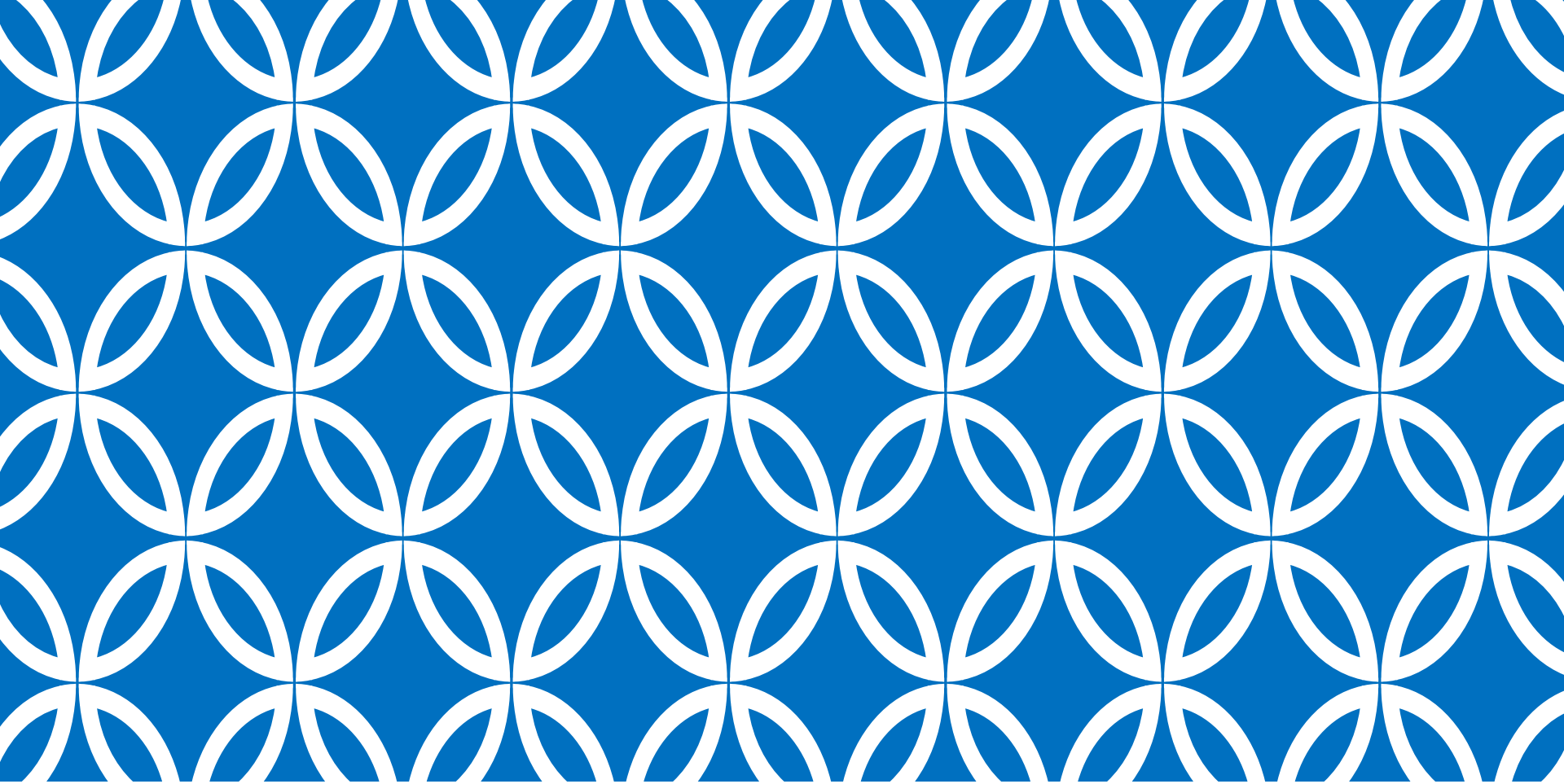
LEARN. DO. EARN

ACADGILD

FRONT END DEVELOPMENT (WITH ANGULARJS)

Website : http://www.acadgild.com
LinkedIn : https://www.linkedin.com/company/acadgild
Facebook : https://www.facebook.com/acadgild



Session 3 – Advanced JavaScript (contd.)

Agenda – Advanced JavaScript

- JavaScript Object Literal
- Creating Object using Constructor
- Creating Object using existing Object
- Private, Privileged, Public & Static Members
- JavaScript Object Property
- Accessing JavaScript Property

JavaScript Object Literal

- An object literal is a comma-separated list of name-value pairs wrapped in curly braces.
- An object literal encapsulate data, enclosing it in a tidy package. This process minimizes the use of global variables which can cause problems when combining the code.
- Object literal property values can be of any data type, including array literals, functions, and nested object literals.

Example:

- var myObject = {sProp: 'some string value', numProp: 2, bProp: false};

Creating Object using Constructor

- Sometimes we like to have an "object type" that can be used to create many objects of one type.
- The standard way to create an "object type" is to use an object constructor function as shown below:

```
function person(first, last, age, eyecolor) {
   this.firstName = first;
   this.lastName = last;
   this.age = age;
   this.eyeColor = eyecolor;
}
var myFather = new person("John", "Doe", 50, "blue");
var myMother = new person("Sally", "Rally", 48, "green");
```

Creating Object Using Existing Object

- **Object.create()** takes one object and makes it the prototype of a newly created object.
- used to create new objects using existing object

Syntax :

- **Object.create**(object, propertiesObject);
  - **object** – Required Argument
  - **propertiesObject** – Optional Argument

- Var newObject = Object.create(myFather);
- 'newObject' will contain same properties as 'myFather' object
- Console.log(newObject.firstName) //prints John

Private, Privileged, Public & Static Members

- Private variables are declared with ‘var’ keyword inside the object and can be accessed only by private functions and privileged methods.
- Privileged methods are declared with *this.methodName=function(){…}* and can be called by privileged methods.
- Public properties are declared with *this.variableName* and can be read or written from outside the object.
- Static properties are defined by *Classname.propertyname = someValue*

```
function Kid (name) {   // Constructor
 var idol = "Paris Hilton"; // Private
 this.getIdol = function () { return idol; };  // Privileged
 this.name = name; // Public
 }
 // Public
 Kid.prototype.getName = function () { return this.name; };
 // Static property
        Kid.town = "South Park";
```

Private, Privileged, Public & Static Members (contd.)

```
// Create a new instance
   var cartman = new Kid("Cartman");
// Access private property
   cartman.idol; // undefined
// Access privileged method
   cartman.getIdol(); // "Paris Hilton”
 // Access public property
   cartman.name; // "Cartman"
// Access public method
   cartman.getName(); // "Cartman"
// Access static property on an instance
   cartman.town; // undefined
// Access static property on the constructor object
    Kid.town; // "South Park"
```

JavaScript Object Property

- Properties are the most important part of any JavaScript object.
- Properties are the values associated with a JavaScript object.
- Properties can usually be changed, added and deleted but some are read only.
- A JavaScript object is a collection of unordered properties.

Accessing JavaScript Property

- The syntax for accessing the property of an object is:
  - objectName.property
  - objectName["property"]
  - objectName[expression]

Lets Discuss Assignments

Assignment

Get in Touch with ACADGILD

Contact Info:

- **Website : http://www.acadgild.com**
- **LinkedIn : https://www.linkedin.com/company/acadgild**
- **Facebook : https://www.facebook.com/acadgild**
- **Support: support@acadgild.com**