SunilOS

JavaScript

www.sunilos.com

www.raystec.com

Think IT Think Us
RAYS

JS control statements

- If-else
- for loop
- while
- do-while
- for in loop
- switch

SunilOS

If-else

- If else statements are used to execute the code whether the condition is true or false.

```
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
```

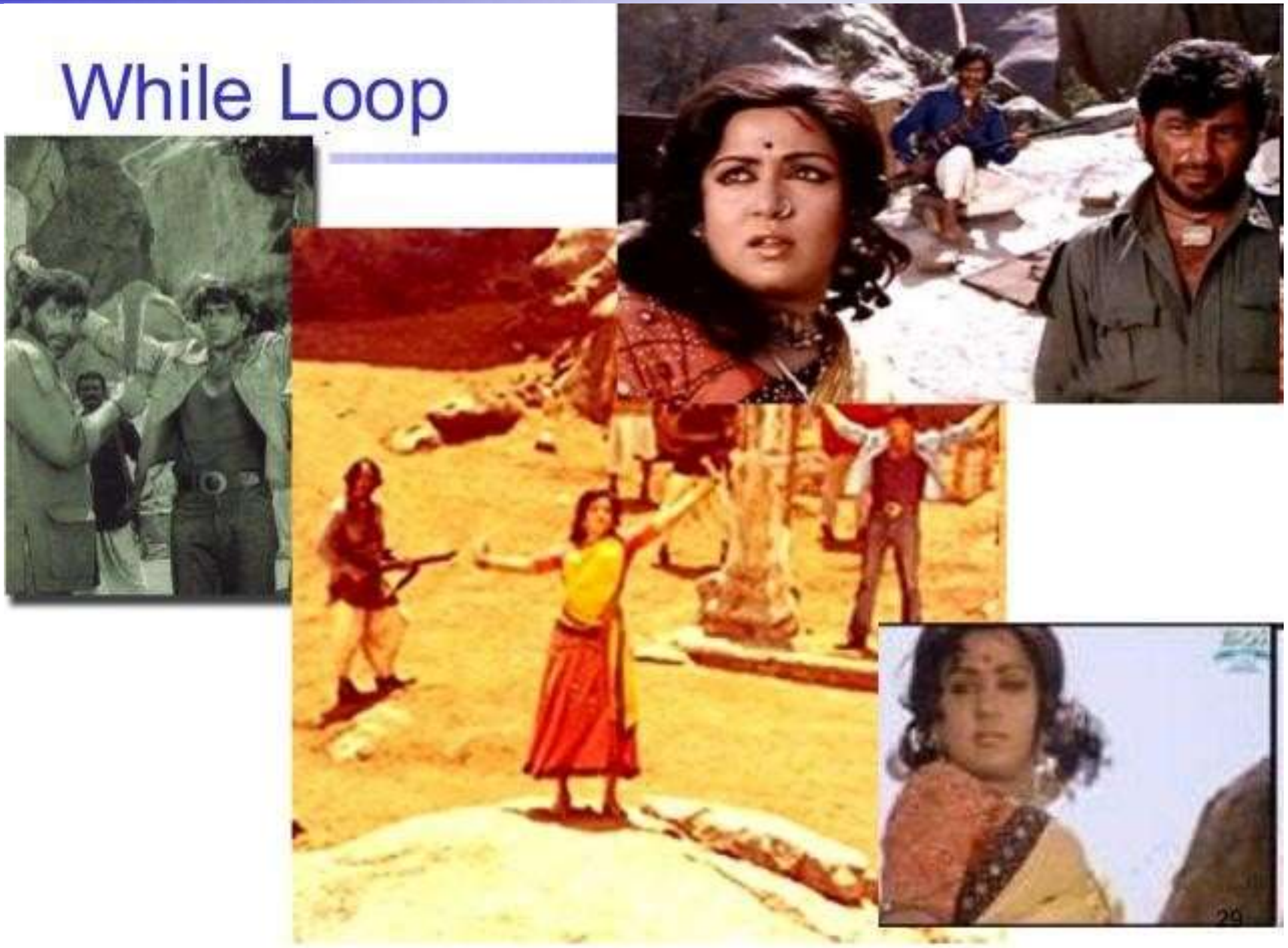
Switch Case

SunilOS

- var grade='B';
- var result;
- switch(grade){
- case 'A':
- result="A Grade";
- break;
- case 'B':
- result="B Grade";
- break;
- case 'C':
- result="C Grade";
- break;
- default:
- result="No Grade";
- }
- document.write(result);

SunilOS

While Loop



SunilOS

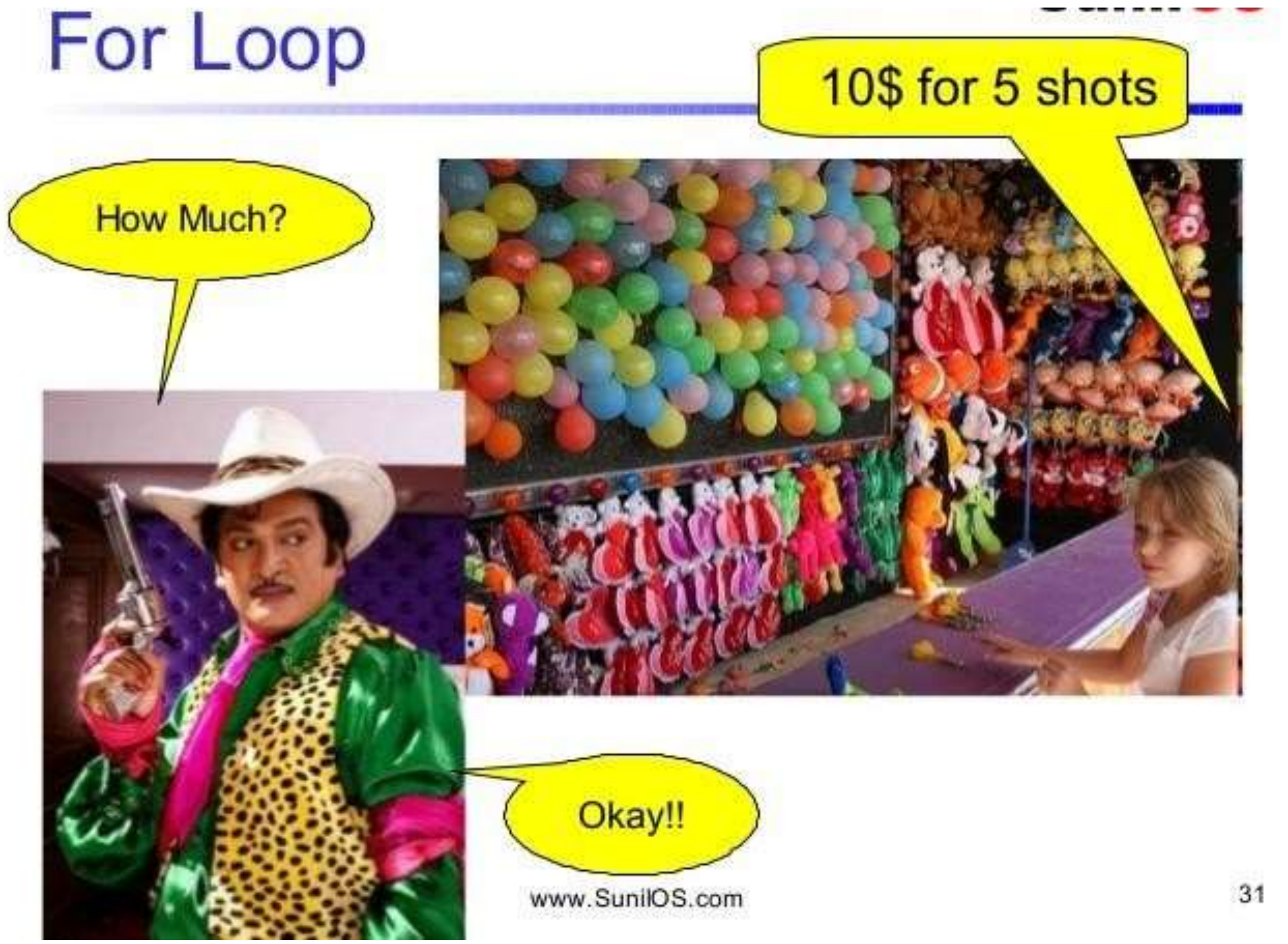
While loop

```
var i=1;
while (i<=10)
{
document.write(i + "<br/>");
i++;
}
```

do-while loop

```
var i=1;
do{
document.write(i + "<br/>");
i++;
}while (i<=10);
```

For Loop



For Loop

for loop

```
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
```

For in loop

-
  ```
  myObj= { "name":"John", "age":30, "car":null };
  ```

-
  ```
  for (x in myObj) {
  ```

-
  ```
  document.getElementById("demo").innerHTML += myObj[x];
  ```

-
  ```
  }
  ```

JS OOP concepts

SunilOS

- What is an Object?
- What is a Class?
- What is a Message?
- Encapsulation?
- Inheritance?
- Polymorphism/Dynamic Binding?
- Data Hiding?
- Data Abstraction?

JS classes

- It is just like a function in JS
- We can define a class by using class Keyword or class expression.
- The JS class contains various class members within a body including methods or constructor.
- If we are declaring a class using class keyword , we cannot re-declare it. It will throw error.
- But if we are declaring a class using class expression then we can re-declare it. It will not throw any error.

Define a class shape

```
class Shape
  {
//Initializing an object
    constructor(color,borderWidth)
    {
      this.Color=color;
      this.borderWidth=borderWidth;
    }
//Declaring method
    detail()
    {
  document.writeln("Color="+this.Color+"<br>
BorderWidth="+this.borderWidth+"<br>")
    }
  }
//passing object to a variable
var s1=new Shape("Red",4);
s1.detail(); //calling method
```

Class using Expression

SunilOS

```
Var shape=class Shape
  {
//Initializing an object
    constructor(color,borderWidth)
    { this.Color=color;
      this.borderWidth=borderWidth;
    }
//Declaring method
    detail()
    {
  document.writeln("Color="+this.Color+"<br>
BorderWidth="+this.borderWidth+"<br>")
    }
  }
```

JS objects

- Basic Unit of JS is Object.
  - Such as program of o sum of two numbers is an object
  - Fibonacci Series is an object
  - SMS Services is an object
  - Email Services is an object
  - Account Services is an object
- Basic unit of JS is an Object.
- JS is template based language not class based.
- We directly create Objects in JS.

Expert Objects

- Object is a real world entity like car, pen etc.
- Each Object is an Expert object.
- Expert object contains related variables and functions.

Creating Objects in JS

- There are 3 ways to create objects.
  - By object literal
  - By creating instance of Object directly (using new keyword)
  - By using an object constructor (using new keyword)

By Object Literal

-
  ```
  emp={id:102,name:"Shyam Kumar",salary:40000}
  ```

-
  ```
  document.write(emp.id+" "+emp.name+" "+emp.salary);
  ```

SunilOS

By using an Object Constructor

- 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);

SunilOS

By Creating instance of Object

- `var emp=new Object();`
- `emp.id=101;`
- `emp.name="Ravi Malik";`
- `emp.salary=50000;`
- `document.write(emp.id+" "+emp.name+" " +emp.salary);`

JS constructor

```
class Shape
  {
//Initializing an object
constructor(color,border
Width){
this.Color=color;
this.bw=borderWidth;
      }
}
```

- A JavaScript constructor method is a special type of method which is used to initialize and create an object.
- It is called when memory is allocated for an object.
- The constructor keyword is used to declare a constructor method.
- The class have only one constructor.
- We can access parent class constructor using super keyword.
- If we did not define any constructor JS defines default constructor.

Encapsulation

- Create Expert Classes.
- Gathering all related methods and attributes in a Class is called encapsulation.
- Often, for practical reasons, an object may wish to expose some of its variables or hide some of its methods
- Access Levels:
  - Public
  - Protected
  - Private

  Note: Var keyword is used to make data members private.

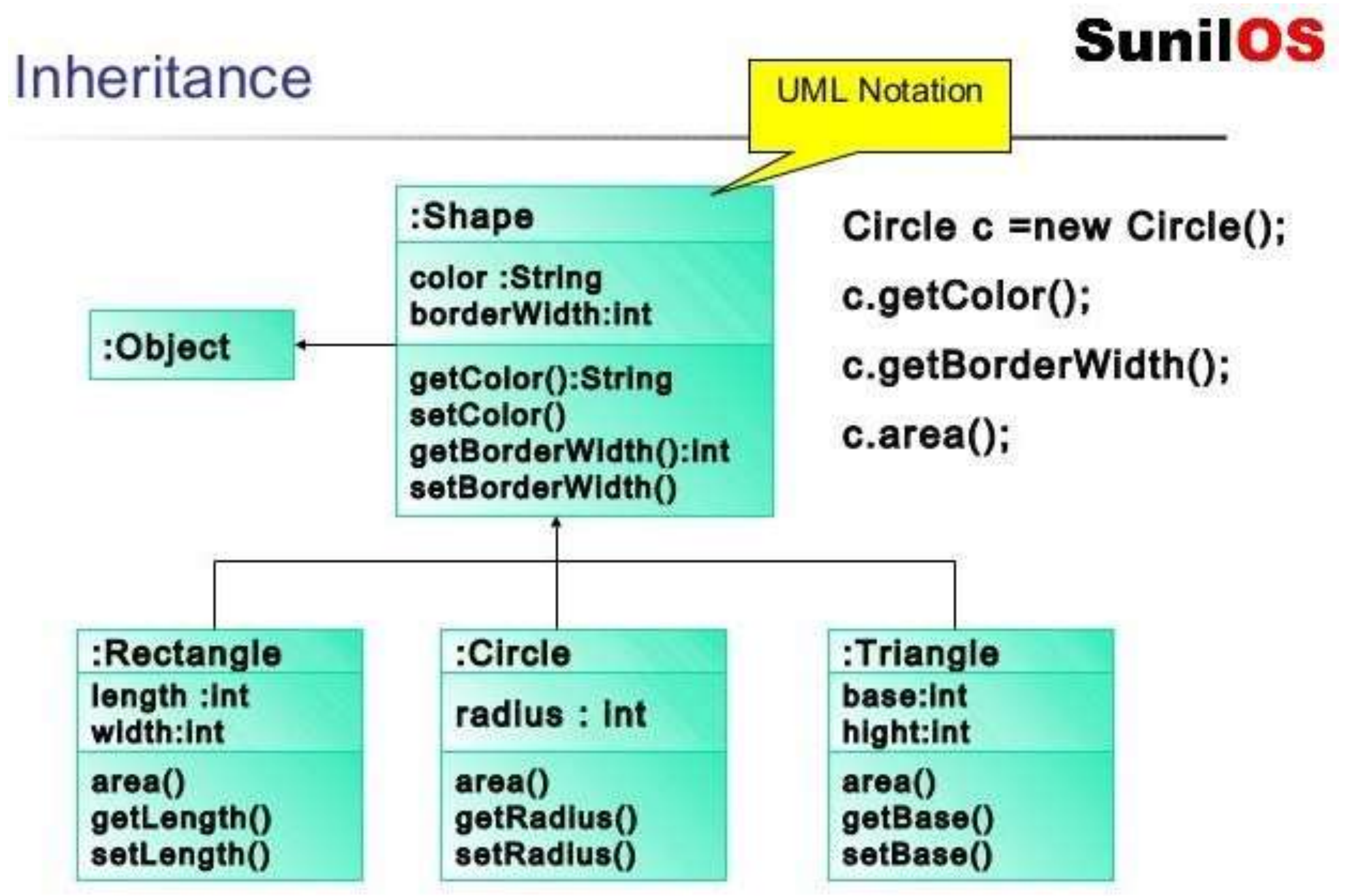
Encapsulation (cont.)

SunilOS

```
class Student{
    constructor({
        var name;
        var marks;
    }
    getName(){
      return this.name;}
    setName(name){
        this.name=name;}
    getMarks(){
        return this.marks;
      }
    setMarks(marks){
      this.marks=marks;
     }
  }
```

SunilOS

JS Inheritance

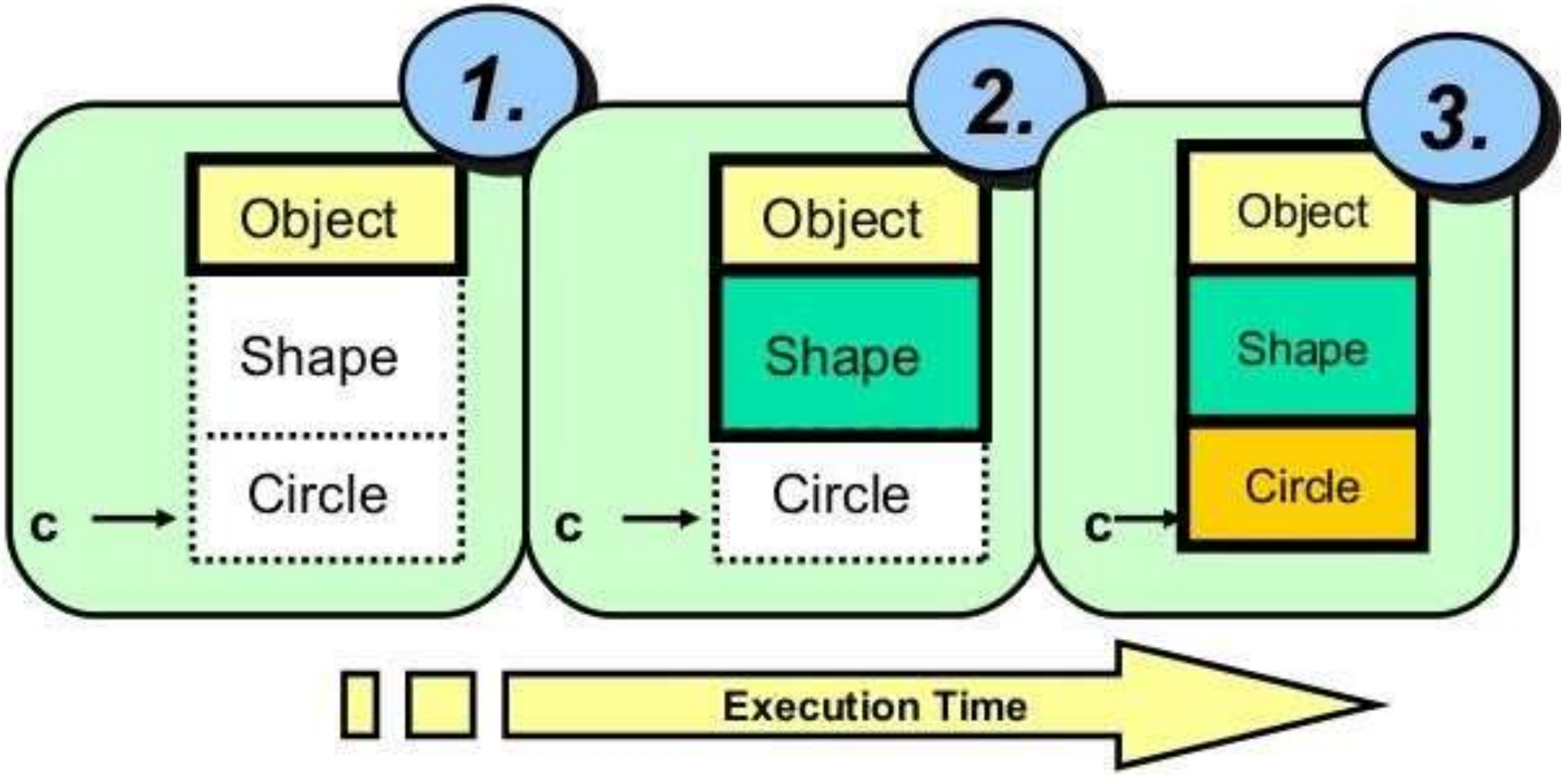


```
Circle c =new Circle();
c.getColor();
c.getBorderWidth();
c.area();
```

SunilOS

How Objects are Created

```
Circle c = new Circle( );
```



- Class Shape{
  - getColor(){
    - document.writeln(“Method in A class”);}
- }
- Class Circle extends Shape{
- }

SunilOS

Polymorphism

- Polymorphism provides a way to perform a single action in different forms.
- It provides an ability to call the same method on different JavaScript objects.

```
class A  {
     display(){
      document.writeln("A is invoked");
    }
  }
class B extends A{
  }
var b=new B();
b.display();
```

SunilOS

JS Abstraction

Data Abstraction(cont.)

- Data abstraction is the way to create complex data types and exposing only meaningful operations to interact with data type, whereas hiding all the implementation details from outside world.
- Data Abstraction is a process of hiding the implementation details and showing only the functionality.
- Data Abstraction is achieved by Abstract class in JS.
- We cannot crate instance of abstract classes.

JS Abstraction Example

```
class Employee{
constructor() {
if(this.constructor == Employee){
throw new Error(" Object of Abstract Class cannot be
created");
}}
display(){
throw new Error("Abstract Method has no implementation");
}}
class Manager extends Employee{
display(){
//super.display();
console.log("I am a Manager");
}
}
//var emp = new Employee;
var mang=new Manager();
mang.display();
```

JS exception Handling

throw

SunilOS

Exception Handling

www.sunilos.com

www.raystec.com

Think IT Think Us
RAYS

catch

Exception

- Exceptional (that is error) condition that has occurred in a piece of code of Java program.

Exception

- It will cause abnormal termination of program or wrong execution result.
- JavaScript provides an exception handling mechanism to handle exceptions.
- Exception handling will improve the reliability of application program.
- JS creates different type of objects in case of different exceptional conditions that describe the cause of exception.

Exception Handling

- Exception handling is managed by try, catch, throw, and finally keywords.
- Exceptions can be generated by
  - “run-time system” are called System-generated exceptions. It is automatically raised by run-time system.
  - Your code are called Programmatic Exceptions. It is raised by throw keyword.
- When an exceptional condition arises, an object is created that contains exception description.
- Handling is done with help of try-catch-finally block.
- Exception raised in try block is caught by catch block.
- Block finally is optional and always executed.

try-catch-finally statements

```
try {
        // code
 } catch (ExceptionType1 identifier) {
        // alternate flow 1
 } catch (ExceptionType2 identifier) {
        // alternate flow 2
} finally {
    //Resource release statements like close file ,
     //close N/W connection,
     //close Database connection, Release memory cache.
 }
```

Handle Exception

SunilOS

```
<!DOCTYPE html>
<html>
<head>
     <title>Exception HAndling</title>
</head>
<body>
<script type="text/javascript">
     try{
             var a=10;
             document.write("a="+a+"  c="+c);
     }catch(e){
             document.writeln("<h1>Error:"+e+"</h1>");
     }
</script>
</body>
</html>
```

Flow of Exception

- try {
  - a
  - b //Throw Exception
  - c
- } catch (Exception e) {
  - d
  - e
- } finally {
  - f
- }

- ❑ Normal Flow
- ❑ a b c f

- ❑ Exceptional Flow
- ❑ a b d e f

Exception Methods

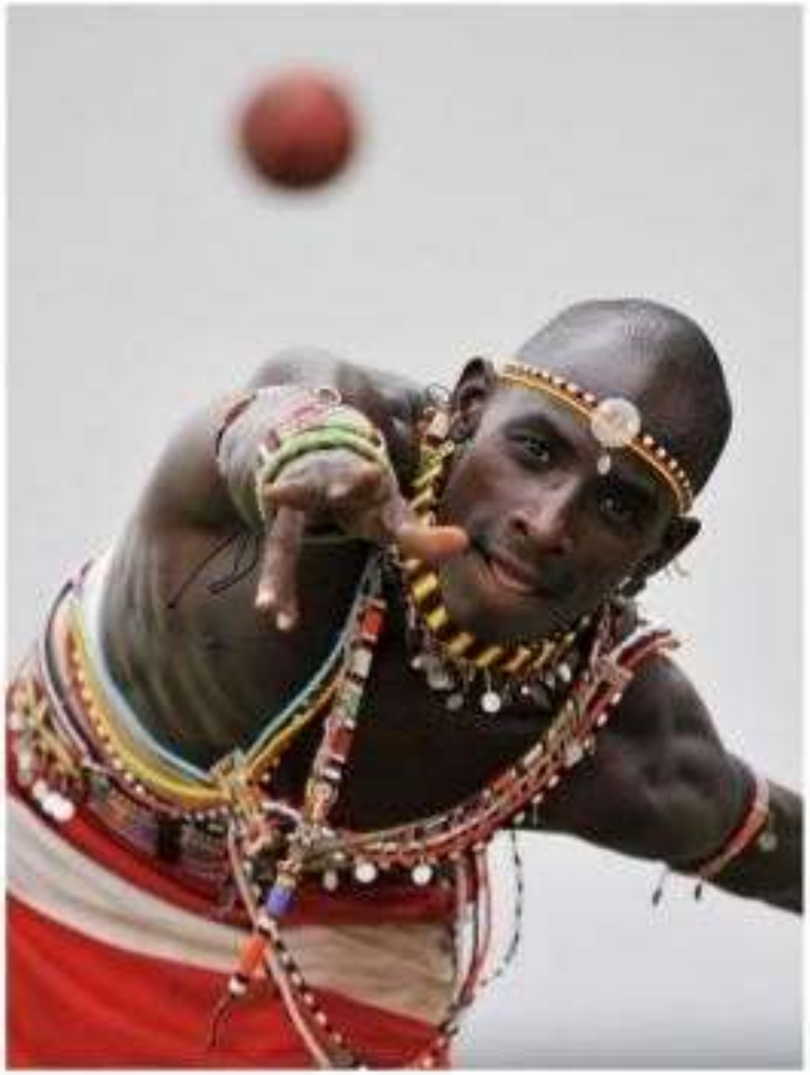
SunilOS

- An error object has two properties
- **name**: This is an object property that sets or returns an error name.
  - e.name
- **message:** This property returns an error message in the string form.
  - e.message

SunilOS

Programmer Defined Exception

Exception are raised by programmer with the help of **throw** keyword



Programmer Exception

SunilOS

- try {
- throw new Error('This is the throw keyword'); //user-defined throw statement.
- }
- catch (e) {
- document.write(e.message); // This will generate an error message
- }

JSON BOM

- The Browser Object Model (BOM) allows JavaScript to "talk to" the browser.
- **The Window Object:**
- It represents the browser's window.
- All global JavaScript objects, functions, and variables automatically become members of the window object.
- Global variables are properties of the window object.
- Global functions are methods of the window object.
- Even the document object (of the HTML DOM) is a property of the window object:
  - `window.document.getElementById("header");`
  - **Same as**
  - `document.getElementById("header");`

Window Methods

SunilOS

- window.open() - open a new window
- window.close() - close the current window
- window.moveTo() - move the current window
- window.resizeTo() - resize the current window

Window Screen

- The window.screen object can be written without the window prefix.
- Properties:
    - screen.width
    - screen.height
    - screen.availWidth
    - screen.availHeight
    - screen.colorDepth
    - screen.pixelDepth

JavaScript Popup Boxes

SunilOS

- 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.
  - `alert("I am an alert box!");`
- 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**.
  - `window.confirm(“Are you sure?");`

JavaScript Popup Boxes

- ### 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.
  - `alert("I am an alert box!");`
- ### 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**.
  - `window.confirm(“Are you sure?");`

Prompt Box

- It is used to get user input before enetering to a web 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.
- `window.prompt("sometext","defaultText");`

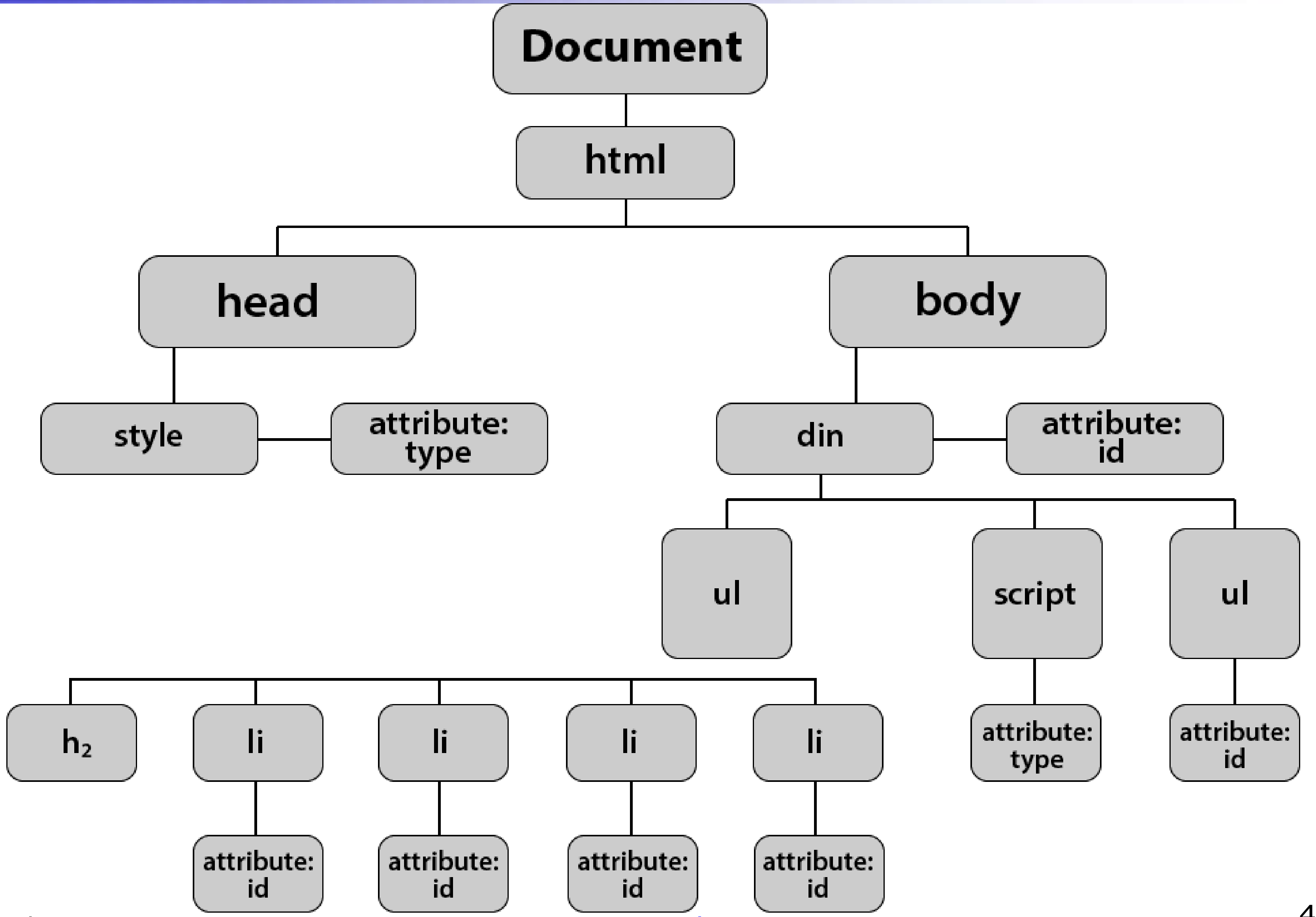
JS DOM

SunilOS

- With the HTML DOM, JavaScript can access and change all the elements of an HTML document.
- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:

SunilOS

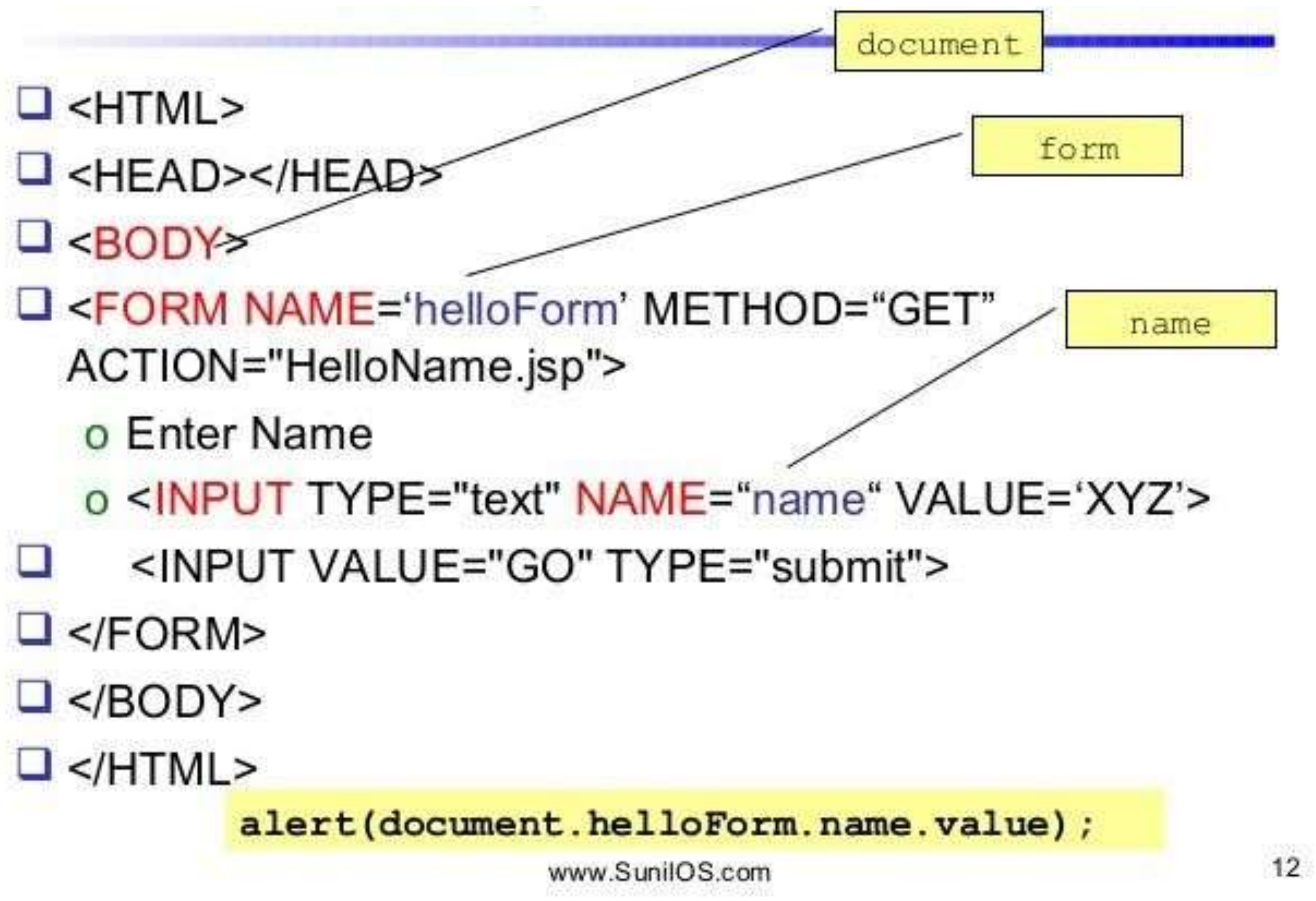
JS DOM



What is the HTML DOM?

- The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:
  - The HTML elements as **objects**
  - The **properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements
- In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

JavaScript Objects



- <HTML>
- <HEAD></HEAD>
- <BODY>
- <FORM NAME='helloForm' METHOD="GET" ACTION="HelloName.jsp">
  - Enter Name
  - <INPUT TYPE="text" NAME="name" VALUE='XYZ'>
-     <INPUT VALUE="GO" TYPE="submit">
- </FORM>
- </BODY>
- </HTML>

```
alert(document.helloForm.name.value);
```

www.SunilOS.com 12

DOM Methods & properties

SunilOS

- ❑HTML DOM methods are **actions** you can perform (on HTML Elements).
- ❑HTML DOM properties are **values** (of HTML Elements) that you can set or change.
- ❑In the DOM, all HTML elements are defined as **objects**.

JS - HTML DOM Methods & properties

- `<html>`
- `<body>`
- `<p id="demo"></p>`
- `<script>`
- `document.getElementById("demo").innerHTML = "Hello World!";`
- `</script>`
- `</body>`
- `</html>`

- In the example above, getElementById is a **method**, while innerHTML is a **property**.

The getElementById Method

- The getElementById Method
  - The most common way to access an HTML element is to use the id of the element.
  - In the example above the getElementById method used id="demo" to find the element.
- The innerHTML Property
  - The easiest way to get the content of an element is by using the innerHTML property.
  - The innerHTML property is useful for getting or replacing the content of HTML elements.

The HTML DOM Document Object

- The document object represents your web page.
- If you want to access any element in an HTML page, you always start with accessing the document object.
  - `document.getElementById(`*`id`*`)`
  - `document.getElementsByTagName(`*`name`*`)`
  - `document.getElementsByClassName(`*`name`*`)`

Finding HTML Elements

SunilOS

- Finding HTML elements by id
  - `var myElement = document.getElementById("intro");`
- Finding HTML elements by tag name
  - `var x = document.getElementsByTagName("p");`
- Finding HTML elements by class name
  - `var x = document.getElementsByClassName("intro");`
- Finding HTML elements by CSS selectors
  - `var x = document.querySelectorAll("p.intro");`
- Finding HTML elements by HTML object collections
  - `var x = document.forms["frm1"];`

SunilOS

Output Methods

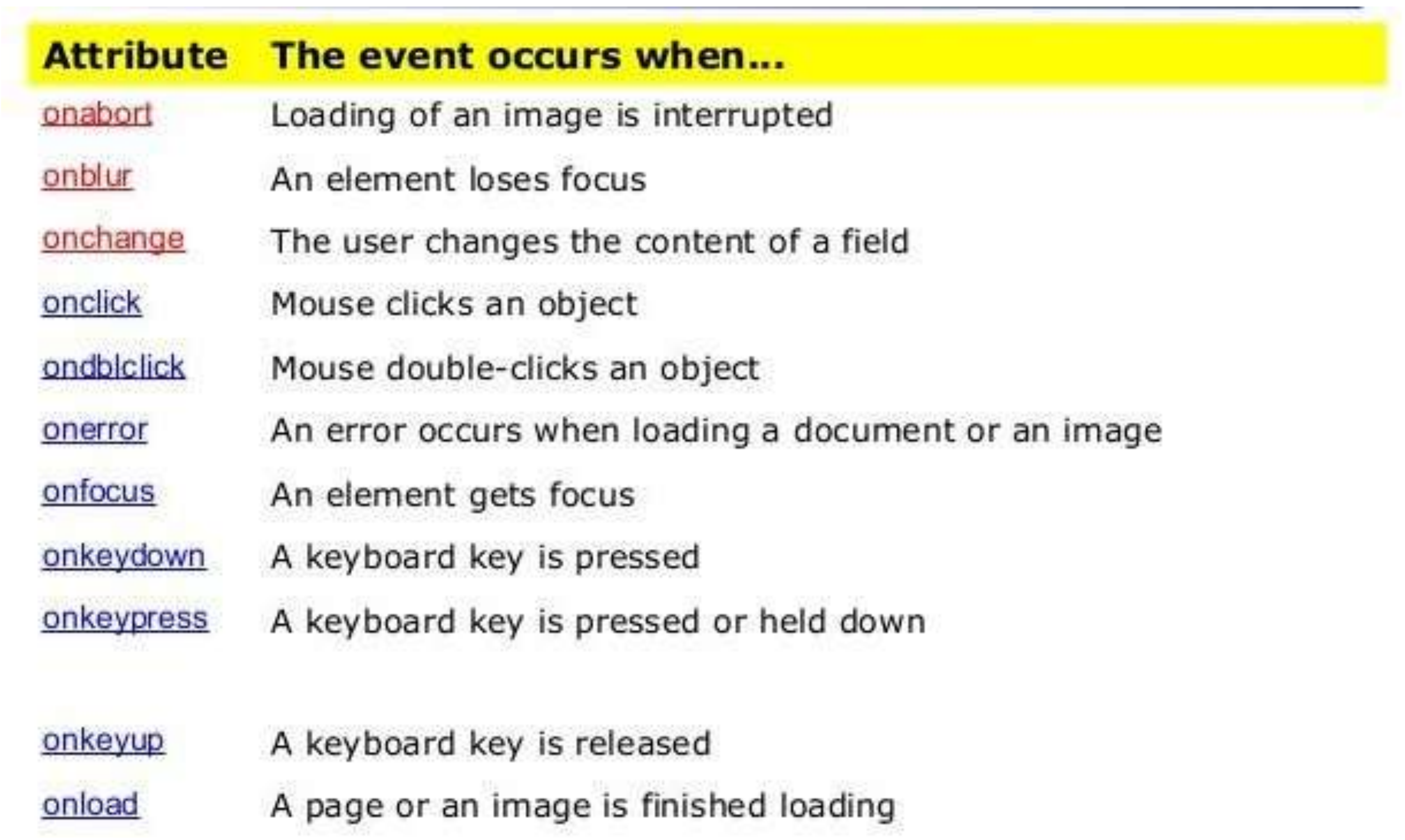
- The HTML DOM allows JavaScript to change the content of HTML elements.
  - document.write(Date());
  - document.getElementById(*id*).innerHTML = *new HTML*
  - document.getElementById(*id*).*attribute = new value*

DOM CSS

SunilOS

- We can change the style of the Html elements.
- `<html>`
- `<body>`
- `<p id="p2">Hello World!</p>`
- `<script>`
- `document.getElementById("p2").style.color = "blue";`
- `</script>`
- `</body>`
- `</html>`

JavaScript Events



Attribute	The event occurs when...
onabort	Loading of an image is interrupted
onblur	An element loses focus
onchange	The user changes the content of a field
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
onkeypress	A keyboard key is pressed or held down
onkeyup	A keyboard key is released
onload	A page or an image is finished loading

OnClick() Event

```
<!DOCTYPE html>
<html>
<head>
    <title>Event Handling</title>
    <script type="text/javascript">
    function changeColor(){
        document.getElementById('demo').style.color="Red";
    }
    </script>
</head>
<body>
<h1 id="demo" onclick="changeColor()">Heading 1</h1>
</body>
</html>
```

JavaScript Form Validation

- We can perform client side validation using JS.
- Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.
- It is faster than server side validation.

Validation on form submit

SunilOS

- `<form onsubmit="return validate(this)">`
- `Enter Name<input type="text" name="name"><br>`
- `Enter Password:<input type="password" name="pwd">`
- `<input type="submit" name="" value="Submit">`
- `</form>`

Validation Method

SunilOS

```
<script type="text/javascript">
    function validate(frm){
     a=frm.name.value;
     b=frm.pwd.value;
     var flag=true;
     if(a=='' ){
           alert("please enter the Name");
           flag=false;
     } if(b==''){
           alert("password cannot null");
           flag=false;
       }
     return flag;
     }
</script>
```

Reusable Function .js file

- Reusable function can be stored in a text file with extension .js.
- Html pages can load that function to use them.
- One Html page can import multiple .js file
- We can include .js file in HTML by <script> tag.
- `<script type=“text/javascript” src=“menu.js>`

AJAX

- Asynchronous JavaScript and XML
- HTML pages call a web resource (web page) asynchronously (in the background) without impacting existing displaying page.
- HTML pages can send and receive data with the help of AJAX from the source server asynchronously.
- Object XMLHttpRequest is used to make asynchronous calls.
- Usually JSON data fetched by AJAX calls

Get XmlHttpRequest Object

- function getXmlHttpObject(){
- var xmlHttp=null;
- try{
- // Firefox, Opera 8.0+, Safari
- xmlHttp=new XMLHttpRequest();
- }
- catch (e){
- // Internet Explorer
- try{
- xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
- }
- catch (e){
- xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
- } }
- return xmlHttp; }

TestAjax.html

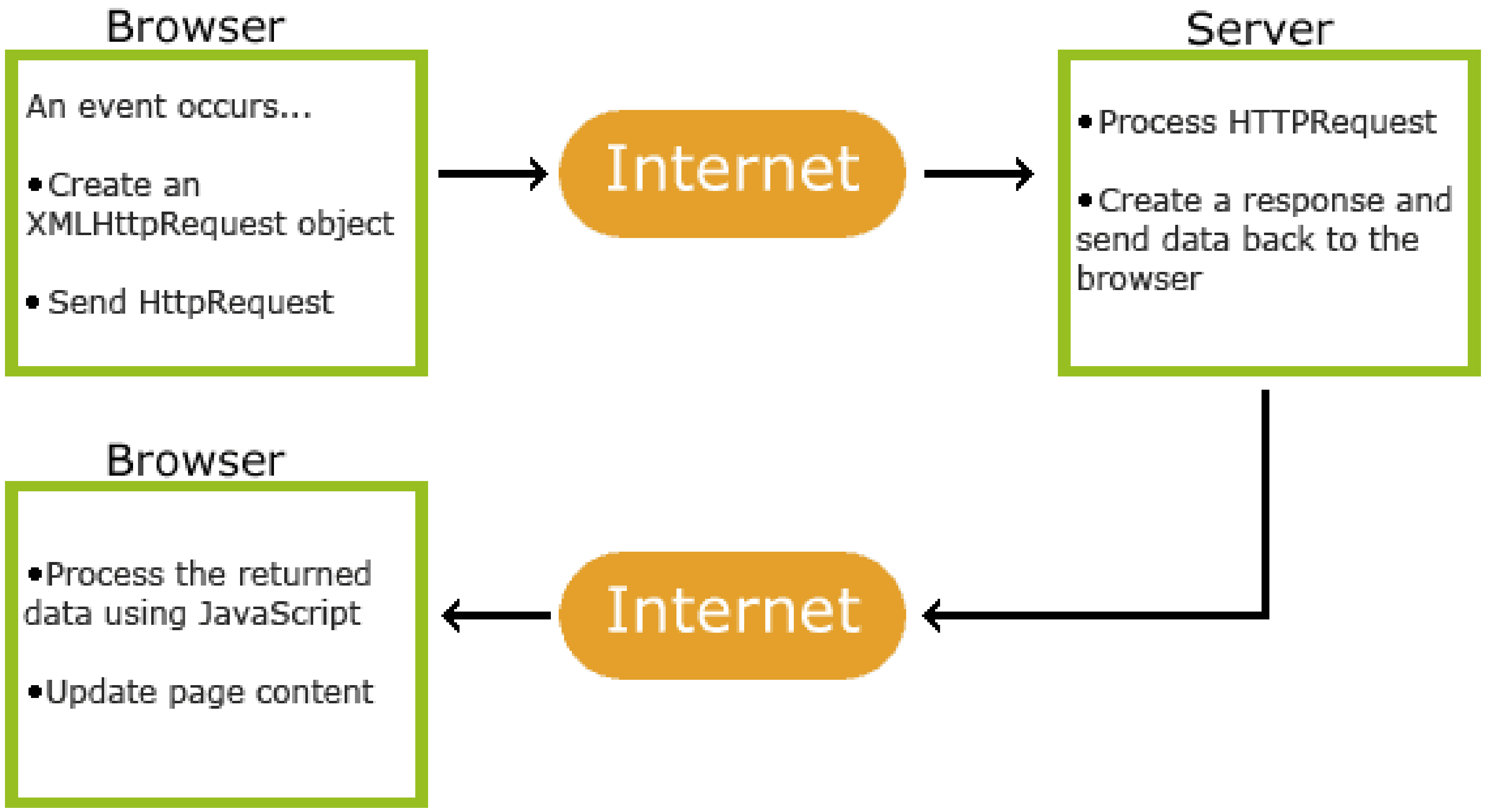
- <body>
- <div id="demo">
-   <h2>Let AJAX change this text</h2>
-   <button type="button" onclick="loadDoc()">Change Content</button>
- </div>
- </body>

Change Content of TestAjax.html using AJAX

- `<script type="text/javascript">`
- `function loadDoc() {`
  - `var xhttp = getXmlHttpObject();`
  - `xhttp.onreadystatechange = function() {`
    - `if (xhttp.readyState == 4 && xhttp.status == 200) {`
    - `document.getElementById("demo").innerHTML = xhttp.responseText;`
- `}`
- `};`
- `xhttp.open("GET", "ajax_info.txt",true);`
- `xhttp.send(null);`
- `}</script>`

SunilOS

How AJAX work?


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Browser
• Process the returned data using JavaScript
• Update page content
Internet

SunilOS

AJAX - Send a Request To a Server

- To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object
- open(*method*, *url*, *async*)Specifies the type of request
  - *method*: the type of request: GET or POST
  - *url*: the server (file) location
  - *async*: true (asynchronous) or false (synchronous)
- send()Sends the request to the server (used for GET)
- send(*string*)Sends the request to the server (used for POST)

Ajax Response

- The onreadystatechange Property
- The readyState: holds the status of the XMLHttpRequest.
- The onreadystatechange : defines a function to be executed when the readyState changes.
- The status and the statusText : holds the status of the XMLHttpRequest object.

Ajax Response (cont.)

SunilOS

- Onreadystatechange:Defines a function to be called when the readyState property changes
- readyStateHolds the status of the XMLHttpRequest.
  - 0: request not initialized
  - 1: server connection established
  - 2: request received
  - 3: processing request
  - 4: request finished and response is ready
- Status
  - 200: "OK‟
  - 403: "Forbidden‟
  - 404: "Page not found”
- statusText:Returns the status-text (e.g. "OK" or "Not Found")

JavaScript JSON

- JSON: **J**ava**S**cript **O**bject **N**otation.
- JSON is language independent.
- JSON is a syntax for storing and exchanging data with server.
- JSON is text, written with JavaScript object notation.
- When exchanging data between a browser and a server, the data can only be text.
- JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.
- We can also convert any JSON received from the server into JavaScript objects.
- This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

SunilOS

JSON syntax

- JSON syntax is derived from JavaScript object notation syntax:
  - Data is in name/value pairs
    - "name":"John"
  - Data is separated by commas:
    - "name":"John“, “Address”:”Indore”
  - Curly braces hold objects:
    - {"name":"John“, “Address”:”Indore”}
  - Square brackets hold arrays:
    - ["John", "Anna", "Peter"]
- In JSON, keys must be strings, written with double quotes.

JSON Data Types

- JSON allows following data types for values:
  - a string
  - a number
  - an object (JSON object)
  - an array
  - a boolean
  - *null*
- Values cannot be type of:
  - function
  - date
  - undefined

JSON Parse Method

SunilOS

- JSON is used to exchange data to/from a server..
- Data received by server is always a String.
- JSON.parse() method is used to parse the received string into JS object.
- Ex. Create Object from String
  - `var str='{ "name":“Vijay", "age":30, "city":“Indore"}‘`
  - `Var obj=JSON.parse(str)`
- JSON does not support date and function as value. If you want to give that type of value then give as string and after that convert them into object.

JSON stringify method

- When sending data to a web server, the data has to be a string.
- Convert a JavaScript object into a string with JSON.stringify().
- JSON.stringify will remove function from object.
- Ex. Convert array into String:
  - `var arr =["Vijay","Ajay","Jay"];`
  - `var str = JSON.stringify(obj);`
- Ex. Convert Object into String
  - `var obj = {"name":"Vijay", "age":30, "city":"Indore"};`
  - `var str = JSON.stringify(obj);`

SunilOS

JSON Object

- JSON objects are surrounded by curly braces {}.
- JSON objects are written in key/value pairs.
- Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
- Keys and values are separated by a colon.
- Each key/value pair is separated by a comma.
- Syntax:
    - `{"name":"Vijay", "age":30, "city":null}`

How to access JSON Object?

- We can access the JSON Object by using
  - a dot (.) notation
  - Or by [] notation.
- Ex. Access name value from object
- `Var obj={“name”:”Vijay”,”age”:30};`
- `X=Obj.name or x=obj[“name”];`

Assign and delete values in JSON Objects

- We can assign values to am object by dot or bracket notation.
  - `Var obj={“name”:”Vijay”,”age”:30};`
  - `Obj.name=“Ajay”;` **OR**
  - `Obj[“name”]=“Ajay”;`
- Delete Values: delete keyword is used to delete properties from json object.
  - `Var obj={“name”:”Vijay”,”age”:30};`
  - `delete Obj.name;` **OR**
  - `delete Obj[“name”];`

JSON array

- JSON array is same as JS array.
- JSON permits array values of type string, number, object, array, boolean, *null or* valid JavaScript expression, including functions, dates, and *undefined.*
- Syntax:

  -
    ```
    Var names=["Ajay","Vijay","Jay"];
    ```

Get , Modify and Delete values from array

- Access values from array:
  - `Var marks=[20,30,40,50]`
  - `Var x=marks[0]`
- Delete values from an array
  - `delete marks[0];`
- Modify values
  - `marks[0]=30;`

SunilOS

Disclaimer

- This is an educational presentation to enhance the skill of computer science students.
- This presentation is available for free to computer science students.
- Some internet images from different URLs are used in this presentation to simplify technical examples and correlate examples with the real world.
- We are grateful to owners of these URLs and pictures.

Thank You!

www.SunilOS.com