LECTURE-4

- OOP Concepts of JS

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 … CONTD.

- Crate using a template – use function

```
// Template (class) definition
function person (first, last, age, color) // Constructor
    {
       this.firstname = first;
       this.lastname = lasat;
       this.age = age;
       this.eyecolor = color;
       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");
```

USEFUL JAVASCRIPT OBJECTS

- String
- Array
- Boolean
- Date
- Math

http://w3schools.com/jsref/

DATA ENCAPSULATION

- Data encapsulation is achieved using
  - C++: public, private protected
  - Java: public, private
- 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

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.