JavaScript objects and Arrays

Javascript Objects

- A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.
- JavaScript is an object-based language. Everything is an object in JavaScript.
- JavaScript is template based not class based.

Creating Objects in JavaScript

- By object literal
  - object={property1:value1,property2:value2.....propertyN:valueN}
- By creating instance of Object directly (using new keyword)
  - var objectname=new Object();
- By using an object constructor (using new keyword)

By object literal

```
<html>
<body>
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+"
   "+emp.salary);
</script>
</body>
</html>
```

By creating instance of Object directly (using new keyword)

```
<html>
<body>
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
</body>
</html>
```

By using an object constructor (using new keyword)

- We need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

```
<html>
<body>
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);
</script>
</body>
</html>
```

Define method in JavaScript object

We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method.

```
<html>
<body>
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;

this.changeSalary=changeSalary;
function changeSalary(otherSalary){
this.salary=otherSalary;
}
}
e=new emp(103,"Sonoo Jaiswal",30000);
document.write(e.id+" "+e.name+" "+e.salary);
e.changeSalary(45000);
document.write("<br>"+e.id+" "+e.name+" "+e.salary);
</script>
</body>
</html>
```

JavaScript Array

- **JavaScript array** is an object that represents a collection of similar type of elements.
- There are 3 ways to construct array in JavaScript
  - By array literal
  - By creating instance of Array directly (using new keyword)
  - By using an Array constructor (using new keyword)

JavaScript array literal

```
<html>
<body>
<script>
var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
</body>
</html>
```

JavaScript Array directly (new keyword)

```
<html>
<body>
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";

for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
</body>
</html>
```

JavaScript array constructor (new keyword)

- We need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

```
<html>
<body>
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
</body>
</html>
```

JavaScript Popup Boxes

- JavaScript has three kind of popup boxes:
    1. Alert box
    2. Confirm box
    3. Prompt box.

Alert Box

- An alert box is often used if you want to make sure information comes through to the user.
- When an alert box pops up, the user will have to click "OK" to proceed.

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

<h2>JavaScript Alert</h2>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  alert("I am an alert box!");
}
</script>

</body>
</html>
```

Confirm Box

- A confirm box is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.

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

<h2>JavaScript Confirm Box</h2>


<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var txt;
  if (confirm("Press a button!")) {
    txt = "You pressed OK!";
  } else {
    txt = "You pressed Cancel!";
  }
  document.getElementById("demo").innerHTML = txt;
}
</script>

</body>
</html>
```

Prompt Box

- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

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

<h2>JavaScript Prompt</h2>

<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  let text;
  let person = prompt("Please enter your name:", "Harry Potter");
  if (person == null || person == "") {
    text = "User cancelled the prompt.";
  } else {
    text = "Hello " + person + "! How are you today?";
  }
  document.getElementById("demo").innerHTML = text;
}
</script>

</body>
</html>
```

JavaScript Form Validation

Ex1:

```
<html>
<body>
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
  alert("Name can't be blank");
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters long.");
  return false;
  }
}
</script>
<body>
<form name="myform" method="post" action="valid.jsp" onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
</body>
</html>
```

Ex2:

```
<script type="text/javascript">
function matchpass(){
var firstpassword=document.f1.password.value;
var secondpassword=document.f1.password2.value;

if(firstpassword==secondpassword){
return true;
}
else{
alert("password must be same!");
return false;
}
}
</script>

<form name="f1" action="register.jsp" onsubmit="return matchpass()">
Password:<input type="password" name="password" /><br/>
Re-enter Password:<input type="password" name="password2"/><br/>
<input type="submit">
</form>
```

Ex3:Show good morning good night wish as per time Javascript

```
<!DOCTYPE html> <html> <body> <head> <title>Show good morning
    good night wish as per time Javascript</title> </head>
    <script type="text/javascript">
    document.write("<center><font size=+3 style='color: green;'>");
    var day = new Date();
    var hr = day.getHours();
    if (hr >= 0 && hr < 12) {
          document.write("Good Morning!");
    } else if (hr == 12) {
          document.write("Good Noon!");
    } else if (hr >= 12 && hr <= 17) {
    document.write("Good Afternoon!");
    } else { document.write("Good Evening!"); }
    document.write("</font></center>");
    </script>
     </body>
</html>
```

JavaScript Events

Events & Event Handlers

- Every element on a web page has certain events which can trigger invocation of event handlers
- Attributes are inserted into HTML tags to define events and event handlers
- Examples of events
  - A mouse click
  - A web page or an image loading
  - Mousing over a hot spot on the web page
  - Selecting an input box in an HTML form
  - Submitting an HTML form
  - A keystroke

Events

- onabort - Loading of an image is interrupted
- onblur - An element loses focus
- onchange - The content of a field changes
- onclick - Mouse clicks an object
- ondblclick - Mouse double-clicks an object
- onerror - An error occurs when loading a document or an image
- onfocus - An element gets focus
- onkeydown - A keyboard key is pressed

Events

- onkeypress - A keyboard key is pressed or held down
- onkeyup - A keyboard key is released
- onload - A page or an image is finished loading
- onmousedown - A mouse button is pressed
- onmousemove - The mouse is moved
- onmouseout - The mouse is moved off an element
- onmouseover - The mouse is moved over an element
- onmouseup - A mouse button is released

Events

- onreset - The reset button is clicked
- onresize - A window or frame is resized
- onselect - Text is selected
- onsubmit - The submit button is clicked
- onunload - The user exits the page

onload & onUnload Events

- The *onload* and *onUnload* events are triggered when the user enters or leaves the page
- The onload event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information
- Both the onload and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page.

onFocus, onBlur and onChange

- The onFocus, onBlur and onChange events are often used in combination with validation of form fields.
- Example: The *checkEmail()* function will be called whenever the user changes the content of the field:

```
<input type="text" size="30"
id="email" onchange="checkEmail()">;
```

Example & Demo: onblur

```
<html>
<head>
<script
   type="text/javascript">
   function upperCase() {
      var x=document.getElementById("fname").value
      document.getElementById("fname").value=x.toU
      pperCase()
   }
</script>
</head>

<body>

Enter your name:
<input type="text" id="fname" onblur="upperCase()">

</body>
</html>
```

onSubmit

- The *onSubmit* event is used to validate all form fields before submitting it.
- Example: The *checkForm()* function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be canceled. The function *checkForm()* returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<form method="post"
action="xxx.html"
onsubmit="return checkForm()">
```

Example & Demo: onSubmit

```
<html>
<head>
<script
   type="text/javascript">
   function validate() {
      // return true or false based on validation logic
   }
</script>
</head>

<body>
      <form action="tryjs_submitpage.htm" onsubmit="return
      validate()">
         Name (max 10 chararcters): <input type="text" id="fname"
         size="20"><br />  Age (from 1 to 100): <input type="text"
         id="age" size="20"><br />
         E-mail: <input type="text" id="email" size="20"><br />
         <br />
         <input type="submit" value="Submit">
      </form>
</body>
</html>
```

onMouseOver and onMouseOut

- onMouseOver and onMouseOut are often used to create "animated" buttons.
- Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="http://www.w3schools.com"
onmouseover="alert('An onMouseOver event');return
false">
<img src="w3schools.gif" width="100" height="30">
</a>
```