```
In our previous understanding of Linear Regression, we were capable of
handling only 1 input variable. We were using that one input variable to
drive our regression line. This is not necessarily the ideal way to
create a predictive model. In the real world, outputs can take on a range
of features, 2,3,5,1000,1000000, the list is limitless. We have to adopt
a new way of expression our update rule so that we can correctly handle
more than 1 input variable.
```

Old Hypothesis

$h_\theta x = \theta_0 + \theta_1 x$

This was the hypothesis generated.

We would use this to drive what we were solving for.

For MSE, we were looking to find the thetas that minimized the loss function

$MSE = \frac{1}{m}\Sigma_{i=1}^{m}(h_\theta x - y_{pred})^2$

When we broke it down to its partial derivatives, we got these two update rules for theta_1 and theta_0

$\theta_0 = \theta_0 - \alpha\frac{1}{m}\Sigma_{i=1}^{m}(h_\theta x - y_{pred})$

$\theta_1 = \theta_1 - \alpha\frac{1}{m}\Sigma_{i=1}^{m}(h_\theta x - y_{pred})x$

New Hypothesis

$h_\theta x = \theta_0 + \theta_1 x^1 + \theta_2 x^2 + \theta_3 x^3 + \ldots + \theta_n x^n$

A way of thinking about this, is that each of these vectors contribute some amount to the output value. The theta value that is assigned as the coefficient is how much they contribute to the output.

We want to get used to using matrix notation, as this will vastly increase the speed of our program. We can simplify our hypothesis one step further.

$h_\theta x = \theta^\intercal x$

We want to use this, but it's not entirely usable in its current format. Remember our rules for Matrix Matrix multiplication. The columns of the first Matrix have to match the rows of the second Matrix.

Right now, our theta vector has one more column than our X has rows. What can we do to skirt around this? We can add a feature to our data set x_0. This is going to be a vector of all 1s.

Adding this in now let's us correctly simplify our hypothesis as stated previously.

Our new update rule

$$\theta_0 = \theta_0 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} (h_\theta x - y_{pred})$$

$$\theta_1 = \theta_1 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} (h_\theta x - y_{pred}) x$$

This was our previous update rule for gradient descent, it allowed us to iteratively find the gradient of the loss function, and then move towards the opposite direction of the gradient. This however only accounts for a singular feature.

Our update rule now has to be able to handle all of the features.

Our new update rule is going to be as follows

$$\theta_j = \theta_j - \alpha \frac{1}{m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

How did we get shorter? Well it's because we've packed more logic into our gradient descent update loop. j is going to assume the values between 0 and n, which means that it's going to update all thetas for all of our feature vectors.

If we were to build it out for a problem that took in two inputs for example it would be as follows.

$$\theta_0 = \theta_0 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 = \theta_1 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 = \theta_2 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

This is the new update cycle that would be repeated until convergence.

This is not a large step up from what we are used to. It simply has to update more thetas since there are more values to account for in our new Regression Model. These thetas are going to be attached to their respective feature.