***Within a Constructor***

- “this.propName” will create a public variable

- “var propertyName” will create a private variable

- “this.funcName = function(){}” will create a public function

- “var funcName = function(){}” will create a private function

- Public properties and methods must be referred to using the “this.” keyword.

- Private properties and methods must simply be referred to with their name without the “this.” keyword.

- Referring to private properties and methods with the “this.” keyword will result in an error and should not be done.

- Attempting to refer to a public property without the “this.” keyword will actually create a new variable that is private and accessible throughout the entire class.

- The context of the “this.” keyword changes when used in a function to passed to “setInterval(functionName, delay)”. When “setInterval()” calls that function, “this.” no longer refers to the class object but to the global object.

To solve this, create a private property “var self = this” and use “self.” instead of “this.” within functions passed to “setInterval()”. Alternatively use “call()” or “apply()” methods to call the functions and specify the context of the “this.” keyword in the arguments.

**/\*Constructor\*/ function MyClass() {**

**this.publicProperty = “Property”;**

**this.Method1 = function()**

**{**

**//New variable created that is private   
publicProperty = “New Private Property”;**

**console.log(this.publicProperty); //Property**

**console.log(publicProperty); //New Private Property**

**}**

**this.Method1();**

**this.Method2 = function Method2()**

**{**

**console.log(this.publicProperty); //Property**

**console.log(publicProperty); //New Private Property**

**}**

**}**

- Internet Explorer does not support the javascript method “Math.sign()