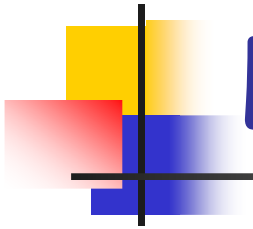
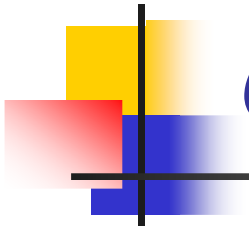


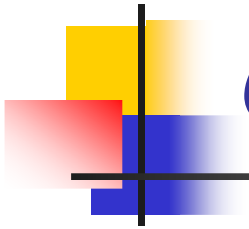
Lecture-3

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



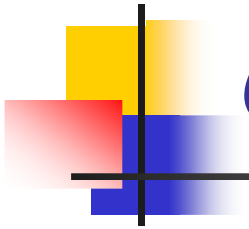
Cookies

- Small amount of information a web server stores on a browser.
- Cookie structure -- <name, value> pairs
- Typically used to
  - Remember login and password
  - User preferences
  - Web sites visited
  - Personalization
- Location where cookies are stored –
  - Different for each browser.
- Cookies have an expiration time
- Cookies can be removed



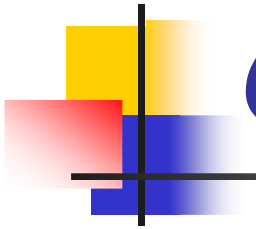
Cookies … contd.

- Cookies <name, value> pairs store
  - Name of the cookie
  - Server name and path
    - If the path is "/", cookie is valid in the entire domain
  - Expiry date
- Each web server
  - Can read its OWN cookies when the web page is loaded.
  - NOT cookies of some other web server
  - Can load multiple (up to a finite limit) cookies on each browser.



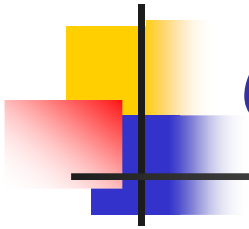
Cookies … contd.

- Cookies
  - are plain text files.
  - can't be used to read other data on the computer.
  - are not executable files
  - Cannot erase data on computer
- A site can open ONLY cookies it owns
- Cookies are set using "Set-Cookie" attribute in HTTP.



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

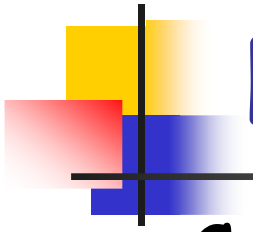
```
p1=new Object( );        // Create an object directly using new
p1.firstname="John";     // Set data variables
p1.lastname="Doe";
p1.age=50;
p1.eyecolor="blue";
p1.incrementAge = changeAge; // Set method
p1.incrementAge( );          // Call method

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

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

Creating JS objects … cond.

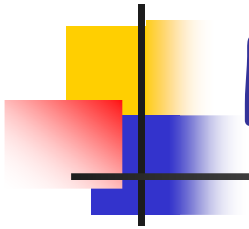
- Crate using a template – use function

```
// 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");
```



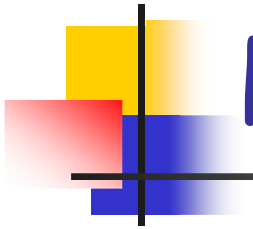
Data encapsulation

- 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



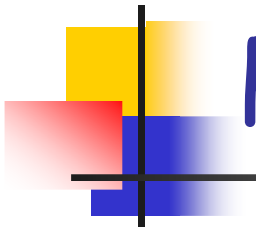
Public members

```
// 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
      }
 }
```