    </ul>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
        <div id="content">
        <div id="footer">
      </div>
    </div>
  </body>
</html>
```

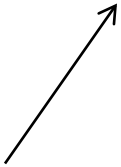
Getting to DOM Elements



Asking the DOM to “do stuff”

the *method* is want you want the document “to do”, usually a verb phrase


```
document.method(“argument”);
```



document represents the entire page and contains the DOM

arguments are additional details that you specify

```
document.writeln(“Hello World”);
```

Access to DOM Elements

- Find a single element

  element = document.getElementById(“input2”);
- Find multiple elements

  list = document.getElementsByTagName(input);

  list = document.getElementsByName(“myInput”);
- Move up in the DOM tree
  - element1 = element2.parentNode;
- Move down in the DOM tree
  - list = element1.childNodes

DOM: Selecting an Element/Node

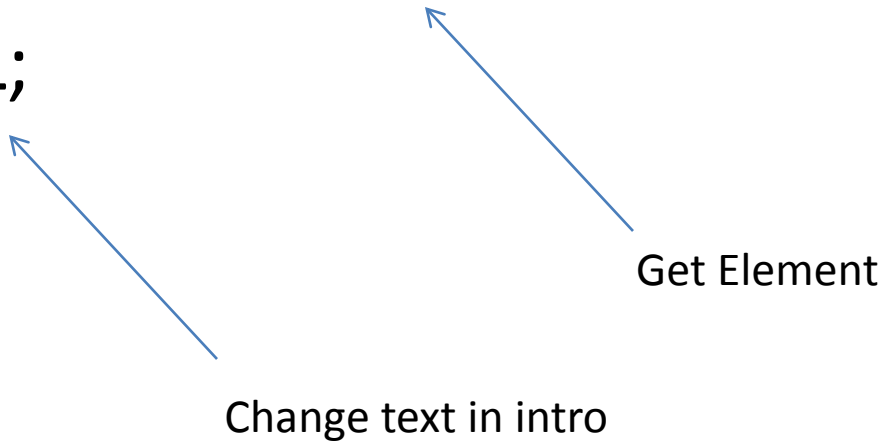
- Get Element by ID:

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

var text= document.getElementById("intro");
text.innerHTML;
```

DOM: Selecting an Element/Node

- Get Element by ID:

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

var text= document.getElementById("intro");
text.innerHTML;
```



DOM: Selecting an Element/Node

- Get Element by Tag:

  Hello World!

  Hello World!

  document.getElementsByTagName(“p");

  → returns collection of “p”s in HTML (array with intro1 and intro2)

```
<!DOCTYPE html>
<html>
<body>

<p> Line Zero</p>
<p id="main"> Line One </p>
<p> Line Two </p>


<script>
var x=document.getElementById("main");
var y=document.getElementsByTagName("p");
document.write("Get element by id " + x.innerHTML +"<br>");
for(c=0;c<3;c++){
  document.write("Get element by tag " + y[c].innerHTML+"<br>");
}
</script>

</body>
</html>
```

DOM: Manipulating Elements/Nodes

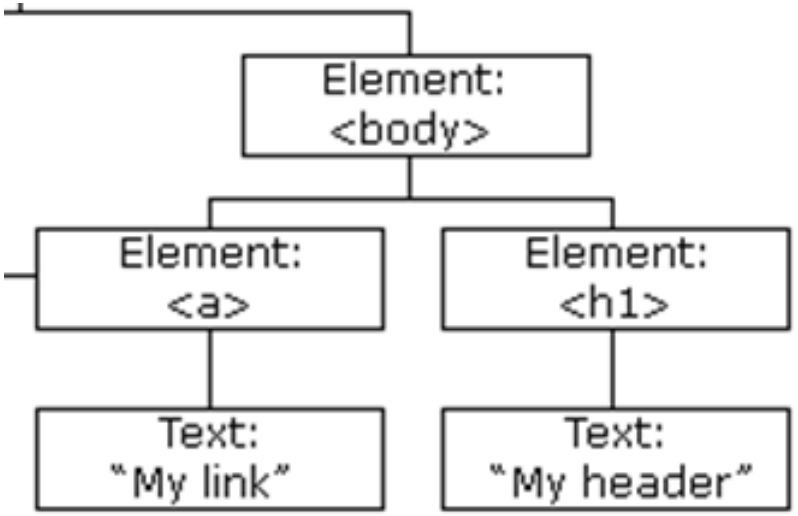
- Modify the document
  - document.write(“text”)

- Modify a node:
  - Content:
    - document.getElementById(“p¨).innerHTML=“Hello”;
  - Attribute:
    - document.getElementById(“image”).src=“newimage”;
  - Style:
    - document.getElementById(“text¨).style.color=blue;

```
<!DOCTYPE html>
<html>
<body>

<img id="image" src=“coffee.gif">

<script>
    document.getElementById("image").src=“tea.jpg";
</script>

</body>
</html>
```

DOM: Manipulating Children Nodes

- Accessing child nodes:
  - element.childNodes();




```
document.getElementById("list").childNodes[1].innerHTML =
"new item";
```

Example

```
<p id="p1">Line One</p>
<p id="p2">Line two</p>
```

Write a JavaScript program that changes line two to font Arial and color blue

```
<!DOCTYPE html>
<html>
<body>

<p id="p1">Line One</p>
<p id="p2">Line Two</p>

<script>
  document.getElementById("p2").style.color="blue";
  document.getElementById("p2").style.fontFamily="Arial";
</script>

<p>The paragraph above was changed by a script.</p>

</body>
</html>
```

DOM: Create Elements/Nodes

- Create a new node:

```
var p = document.createElement("p");
p.innerHTML = "here is some new text.";
document.getElementById("div1").appendChild(p);
```

- Create item for a list:

```
var newItem = document.createElement("li");
newItem.innerHTML = "new list item";
document.getElementById("list").appendChild(newItem);
```

Example

```
<!DOCTYPE html>
<html>
<body>

<div id="div1">
<p id="p1">Line One</p>
<p id="p2“>Line Two</p>
</div>

<script>
  var para=document.createElement("p");
  para.innerHTML=“Line Three”;

  var element=document.getElementById("div1");
  element.appendChild(para);
</script>

</body>
</html>
```

DOM: Eliminate Elements/Nodes

- Select node to eliminate and remove it with removeChild:

```
var list = document.getElementById("list");
var listItem = list.childNodes[1];
list.removeChild(listItem);
```

```
<!DOCTYPE HTML>
<html>
<body>
<div id="myTable" border="1"></div>
<script type="text/javascript">
 var con = document.getElementById("myTable");
 var tab = document.createElement("table");
 tab.setAttribute("border", "1");

 var r,c,td,tr;
 for(r=1;r<5;r++){
  tr = document.createElement("tr");
  for (c=1;c<5;c++){

   td = document.createElement("td");
   td.innerHTML="Row"+r+"Col"+c;
   tr.appendChild(td);
  }

  tab.appendChild(tr);

 }
 con.appendChild(tab);
</script>
</body>
</html>
```

Row1Col1	Row1Col2	Row1Col3	Row1Col4
Row2Col1	Row2Col2	Row2Col3	Row2Col4
Row3Col1	Row3Col2	Row3Col3	Row3Col4
Row4Col1	Row4Col2	Row4Col3	Row4Col4

Documentation Tips

- Reflect your pseudocode in your code
  - Use meaningful variable names
  - Use functions for abstractable concepts
    - And name those functions well
  - Use comments to fill remaining gaps
- Add a comment to identify each revision
  - Give author, date, nature of the change
- Waste space effectively
  - Use indentation and blank lines to guide the eye

Using JavaScript with Forms

HTML:

```
<form name="input" action=" ">
   Please enter a number:
   <input size="10" value=" " name="number"/>
</form>
<form name="output" action=" ">
   The sum of all numbers up to the number above is
   <input size="10" value=" " name="number" readonly="true"/>
</form>
```

Reads in a value from the first form
(*eval* method turns it into a number)

JavaScript:

```
var num = eval(document.input.number.value);
document.output.number.value = 10;
```

Changes the value in the second form

HTML Form Element Types

- Textarea (multiple lines)
- Input
  - Text (single line)
  - Password (like text, but masked)
  - Hidden (like text, but not displayed at all)
  - Button
  - Checkbox (multiple selection)
  - Radio (single selection)
- Select (dropdown list)

see `http://www.w3schools.com/html/html_forms.asp` for examples

Linking Forms to Functions

- Events:
  - Actions that users perform

- An “event handler” is triggered by an event
  - onClick: the user clicked on the item
  - onMouseover: the mouse moved onto the item
  - onMouseout: the mouse moved off of the item

Referring to Form Content

```
<form name=years>
  <b>Please enter your age</b>
  <input type=text name=box />
</form>
…
var age = document.years.box.value;
```

```
<form action = " ">
   <p>Enter integer search key<br />
   <input id = "inputVal" type = "text" />
</form>
…
var inputVal   = document.getElementById("inputVal");
var searchKey  = inputVal.value;
```

Events

- Events allow an HTML webpage to react to a users' behavior (event handler)
  - Click of a mouse
  - Mouse over an element
  - User strokes a key
  - Image has been loaded
  - Input field is changed (forms)

More complex events…Forms

- Click on button to select an action
- Text box for entering a line of text
- Radio buttons for making one selection among a group of options
- Check boxes for selecting or deselecting a single, independent option
- Lists of things to select one

Push Button

```
<!DOCTYPE html>
<html>
<body>

<form>
   <input type="button" value="This is a button">
</form>

</body>
</html>
```

Text Box

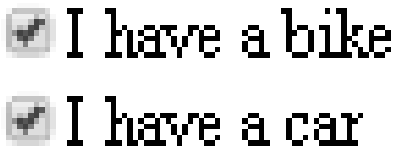
```
<!DOCTYPE html>
<html>
<body>

<textarea rows="3" cols="30">
    This is a text area.
</textarea>

</body>
</html>
```

Radio Button

```
<!DOCTYPE html>
<html>
<body>

<form>
  <input type="radio" name=“fruit" value="apples">Apples<br>
  <input type="radio" name=“fruit" value="oranges">Oranges<br>
  <input type="submit" value="Send">
</form>

</body>
</html>
```

Check Box

```
<!DOCTYPE html>
<html>
<body>

<form >
  <input type="checkbox" name="vehicle" value="Bike">I have a
    bike<br>
  <input type="checkbox" name="vehicle" value="Car">I have a car
</form>

</body>
</html>
```



List

```
<!DOCTYPE html>
<html>
<body>

<form >
  <select name="fruits">
    <option value="oranges">Oranges</option>
    <option value="apples">Apples</option>
    <option value="lemons">Lemons</option>
    <option value="bananas">Bananas</option>
  </select>
</form>

</body>
</html>
```

Handling Events…JavaScript!

- JavaScript is responsible for “doing things” when buttons are clicked

```
<input type="button" value="This is a button">

<input type="button" value="This is a button"
onclick="myFunction()">
```

Handling Events…JavaScript!

```
<input type="button" value="This is a button"
onclick="myJSFunction()">

<script>
function myJSFunction()
{ do something}
</script>
```

Example with Push Button

Try this

```
<!DOCTYPE html>
<html>
<body>
<script>
function myJSFunction()
{
document.getElementById("demo").innerHTML="Hello World";
}
</script>
<form>
   <input type="button" onclick="myJSFunction()" value="This is a button">
</form>

<p id="demo"></p>
</body>
</html>
```

Text Box: access text

```
<p><input type="text"
   id="textInput"><br/></p>


var textInput =
   document.getElementById("textInput");
var textValue = textInput.value;
alert(textValue);
```

Text Box

```
<!doctype html>
<html lang="en">
<body
<title>Forms</title>
<meta charset="utf-8">

<script>
function handleClick() {
  var textInput = document.getElementById("textInput");
  var textValue = textInput.value;
  alert("The input text is: " + textValue);
}
</script>
</head>
<body>

<form>
<p><input type="text" id="textInput" size="40"><br/></p>

<p><input onclick="handleClick()" type="button" id="button" value="Click me!"></p>
</form>

</body>
</html>
```

USE THE DOM

Radio Button

```
<input id="radioone" type="radio" name=“select"
   value=“one“>One
<input id="radiotwo" type="radio" name=“select"
   value=“two“>Two</p>

var onechecked =
   document.getElementById("radioone").checked;
alert(“button one checked: " + onechecked);
```

```
<!doctype html>
<html lang="en">
<body
<title>Forms</title>
<meta charset="utf-8">

<script>
function handleClick() {
  var onechecked = document.getElementById("radioone").checked;
  var twochecked = document.getElementById("radiotwo").checked;
  alert("one checked: " +  onechecked+" two checked "+twochecked);

}
</script>
</head>
<body>

<form>
   <input id="radioone" type="radio" name=“select" value=“one“>One
   <input id="radiotwo" type="radio" name=“select" value=“two“>Two</p>

   <p><input onclick="handleClick()" type="button" id="button" value="Click me!"></p>
</form>

</body>
</html>
```

Check Box

```
<input id="checkDog" type="checkbox"
   name="pet" value="dog">I have a dog<br>
```

```
var hasDog =
   document.getElementById("checkDog").checked;
```

```
<!doctype html>
<html lang="en">
<body
<title>Forms</title>
<meta charset="utf-8">

<script>
function handleClick() {
  var hasDog = document.getElementById("checkDog").checked;
  var hasCat = document.getElementById("checkCat").checked;
  alert("dog? " + hasDog+ " cat? " +  hasCat);

}
</script>
</head>
<body>

<form>
<p>
<input id="checkDog" type="checkbox" name="pet" value="dog">I have a dog<br>
<input id="checkCat" type="checkbox" name="pet" value="cat">I have a cat
</p>

<p><input onclick="handleClick()" type="button" id="button" value="Click me!"></p>
</form>

</body>
</html>
```

List

```
<select id="selectInput">
  <option value="oranges">Oranges</option>
</select>

<script>
var selectInput =
    document.getElementById("selectInput");
var selectedIndex = selectInput.selectedIndex;
 var selectedValue =
    selectInput.options[selectedIndex].value;
```

```
<!doctype html>
<html lang="en">
<body
<title>Forms</title>
<meta charset="utf-8">

<script>
function handleClick() {
  var selectInput = document.getElementById("selectInput");
  var selectedIndex = selectInput.selectedIndex;
  var selectedValue = selectInput.options[selectedIndex].value;
  alert("Which fruit do you prefer? " + selectedValue);

}
</script>
</head>
<body>

<form>
<p>
<select id="selectInput">
  <option value="oranges">Oranges</option>
  <option value="lemons">Lemons </option>
  <option value="apples">Apples</option>
  <option value="bananas">Bananas</option>
</select>
</p>

<p><input onclick="handleClick()" type="button" id="button" value="Click me!"></p>
</form>

</body>
</html>
```

Term Project

- Option 1:
  - Drupal, Joomla, …
- Option 2: Programming project
  - Javascript, PHP, Java, C, Python, Ruby, …
- Four “deliverables”
  - P1 requests teaming preferences
    - Assigned based on skills diversity
  - P2 is project plan
  - P3 is project design
  - P4 is presentation slides (from class presentation)

Before You Go

On a sheet of paper, answer the following (ungraded) question (no names, please):

What was the muddiest point in today's class?