

PRACTICAL - 2 A

Aim:- Write a program to demonstrate cookies and events

Theory:-

HTML form validation can be done by Javascript. If a form field (fname) is empty, this alerts a message, and returns false, to prevent the form being submitted.

Code:-

```
<!DOCTYPE html>
<html>
<head>
<script>
function validationForm() {
let x = document.forms ["myForm"] ["fname"].value;
if (x == " ") {
alert ("Name must be filled out");
return false;
}
}
</script>
</head>
<body>
<h2> JavaScript validation </h2>
<form name="myForm" action="/action_page.php" onsubmit="
return validation()" method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form> </body> </html>
```


Output :-

Java Script Validation
Please enter a input between 1 and 20.

Name :-

Input received

fname = Aniket

PRACTICAL - 2B

1] Aim :- Demonstrate Encapsulation, Inheritance, Polymorphism abstraction.

Theory :-

Encapsulation :-

The JavaScript Encapsulation is a process of binding the data (i.e. variable) with the function acting on that data. It allows us to control the data and validation it.

- (i) Use var keyword to make data members private.
- (ii) Use setter methods to set the data and getter methods to get that data.

handle an object of encapsulation properties.

Read / write - Here, we use setter methods to write the data and getter methods to read that data.

Read only :- In this case, we use getter method only.

Write only :- In this case, we use setter methods only.

Example:- <script>

class Student

{

Constructor ()

{

var name;

var marks;

}

get Name ()

{

return this.name;

}

set Name (name)

{

this.name = name;

}

get Marks ()

{

return this.marks;

}

set marks (marks)

{

this.marks = marks;

}

}

var stud = new Student ();

stud.setName ("John");

stud.setMarks (80);

document.writeln (stud.getName () + " " + stud.get
Marks ());

</script>

Output 2-

John So

Aniket 351

Inheritance

Theory :-

→ The JavaScript inheritance To Create a class inheritance, use the extends keyword.

→ A class created with a class inheritance all the methods from another class.

Main points:-

→ It maintains an IS - A relationship

→ The extends keyword is used in class expressions or class declarations.

→ Using extends keyword,

→ we can also use a prototype-based approach to achieve inheritance

JavaScript extends ex:- inbuilt object.

Code :- <script>

```
class Moment extends Date {
```

```
  constructor() {
```

```
    super();
```

```
  }
```

```
  var m = new Moment();
```

```
  document.writeln ("Current date: ")
```

```
  document.writeln (m.getDate() + "-" + (m.getMonth() + 1) + "-" + m.getFullYear());
```

```
</script>
```


Output :-

Current date : 2-9-2021

Polymorphism

Theory:-

The polymorphism is a core concept of an object-oriented programming that provides a way to perform a single action in different forms.

It provides an ability to call the same method on different JavaScript object.

As a Java-Script is not a type-safe language we can pass any type of data members with the methods.

Code :-

Let's see an example where a child class object invokes the parent class method.

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


Output :-

A is involved

Abstraction

Theory:- An abstraction is a way of hiding the implementation details and showing only the functionality to the user.

In other words, it ignores the irrelevant details and shows only the required one.

Main points

- We cannot create an instance of abstract class.
- It reduces the duplication of code.

Abstraction example:

Let's check whether we can create an instance of abstraction class or not.

<Script>

// Creating a constructor function
function vehicle()

{

this.vehicleName = vehicleName;

throw new error("You cannot create an instance of abstract class");

}

Vehicle.prototype.display = function()

{

return this.vehicleName;

}

var vehicle = new vehicle();

</Script>