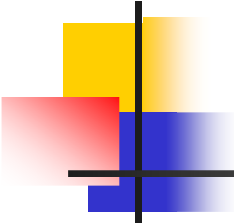


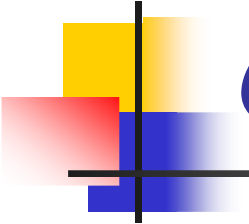
Lecture-5

- Javascript
 - OOP concepts of JS
 - Creating Javascript objects
 - Data encapsulation
 - Inheritance
 - Polymorphism
 - Useful Javascript objects



OOP features

- Main OOP concepts
 - Treat real world entities as “objects”
 - Has data and methods
- Importance features of OOP
 - Data encapsulation
 - Inheritance
 - Polymorphism
- JS supports these OOP features
 - But note: JS is a weakly typed language.
 - Implementation of these features
 - Different from strongly typed languages like C++ and JAVA



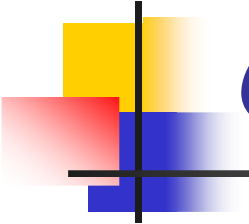
Creating JS objects

- Create an instance of an object directly

```
p1=new Object( );           // Create an object directly using new
p1.firstname="John";        // Set data variables
p1.lastname="Doe";
p1.age=50;
p1.eyecolor="blue";
p1.incrementAge = changeAge; // Set method
p1.incrementAge( );          // Call method

function changeAge( )        // Function definition
{
    this.age++;
}
```

Note: There is **NO** class keyword, as in C++, JAVA



Creating JS objects ... cond.

- Crate using a template - use function

// Template (class) definition

function person (firstname, lastname, age, eyecolor) // Constructor

```
{
  this.firstname = firstname;
  this.lastname = lastname;
  this.age = age;
  this.eyecolor = eyecolor;
  this.incrementAge = changeAge; // Define a member function
}
```

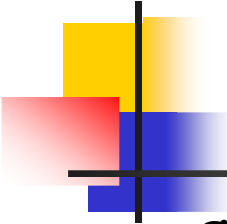
// Function definition

function changeAge()

```
{
  this.age++;
}
```

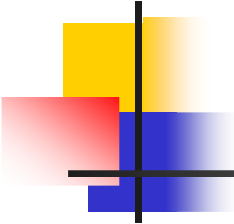
// Creating a new object of person

p1 = new person ("David", "Miller", 50, "brown");



Data encapsulation

- C++, JAVA - data encapsulation is achieved using
 - public, private, **protected**
- JS
 - public - accessible to class/external members
 - private - accessible to private/privileged members
 - **Privileged methods**
 - Can access private functions
 - Can access and change private data
 - External methods can access private members of class
 - Something like public access functions of C++, JAVA
 - **NO** protected data/methods



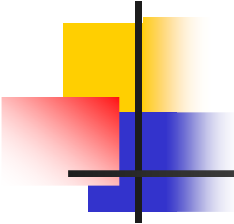
Public members

// Public data member definition

```
function public_Fn_Eg (...)  
{  
  this. publicMember = <value>;  
}
```

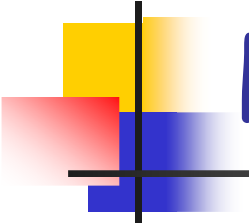
// Public function definition

```
public_Fn_Eg.prototype.pubFn = function (<params>)  
{  
  // code  
}
```



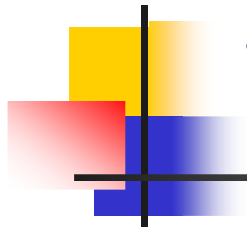
Private members

```
function private_Fn_Eg (...)  
{  
    // private data members  
    var privateMember = <value>;  
  
    //private functions  
    function privateFunction_1 (<params>)  
    {  
        // code  
    }  
  
    var privateFunction_2 = function(<params>)  
    {  
        // code  
    }  
}
```



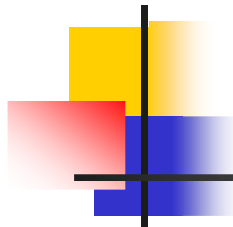
Privileged functions

```
function privileged_fn_Eg
{
    this.privilegedFn = function(...)
    {
        // CAN access private functions
        // CAN access/change private data
    }
}
```



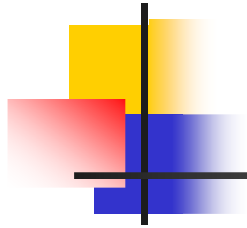
Inheritance

- Define parent and child template functions as before.
- To define the inheritance, use
 - `child.prototype = new parent;`
- Children do NOT have access to parent's private members.



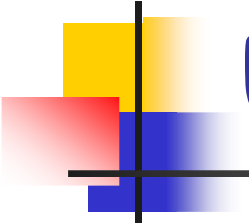
Polymorphism

- Inherently supported in Javascript
- Any object calls member function in the most specific template class.
- Child objects call member functions
 - From the child class if defined in child objects.
 - From the parent class, otherwise.
- Parent objects calls the function from the parent template class.



Static members

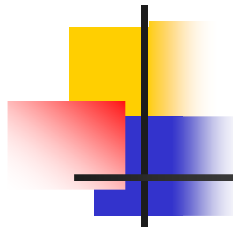
- Static member
 - Specific to the class
 - **NOT** object specific



Useful objects in JS

- String
- Array
- Boolean
- Date
- Math

<http://w3schools.com/jsref/>



Arrays

```
var mycars = new Array();  
mycars[0] = "Saab";  
mycars[1] = "Volvo";  
mycars[2] = "BMW";
```

or

```
var myCars=new Array("Saab","Volvo","BMW");  
  
myCars.push("Toyota", "Lexus"); // Add more cars  
myCars.pop( );  
document.write (myCars);
```