costFunction Or compute Cost (X, y, theta).

Same for multi

```
m= length(y)
J = 0

h = X * theta
squaredError = (h-y).^2
J = 1/(2*m) * sum (squaredError)
```

Gradient Descent (X, y, theta, alpha, num_iters)

```
    m= length(y)
    J_history = Zeros(num_iters, 1)
    for iter=1 : num_iters
        t1 = theta(1) - alpha * (1/m) * Sum ((X - theta) - y) .*X(:,1)
        t2 = theta(2) - alpha * (1/m) * Sum ((X - theta) - y) .*X(:,2)

        theta(1) = t1
        theta(2) = t2
```

Gradient Descent Multivariable :- (same same single

```
thetas = zeros(size(X,2),1);
   // store computed theta value for a simultaneous update

  for i=1:size(X,2),
      t= theta(i) - alpha * (1/m)* sum((X*theta)-y)
                                             .* X(:,i);

      thetas(i)=t
  end

  theta = thetas
```

FeatureNormalization

```
function[ X_norm, mu, sigma) =FNormalize(X)

   X_norm = X ;

   mu = zeros(1, size(X,2));

   Sigma = zeros(1, size(X,2)) ;

   mu = mean(X);

   Sigma = std(X);

  For i=1:size(X,2);
      norm = (X(:,i) -mu(i)) / sigma(i);
      X_norm(:,i) =norm
End.
```