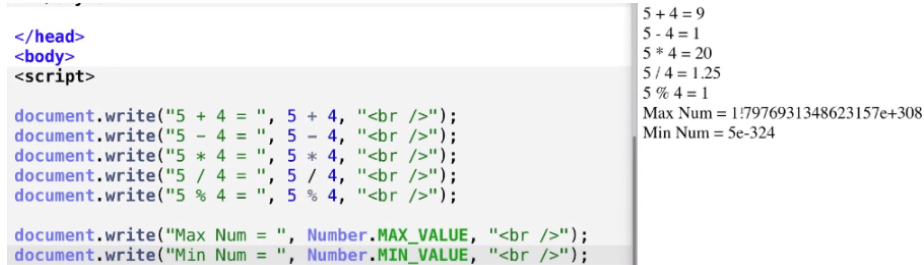


## JavaScript

### Numbers:

- (a) Here are some basic number functions:



```

</head>
<body>
<script>

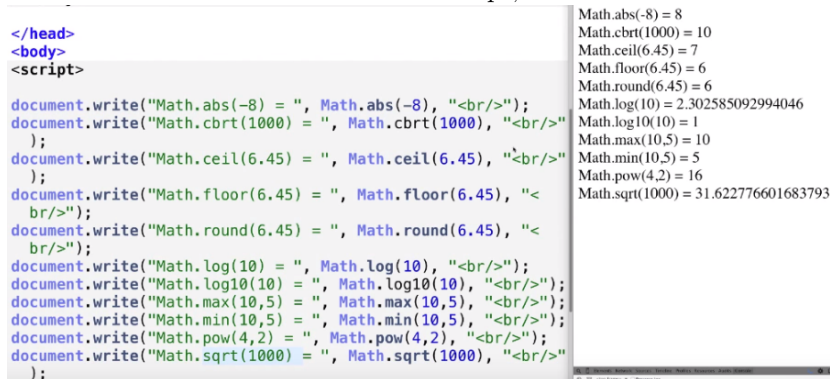
document.write("5 + 4 = ", 5 + 4, "<br />");
document.write("5 - 4 = ", 5 - 4, "<br />");
document.write("5 * 4 = ", 5 * 4, "<br />");
document.write("5 / 4 = ", 5 / 4, "<br />");
document.write("5 % 4 = ", 5 % 4, "<br />");

document.write("Max Num = ", Number.MAX_VALUE, "<br />");
document.write("Min Num = ", Number.MIN_VALUE, "<br />");

```

5 + 4 = 9  
 5 - 4 = 1  
 5 \* 4 = 20  
 5 / 4 = 1.25  
 5 % 4 = 1  
 Max Num = 1.7976931348623157e+308  
 Min Num = 5e-324

- (b) Numbers can only be precise with up to 16 digits of precision.
- (c) To round a decimal to 'int' digits we would use: 'toFixed(int)'
- (d) '++x' will increment the number 'x' by 1 and then apply it to a function, while 'x++' does these in the opposite order. 'x+=1' does the same thing as the former, but it can be done with different numbers and operators '-=', '\*=', '/='.
- (e) PEMDAS applies in Javascript, so parentheses must be used in some cases of calculations.
- (f) Here are some Math Libraries in Javascript, Math.E and Math.PI are not included:



```

</head>
<body>
<script>

document.write("Math.abs(-8) = ", Math.abs(-8), "<br />");
document.write("Math.cbrt(1000) = ", Math.cbrt(1000), "<br />");
);
document.write("Math.ceil(6.45) = ", Math.ceil(6.45), "<br />");
);
document.write("Math.floor(6.45) = ", Math.floor(6.45), "<br />");
document.write("Math.round(6.45) = ", Math.round(6.45), "<br />");
document.write("Math.log(10) = ", Math.log(10), "<br />");
document.write("Math.log10(10) = ", Math.log10(10), "<br />");
document.write("Math.max(10,5) = ", Math.max(10,5), "<br />");
document.write("Math.min(10,5) = ", Math.min(10,5), "<br />");
document.write("Math.pow(4,2) = ", Math.pow(4,2), "<br />");
document.write("Math.sqrt(1000) = ", Math.sqrt(1000), "<br />");
);

```

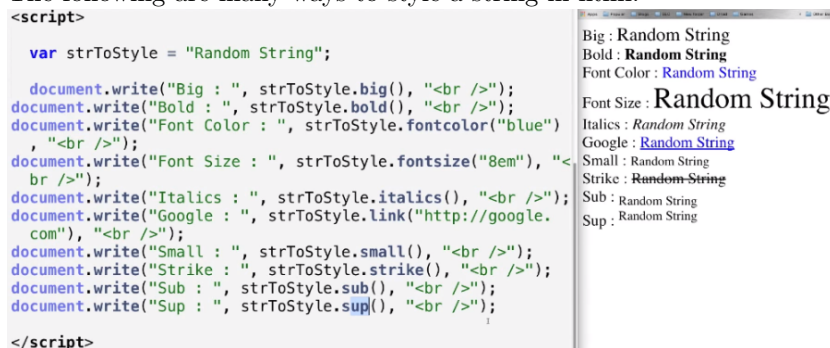
Math.abs(-8) = 8  
 Math.cbrt(1000) = 10  
 Math.ceil(6.45) = 7  
 Math.floor(6.45) = 6  
 Math.round(6.45) = 6  
 Math.log(10) = 2.302585092994046  
 Math.log10(10) = 1  
 Math.max(10,5) = 10  
 Math.min(10,5) = 5  
 Math.pow(4,2) = 16  
 Math.sqrt(1000) = 31.622776601683793

- (g) To generate a random number between 1-10, we would use: 'Math.floor((Math.random()\*10)+1)'
- (h) To convert a 'string' into a number we can use 'Number("string")'. Similarly we can use 'parseInt("5")' to convert '5' into an integer. Similarly we can use 'parseFloat("3.14")' to convert '3.14' into a float.

### Strings:

- (a) If we are given the string 'randStr', we can find its length by using: 'randStr.length'

- (b) Since strings are just arrays of characters, we can find the index that the first character of a substring 'subStr' of 'randStr' by using 'randStr.indexOf(subStr)'. Similarly if we wanted to find the word 'hello' we would do 'randStr.indexOf("hello")'.
- (c) We can select a portion of a string from index 'int1' to 'int2' by using: 'randStr.slice(int1, int2)'. We can instead select the portion from 'int1' to the end of the string by using: 'randStr.slice(int1)'. If we wanted a subsection of length 'int3' that started at 'int1' we could use: 'randStr.substr(int1, int3)'.
- (d) We can replace the substring 'subStr' of 'randStr' with 'newSubStr' by using: 'randStr.replace(subStr, newSubStr)'.
- (e) We can select a character at index 'int' of 'randStr' by using: 'randStr.charAt(int)'.
- (f) We can split a string into an array based off of certain substrings. Normally this would be used for single characters like spaces. Splitting 'randStr' by spaces would be done by using: 'randStr.split(" ")'
- (g) We can trim all whitespace (spaces before and after other characters in string) by using: 'randStr.trim()'.
- (h) We can change all characters of 'randStr' to uppercase by using: 'randStr.toUpperCase()', and we can do the same for lowercase by using: 'randStr.toLowerCase()'.
- (i) The following are many ways to style a string in html:



### Conditionals:

- (a) Most operators are the same as in Java/C etc... There is a difference in the operator '=='. '==' compares the value of the two variables beside it. The operator that is the same as the Java '==' is '===', it compares the value of the two variables beside it but also compares the variable type. The same can be said about '!=' and '!=='.
- (b) An important operator to learn is the ternary operator, denoted by a '?'. We could create the string 'canIVote' in the following way: 'var canIVote = (age<18) ? "yes" : "no";'. What this statement means is that if the variable 'age' is greater than 18 then 'canIVote' would be set to 'yes', but if it isn't then it would be set to 'no'; the left side is ifTrue, the right is ifFalse.
- (c) All conditional statements (for-loops, if-statements, etc...) are the same except we never learned the switch statement:

```

var age = 5;
switch(age){ //the switch checks different values for 'age'
  case 5 : //if (age == 5) DO, the space before the ':' may be important
  case 6 : //OR if (age == 6) DO
    document.write("Go to Kindergarten" + "<br />"); //display string and add 'enter'
    break; //exit switch, don't check any other cases
  case 7 :
    document.write("Go to 1st grade" + "<br />");
    break;
  default : //always do this unless there was a 'break'
    document.write("Go somewhere else" + "<br />");
}

```

- (d) There is one for-loop type that doesn't apply to Java. If we had an array with keys such as 'var customer = name: "Numa", city: "Montreal", age: '20';', we can iterate through the keys and write their contents by doing 'for(k in customer){ document.write(customer[k] + "<br />");}'.

#### Arrays:

- (a) The following displays the majority of array utility:

```

var tomSmith = ["Tom Smith", "123 Main", 120.50]; //initialize array
document.write("1st index ", tomSmith[0], "<br />"); //display name
tomSmith[3]="tSmith@aol.com"; //append an email to the array
tomSmith.splice(2,1,"Pittsburgh", "PA"); //choose index 3 and the next 1 indices (so just
index 3) and replace them with the two indices 'Pittsburgh' and 'PA'
tomSmith.splice(4,1); //this is the same as deleting index 5
tomString1 = tomSmith.valueOf(); //converts the array into a string
tomString2 = tomSmith.join(", "); //converts the array into a string seperated by commas and
spaces
delete tomSmith[3]; //deletes index 4
tomSmith.sort(); //sorts array by alphabetical order
var numbers=[4,3,9,1,20,43]; //random array of numbers
numbers.sort(function(x,y){ return x - y }); //sorts 'numbers' by ascending order
numbers.sort(function(x,y){ return y - x }); //sorts 'numbers' by descending order
numbers.toString(); //convert an array into a string [again...]
var combinedArray = numbers.concat(tomSmith); //appends 'tomSmith' array to 'numbers' array
tomSmith.pop(); //remove the last index of 'tomSmith'
tomSmith.push("555-1212", "US"); //append things to the end of 'tomSmith'
tomSmith.shift(); //delete the first index of 'tomSmith'
tomSmith.unshift("Tom Smith"); //add items to the beginning of 'tomSmith'

```

- (b) The length of an array can be accessed with the '.length' function.

#### Functions:

- (a) Functions are the methods of Javascript. The format is much more simple:

```

function times2(num){ //no return declared
    return num*2; //don't need to specify return statement in function name
}
function times3(num){
    return num*3;
}
function multiply(func, num){
    return func(num); //we can call functions inside of eachother
}
document.write("2 * 15 = ", multiply(times2, 15), "<br />");
document.write("3 * 15 = ", multiply(times3, 15), "<br />");
function getSum(){ //no arguments have been declared
    var sum=0;
    for(i=0; i<arguments.length; i++){
        sum += arguments[i]; //keep note of arguments[] array
    }
    return sum;
}
document.write("Sum = ", getSum(1,2,3,4,5,6), "<br />"); //as we can see from this, even
though our function did not declare any arguments, we can still access arguments in a called
function by using the arguments[] array

```

### Event Handling:

- (a) The following image has HTML in it. The first portion of the HTML has its own events in it, while the second part defines some image/buttons/inputs that can have events defined around them:

```

function openAlert(mess){ //accepts some variable named 'mess'
    alert(mess); //displays the 'mess' as an alert on browser
}

<a href="JavaScript:void(0)" onClick="alert('Hello');">SayHello</a><br /> //This will alert
the message 'Hello' when 'Say Hello' is clicked. 'Javascript:void(0)' stops the page from
changing on click.
<a href="JavaScript:void(0)" onClick="openAlert('Hello');">SayHello</a><br /> //Same effect
except that the function 'openAlert' will be called.

<a href="JavaScript:void(0)" //reference is void
onmouseover="this.style.color='red'" //text turns red when cursor moves over it
onmouseout="this.style.color='blue'" //text turns blue when cursor moves off of it
ondblclick="this.text='You Double clicked me'" //text says 'you Double clicked me' when
double clicked
onmousedown="this.text='Don\'t Press too hard'" //text says 'Don't Press too hard' when the
mouse is pressed down.; the \' is to help the code differentiate between the two text types
onmouseup="this.text='Thank you'" //Make me Red</a><br /> //when mouse is no longer clicking
down on the text, the text says 'Thank you'

<input type="text" id="randInput" //textbox named 'randInput'; textbox can have text inputted
on Change="var dataEntered=document.getElementById('randInput').value;" //creates
'dataEntered' with 'randInput''s value
+ "alert('User Typed' + dataEntered)" //alerts of 'dataEntered'

<form action="#" id="sampForm"> //see what happens with key pressing, goes no where when
keys are pressed
    <input id='charInput' type='text'> //input box where text is inputted
    <p id = "keyData">Key Data Here</p><br /> //paragraph where typed things are displayed,
    does not need break tag
    <input type = "submit" value="Submit"> //input box that displays submit button
    <input type="reset" value="Reset"> //input box that resets (refreshes) page
</form><br />
 //images is imported from the 'ntt-logo.png' source link
<button id="logoButton">Get Logo</button><br /> //a button that will bring up the image,
does not need break tag
<input id="mouseInput" type="text" size='30'><br /> //mouse input
Mouse X: <input type="text" id='mouseX'><br /> //mouse x-coordinate
Mouse Y: <input type="text" id="mouseY"><br /> //mouse y-coordinate

<button id="clearInputs">Clear Inputs</button><br /> //creates a button, does not need break
tag

```

Here is the Javascript that creates events to work on the image/buttons/inputs:

```

<script>
function getChar(event){ //function accepting an event
    if(event.which!=0 && event.charCode!=0){ //'event.which' & 'event.charCode' return the
        inputs as well
        return String.fromCharCode(event.which); //returns same as 'event.keyCode'
    }
    else{ //occurs upon unknown event input
        return null;
    }
}

document.getElementById('charInput').onkeypress=function(event){ //the id 'charInput' is the
    textbox waiting for text input, when a key is pressed, the key pressed 'event' will be used
    var char = getChar(event); //'char' will be this character event
    if(!char) return false; //the key was null, the last case of 'getChar' event
    document.getElementById('keyData').innerHTML=char+" was clicked"; //the paragraph
    'keyData' will have its 'innerHTML' changed to the character 'char' and ' was clicked'
    return true;
}

document.getElementById('charInput').onfocus=function(event){ //waits for the text box to be
    clicked, clicking is still a character input
    document.getElementById('keyData').innerHTML="Input Gained Focus"; //the paragraph
    'keyData' will have its 'innerHTML' changed to 'Input Gained Focus'
}

document.getElementById('charInput').onselect=function(event){ //waits for the text box to
    have its text selected
    document.getElementById('keyData').innerHTML="Text Selected"; //the paragraph
    'keyData' will have its 'innerHTML' changed to 'Text Selected'
}

document.getElementById('logoButton').onclick=function(event){ //when 'LogoButton' is
    clicked, the function occurs
    document.getElementById('logo').className="show"; //(this is some CSS) the className of
    the image is changed to 'show' so that the image appears upon click
}

document.getElementById('logo').onclick=function(event){ //when 'Logo' is clicked the
    function occurs
    document.getElementById('logo').className="hidden"; //the className of the image is
    changed to 'hidden' so that the image disappears upon click
}

```

```

document.getElementById('logo').onmouseover=function(event){ //when the mouse is on top of
'Logo', the function occurs
    document.getElementById('logo').src="ntt-logo-horz.png"; //the image 'Logo' changes to
the image 'ntt-Logo-horz.png'
    document.getElementById('mouseInput').value="Mouse Over Image"; //the textbox
'mouseInput' will display 'Mouse Over Image' as its value; this is 'value' insted of
'innerHTML' because we are interacting with an input box rather than a paragraph
}
document.getElementById('logo').onmouseout=function(event){ //when the mouse is no longer on
top of 'Logo', the function occurs
    document.getElementById('logo').src="ntt-logo.png"; //the image 'Logo' changes back to
the image 'ntt-Logo.png'
    document.getElementById('mouseInput').value="Mouse Left Image"; //the textbox
'mouseInput' will display 'Mouse Left Image'
}
document.body.onmousemove=function(e){ //when the mouse moves while on the window, the
function occurs, i'm not sure what 'e' is, but I assume it is some string/array holding
mouse location information
    var pageX=e.pageX; //creates variable that is equal to the mouse x-coordinate
    var pageY=e.pageY; //creates variable that is equal to the mouse y-coordinate
    if(pageX===undefined){ //if 'e' doesn't work properly, so it doesn't update variables:
        pageX=e.clientX+document.body.scrollLeft+document.documentElement.scrollLeft; //not
        pageY=e.clientY+document.body.scrollTop+document.documentElement.scrollTop; //sure
    }
    document.getElementById('mouseX').value=pageX; //change the 'value' of the textbox
'mouseX' to the 'pageX' variable value
    document.getElementById('mouseY').value=pageY; //change the 'value' of the textbox
'mouseY' to the 'pageY' variable value
}
document.getElementById('clearInputs').onclick=function(event){ //when the 'clearInputs'
button is clicked, a function occurs
    var inputElements=document.getElementsByTagName('input'); //creates an array that hass
all of the elements named 'input' in it
    for(var i=0; i<inputElements.length; i++){ //for all of the inputs
        if(inputElements[i].type=="text"){ //if they are of type 'text', textbox inputs
            inputElements[i].value=""; //change values to '' so they no longer have text
        }
    }
}
</script>

```

## Styling:

- (a) There are many ways to style a webpage. For many different ways to style, check out the link here: "[http://www.w3schools.com/jsref/dom\\_obj\\_style.asp](http://www.w3schools.com/jsref/dom_obj_style.asp)".

Here are a few examples of some buttons changing the styling of some text:

```

<div id="sampDiv"> Here is some text that can have lots of things done to it. Lorem ipsum
dolor sit amet, consectetur adipiscing elit. Proin eget turpis eget quam luctus maesuada ut
ac nulla. </div><br /> //Here is some text to be edited, break not needed

<button id="chgBkColor">Background Color</button><br /> //button 'chgBkColor', b.n.n
<button id="chgBkImg">Background Image</button><br /> //button 'chgBkImg', b.n.n
<button id="chgBorderStyle">Border Style</button><br /> //button 'chgBorderStyle', b.n.n
<button id="chgBorderWidth">Border Width</button><br /> //button 'chgBorderWidth', b.n.n
<button id="chgBorderColor">Border Color</button><br /> //button 'chgBorderColor', b.n.n

<script>
document.getElementById('chgBkColor').onclick=function(event){ //button 1 is clicked
    document.getElementById('sampDiv').style.backgroundColor="#EFDECD"; //color changed to
'#EFDECD'
}
document.getElementById('chgBkImg').onclick=function(event){ //button 2 is clicked
    document.getElementById('sampDiv').style.backgroundImage="url('repeatBkgrnd.png')";
    //background image changed to the image at "url('repeatBkgrnd.png')
}
document.getElementById('chgBorderStyle').onclick=function(event){ //button 3 is clicked
    document.getElementById('sampDiv').style.borderStyle="solid"; //creates a border
}
document.getElementById('chgBorderWidth').onclick=function(event){ //button 4 is clicked
    document.getElementById('sampDiv').style.borderWidth="thick"; //makes border thick
}
document.getElementById('chgBorderColor').onclick=function(event){ //button 5 is clicked
    document.getElementById('sampDiv').style.borderColor="blue"; //makes border blue
} //all edits affect the <div> paragraph tag thing
</script>

```



### Manipulating URLs:

- (a) The presenter spoke of this as ‘manipulating the DOM’, i’m not sure what ‘DOM’ is. For this section, a large `div` was included with multiple tags to manipulate. An image is included as well as 4 buttons to play with:

```
<div id="sampDiv2"><p>Lorem ipsum dolor sit amet, <em>
consectetur adipiscing</em> elit. Proin eget turpis eget
quam luctus malesuada ut ac nulla. Suspendisse fermentum
magna neque, a auctor felis pretium eget. Fusce ornare
feugiat magna, ut faucibus sapien congue ut. Nunc nec
fringilla ex, nec varius nisl. Ut eget laoreet nisi. Aenean
quis venenatis mauris, at volutpat ante. Donec sollicitudin
lacinia ornare. In quis accumsan ligula, id egestas enim.</p>
<p>Lorem ipsum dolor sit amet, <b>consectetur adipiscing</b>
elit. Proin eget turpis eget quam luctus malesuada ut ac
nulla. Suspendisse fermentum magna neque, a auctor felis
pretium eget. <em>Fusce ornare</em> feugiat magna, ut
faucibus sapien congue ut. <b>Nunc nec fringilla</b> ex, nec
varius nisl.</p></div>

<br />

<button id="goToGoogle">Go to Google</button><br />

<button id="forwardPage">Forward Page</button><br />

<button id="backPage">Back Page</button><br />

<button id="reload">Reload Page</button><br />
```

Here is some Javascript that edits the HTML:

```
<script>
document.write("Current URL : ", window.location.href, "<br />"); //will display current URL
document.write("Current HOST : ", window.location.hostname, "<br />"); //... current HOST
document.write("Current Path : ", window.location.pathname, "<br />"); //... current path

document.getElementById('goToGoogle').onclick = function(event){ //if the 'goToGoogle'
button is clicked, a function occurs
    window.location.href="http://google.com" //the window is redirected to "...google.com"
}
document.getElementById('forwardPage').onclick=function(event){ //'forwardPage' click
    history.forward(); //goes forward a page
    //history.go(int) goes 'int' pages forward in history
}
document.getElementById('backPage').onclick=function(event){ //'backPage' click
    history.back(); //goes back a page
    //history.go(-int) goes 'int' pages back in history
}
document.getElementById('reload').onclick=function(event){ //'reload' click
    window.location.reload(true); //reload is 'true', so the page is reloaded
}
</script>
```

### Editing Child Nodes:

- (a) The following Javascript edits the same HTML as the ‘Manipulating URLs’ section, but in this case the focus is on child node properties:

```

<script>
var pElements = document.getElementsByTagName('p'); //makes an array of elements where
each element is a paragraph
pElements[1].style.backgroundColor="#EFDECD"; //changes the background color of the 2nd
paragraph on the page to '#EFDECD'

document.childNodes[1].style.backgroundColor = "#FAEBD7"; //changes the background color
of the second child of the document to '#FAEBD7'
var sampDiv2 = document.getElementById('sampDiv2'); //selects element of id 'sampDiv2';
this is the element that we created in our HTML
sampDiv2.childNodes[0].style.backgroundColor="#F0FFFF" //changes the background color of
the 1st child, which is the 1st tag, to '#F0FFFF'
sampDiv2.childNodes[0].childNodes[1].style.backgroundColor="#BFAFB2" //changes the
background color of the 2nd child of what we just edited, <em>, to '#BFAFB2'

document.write("Node Type : ", sampDiv2.childNodes[0].childNodes[0].nodeType, "<br />");
//this will print 'Node Type : 3'; a type-3 node is a text-node; text-nodes cannot be
styled; the node inspected here is neight to the one inspected in the last line of code;
type-1 nodes are non-text-nodes; if we delete all nodes that are 'whitespace' then we can
use functions 'lastChild()' and 'firstChild()' to move around nodes
document.write("Node Name : ", sampDiv2.childNodes[0].childNodes[0].nodeName, "<br />");
//prints '#text', had we done 'sampDiv2.childNodes[0].childNodes[1].nodeName' we would
have printed 'EM' for the <em> tag

var nttLogo2 = document.getElementById('logo2'); //sets element of id 'logo2' to variable
document.write("Logo has alt : ", nttLogo2.hasAttribute("alt"), "<br />"); //sees if the
element with id 'logo2' had 'alt' as an attribute and writes 'Logo has alt : true', which
makes sense because it does have this attribute
nttLogo2.setAttribute("alt", "NTT Logo2"); //sets the value of 'alt' to 'NTT Logo2'
document.write("Logo alt value : ", nttLogo2.getAttribute("alt"), "<br />"); //writes out
'Logo alt value : NTT Logo2'
var attribList = document.getElementById('logo2').attributes; //sets attributes of 'logo2'
into the array 'attribList'
for(i=0; i<attribList.length; i++){ //for all attributes
    document.write("Attribute ", i, " : ", attribList[i].nodeName, " : ",
        attribList[i].nodeValue, "<br />"); //writes out 'Attribute i : name : value' for ever
        attribute, but more specifically we get more specifically we have 'Attribute 0 : src :
        ntt Logo.png // Attribute 1 : id : Logo2 // Attribute 2 : alt : NTT Logo2 // Attribute
        3 : height : 180 // Attribute 4 : width : 180'
}

var paragraph3 = document.createElement("p"); //creates a new paragraph element and sets
it to the variable 'paragraph3'
paragraph3.setAttribute("id", "paragraph3"); //sets the attributes
paragraph3.innerHTML="Proin eget turpis eget quam luctus malesuada."; //sets the innerHTML
sampDiv2.appendChild(paragraph3); //inserts 'paragraph3' after 'sampDiv2'
sampDiv2.insertBefore(paragraph3, sampDiv2.childNodes[0]); //inserts 'paragraph3' right
before the 1st childNode of 'sampDiv2' which is just before the entire thing
</script>

```

## Object-Oriented Programing:

- Object manipulation has some tricky syntax, pay attention to this closely.
- Creating an object is just assigning one variable many different variables and defining those variables for that object-variable.

```

var cust1 = { //create variable 'cust1'
    name: "John Smith", //it has a variable 'name' defined as 'John Smith', so a string
    street: "123 Main", //.....'street'.....'123 Main'.....
    city: "Pittsburgh", //.....'city'.....'Pittsburgh'.....
    state: "PA", //.....'state'.....'PA'.....
    email: "jsmith@aol.com", //.....'email'.....'jsmith@aol.com'.....
    balance: 120.50, //.....'balance'.....'120.50, so a double
    payDownBal: function(amtPaid){ //it has a function 'payDownBal' accepting 'amtPaid'
        this.balance-=amtPaid; //the 'balance' of 'cust1' goes down by 'amtPaid'
    }, //YOU MUST HAVE A COMMA AFTER EVERY VARIABLE AND FUNCTION
    addToBal: function(amtCharged){ //.....'addToBal'.....'amtCharged'
        this.balance+=amtCharged; //.....goes up by 'amtCharged'
    }
}; //ALWAYS REMEMBER THE SEMICOLON

```

- The following are a few ways of editing an objects variables afterwards:



```

document.write("Customer Name : ", cust1.name, "<br />"); //access object variables
cust1.street = "215 main"; //change object variables
cust1.country = "US"; //create new variables for objects
delete cust1.country; //delete an object's variable
for(var prop in cust1){ //for all variables called 'prop' found in object 'cust1'
    document.write(prop, "<br />"); //write out the name of 'prop'; this includes
    'name', 'street', etc..., not 'John Smith', etc...
}

```

- (d) Here is an example of a function that creates the object for you (called a constructor). There are also some examples of adding new features to this function after it has been created.

```

function Customer(name, street, city, state, email, balance){ //constructor for a customer
    this.name=name; //makes variables equal to input names;
    this.street=street; // and streets
    this.city=city; // etc...
    this.state=state;
    this.email=email;
    this.balance=balance;
    this.payDownBal=function(amtPaid){ //customer object subfunction, accepts 'amtPaid'
        this.balance-=amtPaid; //balance decreases by 'amtPaid'
    }; //DON'T FORGET SEMICOLON AFTER FUNCTION DEFINITION
    this.addToBal=function(amtCharged){
        this.balance+=amtCharged;
    }; //DON'T FORGET SEMICOLON AFTER FUNCTION DEFINITION
}
var cust2 = new Customer("Sally Smith", "234 Main", "Pittsburgh", "PA", "ssmith@aol.com",
0.00); //construction requires 'new' call
cust2.addToBal(15.50); //addToBal function used on 'cust2'
Customer.prototype.isCreditAvail = true; //all 'Customer' created objects also get the
'isCreditAvail' variable that is set to 'true' for all of these objects
Customer.prototype.toString=function(){ //all 'Customer' created objects also get the
'toString()' function; easily override the normal 'toString()' function for objects
    return this.name + " lives at " + this.street + " in " + this.city + " " + this.state +
    " email : " + this.email + " and has a balance of $" + this.balance.toFixed(2) + "
CreditWorthy : " + this.isCreditAvail; //returns a long string with the object's values
}; //DON'T FORGET SEMICOLON AFTER FUNCTION DEFINITION
document.write(cust2.toString()); //cust2 has the new 'toString' function, but it also has
the new 'isCreditAvail' variable set to true (can be seen when written out)

```

### Form Validation:

- (a) Here is some initial HTML code that defines what we're working with:

```

<div> //divides all HTML into blocks
Enter your name: //has text 'Enter your name:' in it
<input id='name' name='name' type='text' size='30' onblur='isTheFieldEmpty(this,
document.getElementById('name_help'))' /> //creates a button with id 'name', a name 'name',
that takes text as an input and has size '30' for the amount of columns for the text input,
this button 'onblur' (moving mouse away from button) calls the function 'isTheFieldEmpty'
that takes as input the text input tag (this) as well as the element with id 'name_help'
<span id='name_help'></span>[?] //should show an error message if there is something wrong
with it, will be called 'onblur' of 'name'
</div>

```

- (b) Here are some functions that allow us to dig through the HTML:

```

function editNodeText(regex, input, helpId, helpMessage){ //editText function; Look up
'regular expressions' known as 'regex', checks if information is valid, probably as
functions; 'input' is what was input into 'name' input box, gives the id of the span
'name_help', and lastly has a message passed
    if(!regex.test(input)){ //checks to see if the 'regex' is valid or not, this will occur
if it is what we expect in the input box 'input'
        if(helpId!=null){ //always TRUE unless 'helpID' is empty
            while(helpId.childNodes[0]){ //while-loop keeps iterating until the first
'childNodes' is empty, this expression is FALSE when 'helpId' is empty
                helpId.removeChild(helpId.childNodes[0]); //the first node is deleted
            }
            helpId.appendChild(document.createTextNode(helpMessage)); //makes the only child
node a node that is the node form of the message 'helpMessage' it is appended
with the 'appendChild' function and created with 'document.createTextNode'
        }
    }
    else{ //if the 'regex' is valid, expected input box 'input'
        if(helpId!=null){
            while(helpId.childNodes[0]){
                helpId.removeChild(helpId.childNodes[0]);
            } //same as before except there is no 'helpMessage' appended to the end
        }
    }
}
}
}

```

### Exception Handling:

- (a) This is essentially same as tries and catches from Java:

```

var custArray=["Tom", "Bob", "Sally", "Sue"]; //define array Length 4
var getCust = function(index){ //function 'getCust' accepts 'index'
    if((index<0) || (index>custArray.length)){ //'index' can't be within 0 and 'custArray'
length to pass this 'if'
        throw new RangeError("Index must be >=0 and < " + custArray.length); //error
        'RangeError' created with this message ^^^
    }
    else{ //'index' is within bounds of array indices
        return custArray[index]; //value at index returned
    }
}
try{ //try-block
    document.write("Customer : ", getCust(5), "<br />"); //tries to write something with the
5th index of 'getCust' which doesn't exist, so the 'RangeError' will be thrown
}
catch(ex){ //catch-block accepts exceptions 'ex', this happens if the try caught something
    if(ex instanceof RangeError){ //if the error 'ex' contains a 'RangeError'
        document.write(ex.message+"<br />"); //the 'message' of 'ex' will be written
    }
}
}

```