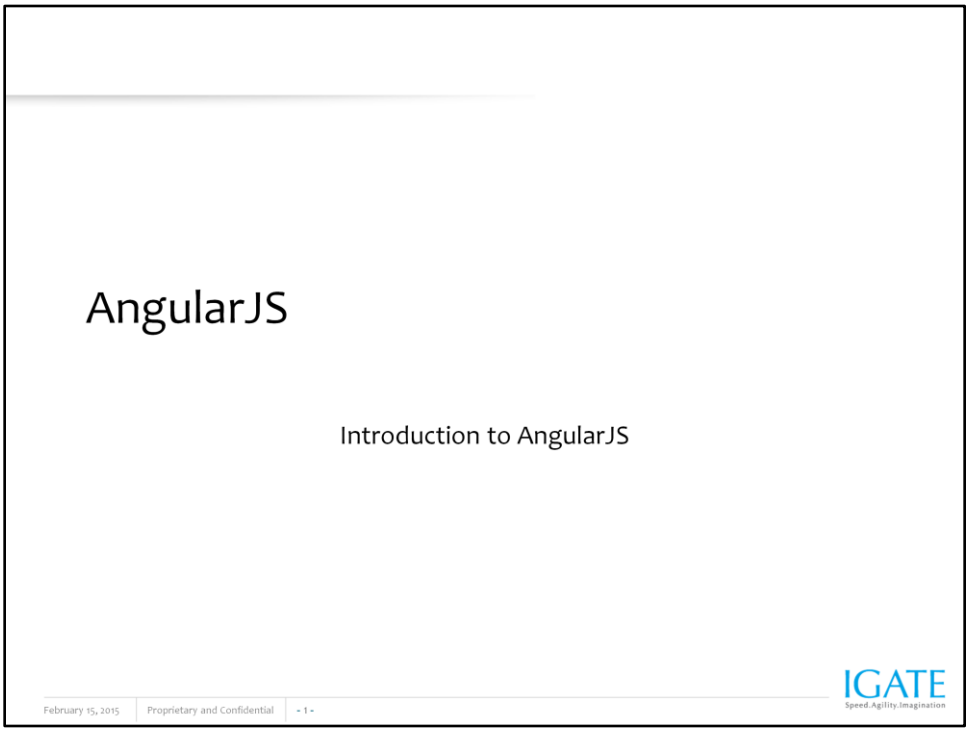
Instructor Notes:

Add instructor notes here.



Instructor Notes:

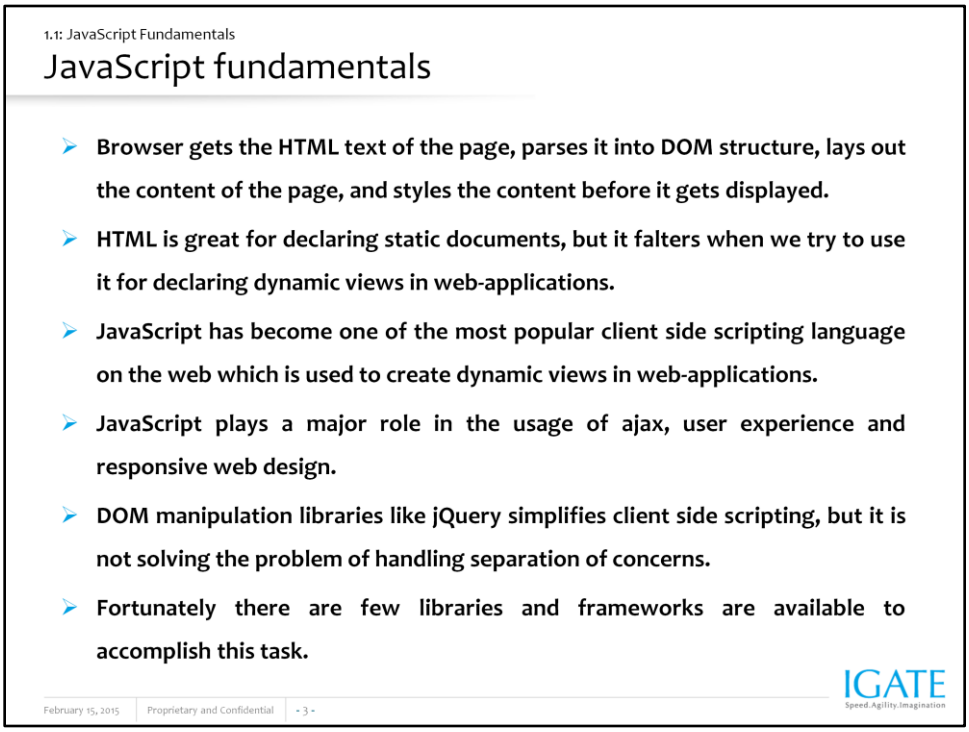
Add instructor notes here.

Lesson Objectives

- JavaScript fundamentals
- MV* Frameworks
- AngularJS Fundamentals

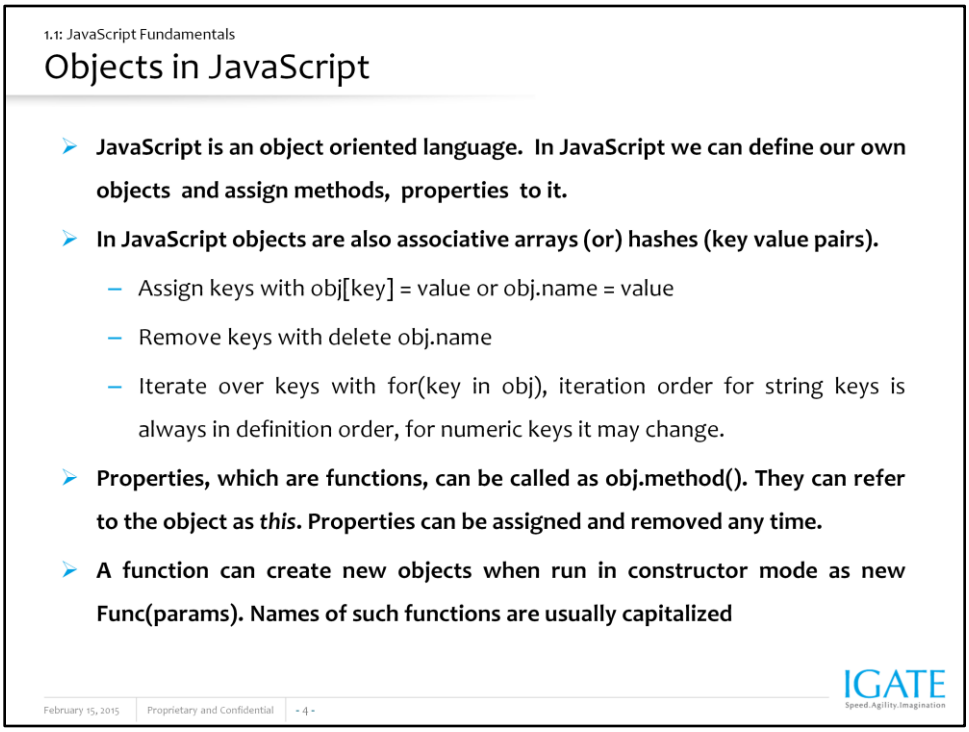
Instructor Notes:

Add instructor notes here.



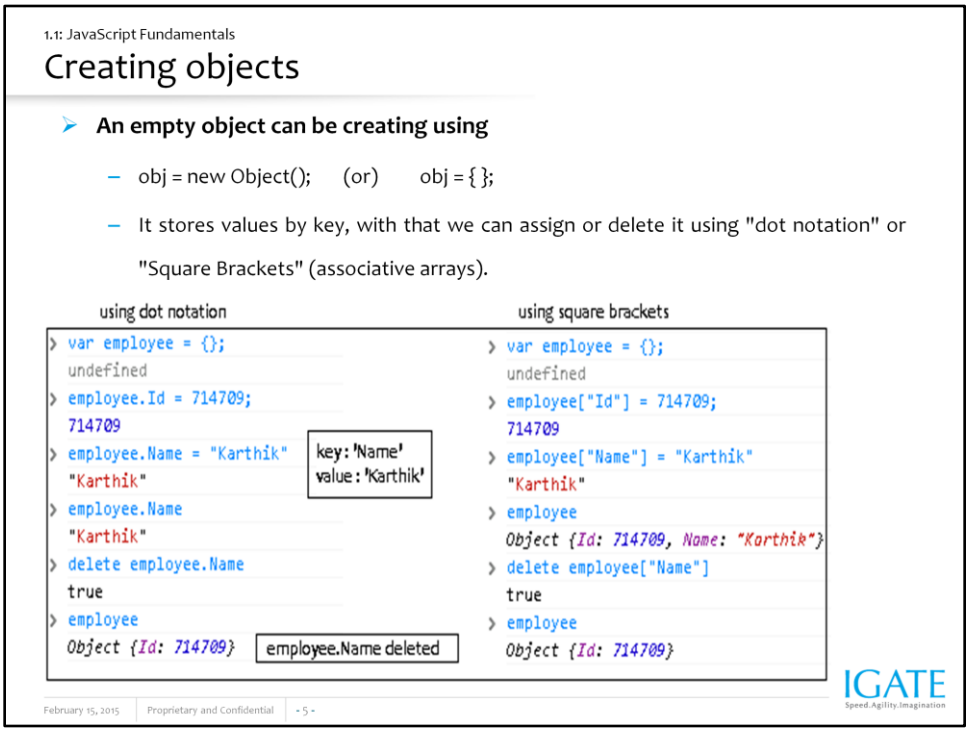
Instructor Notes:

Add instructor notes here.



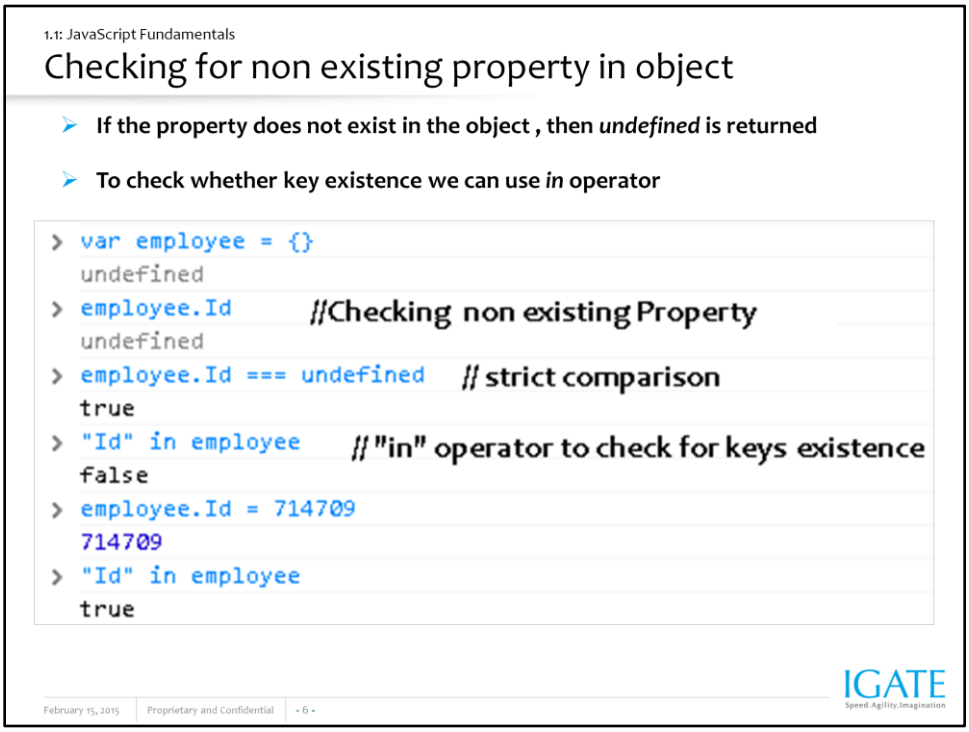
Instructor Notes:

Add instructor notes here.



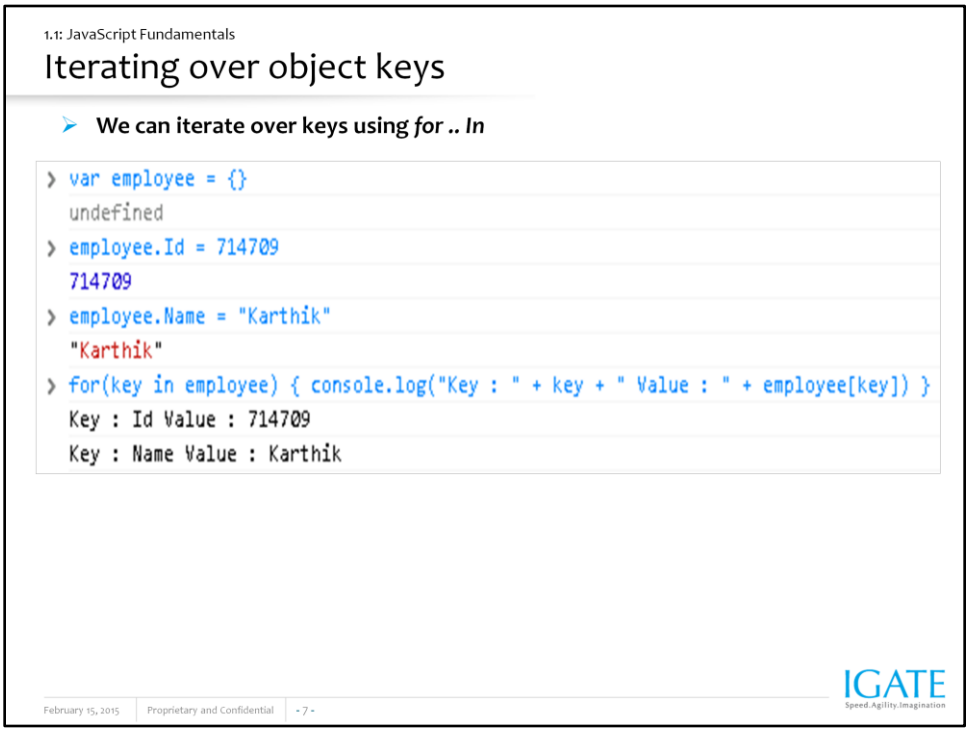
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.

1.1: JavaScript Fundamentals

Object reference

- A variable which is assigned to object actually keeps reference to it.
- It acts like a pointer which points to the real data. Using reference variable we can change the properties of object.
- Variable is actually a reference, not a value when we pass an object to a function.

```
> var employee = {};
  undefined
> employee.Id = 714709;
  714709
> var obj = employee;  // now obj points to same object
  undefined
> obj.Id = 707224;
  707224
> employee.Id
  707224
```


IGATE Speed.Agility.Imagination

Pass by value

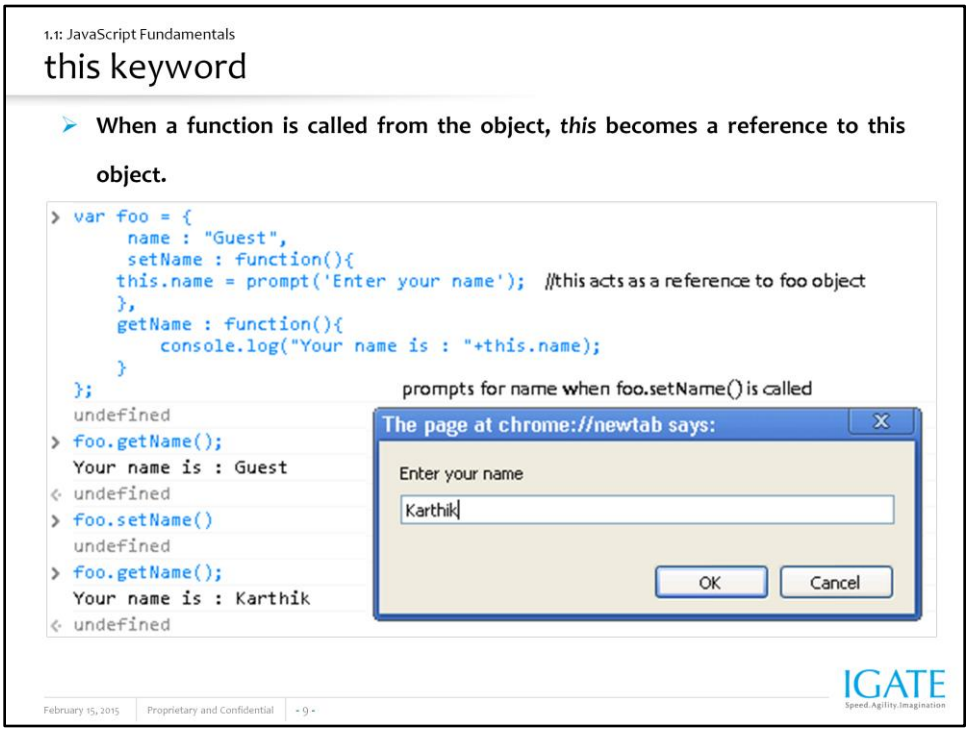
```
> function incrementValue(x){
      x++;
     console.log("Inside function x = "+x);
  }

  var x = 5;
  incrementValue(x); //Passing the Value
  console.log("x = "+x);
  Inside function x = 6
  x = 5
```

Pass by reference

```
> function incrementValue(obj){
      obj.x++;
     console.log("Inside function x = "+obj.x);
  }

  var obj = {x:5};
  incrementValue(obj); //Passing the reference
  console.log("x = "+obj.x);
  Inside function x = 6
  x = 6
```

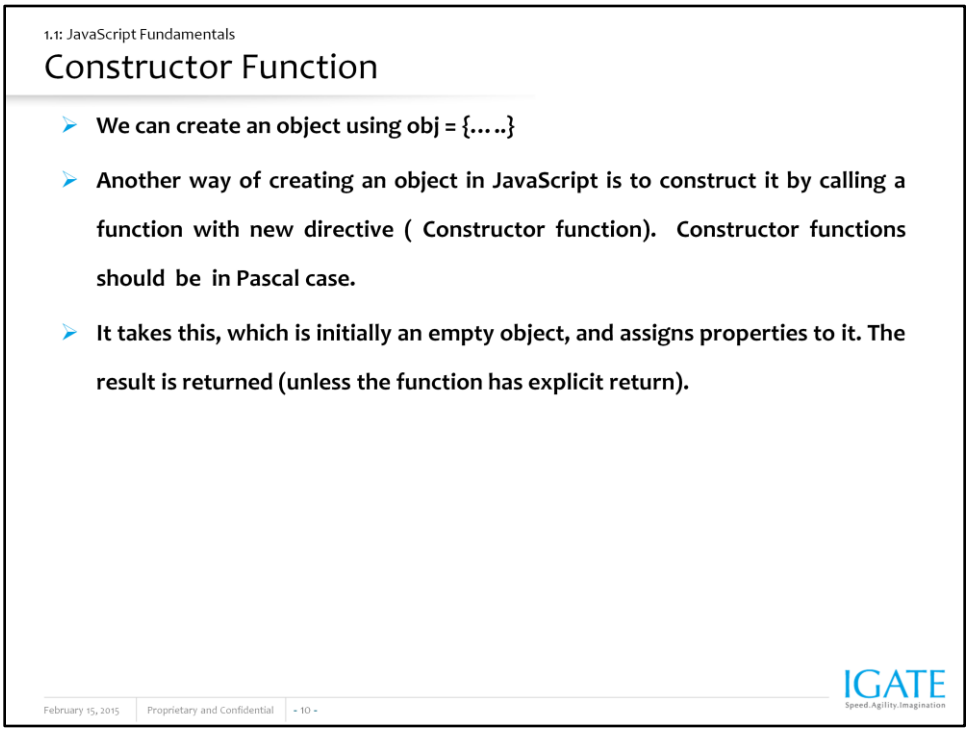
Instructor Notes:

Add instructor notes here.



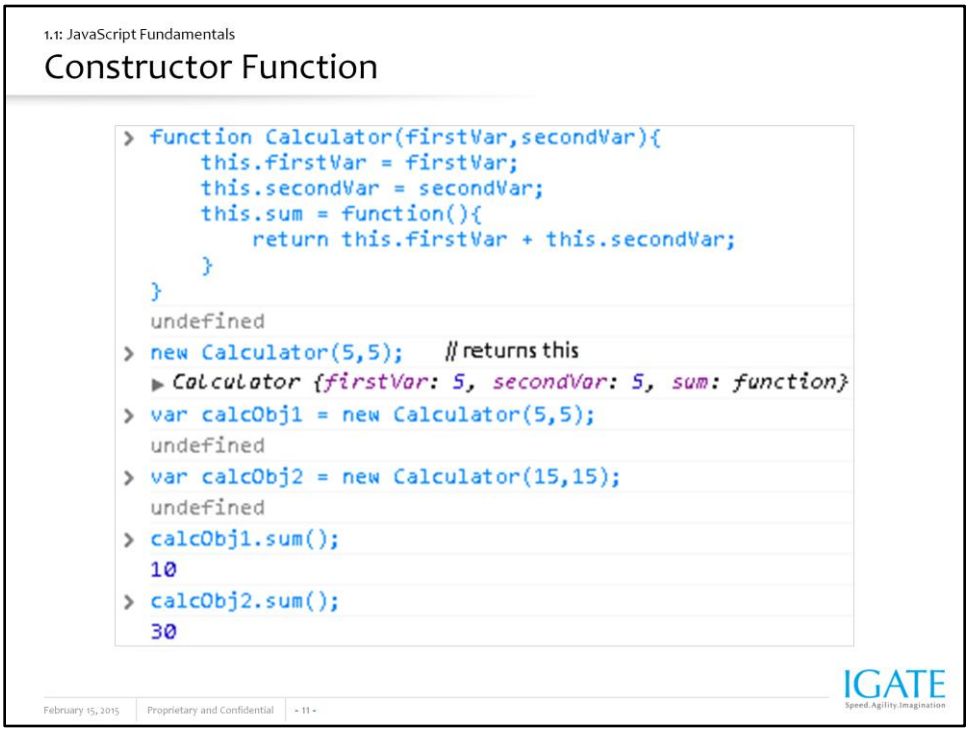
Instructor Notes:

Add instructor notes here.



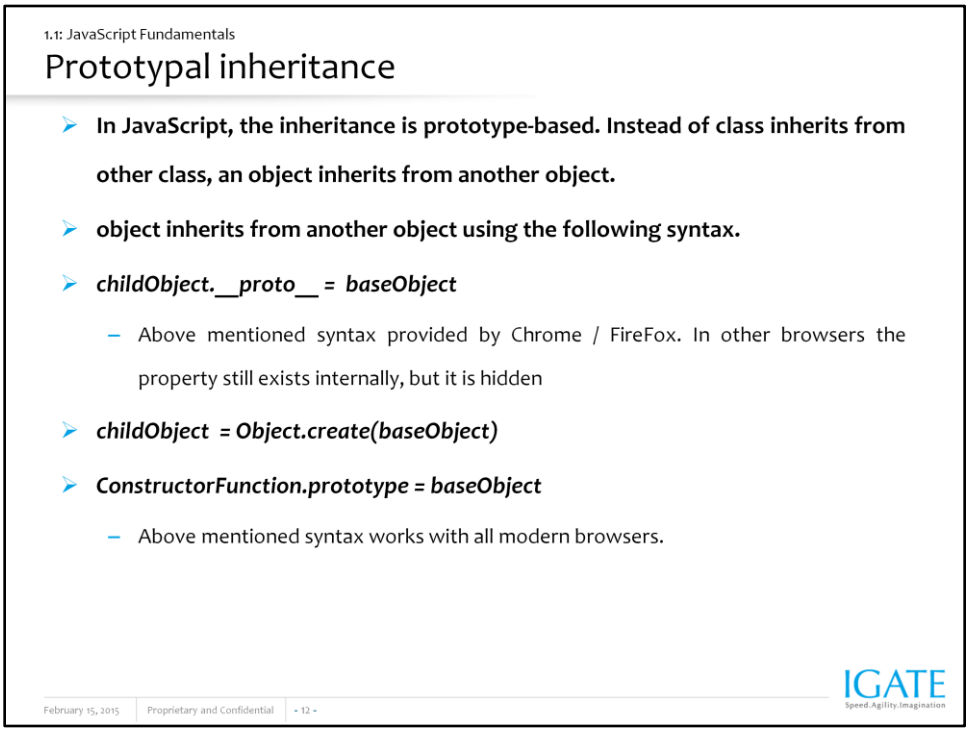
Instructor Notes:

Add instructor notes here.



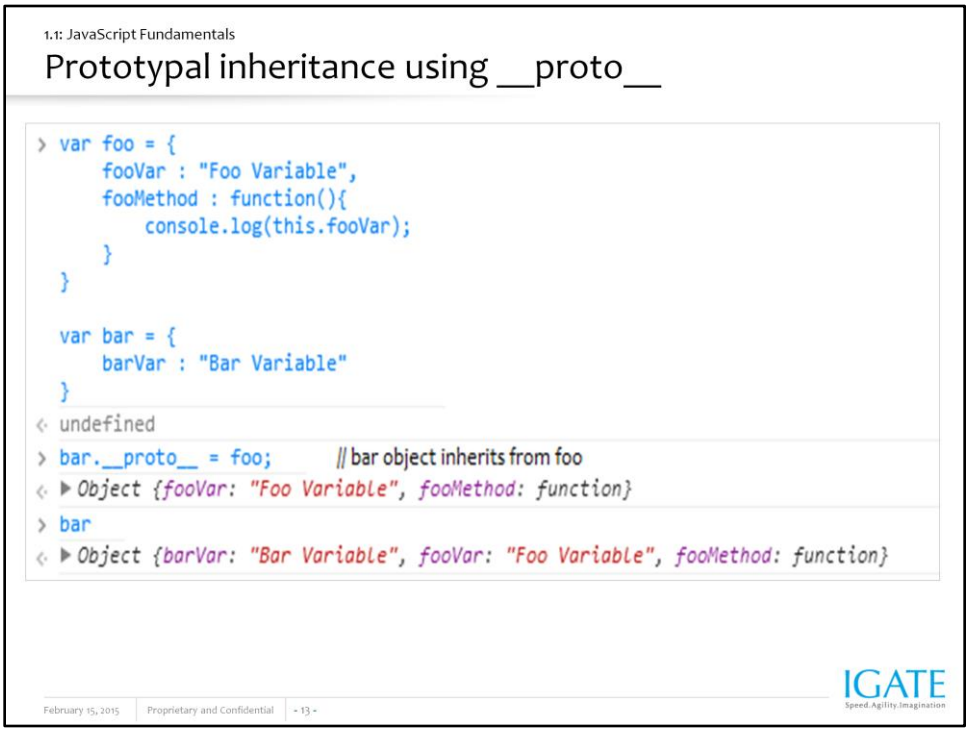
Instructor Notes:

Add instructor notes here.



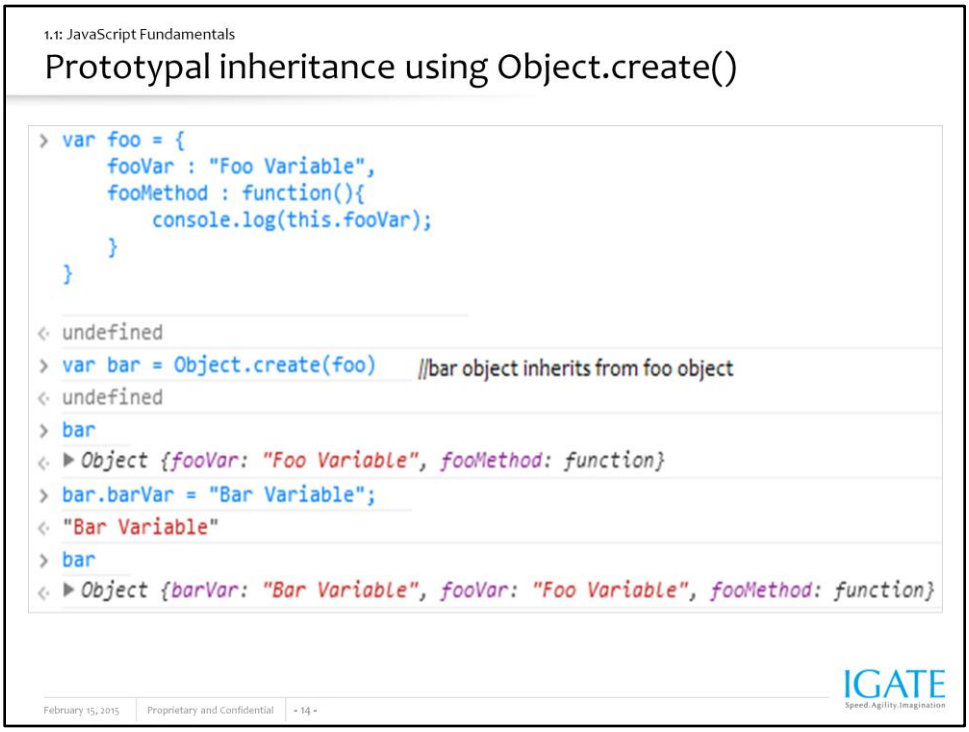
Instructor Notes:

Add instructor notes here.



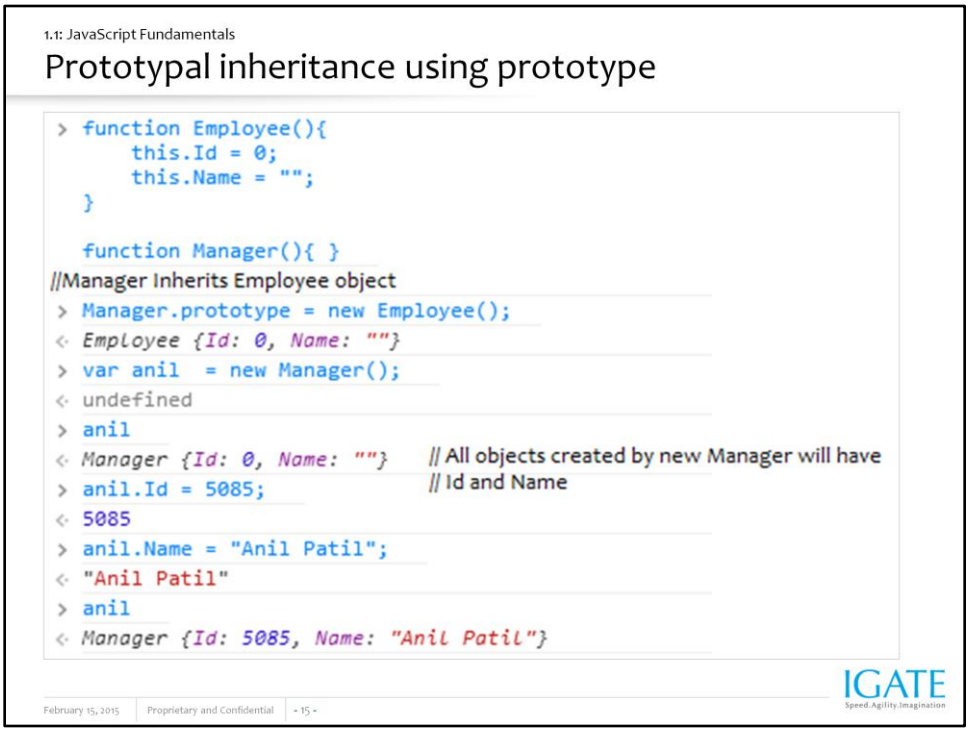
Instructor Notes:

Add instructor notes here.



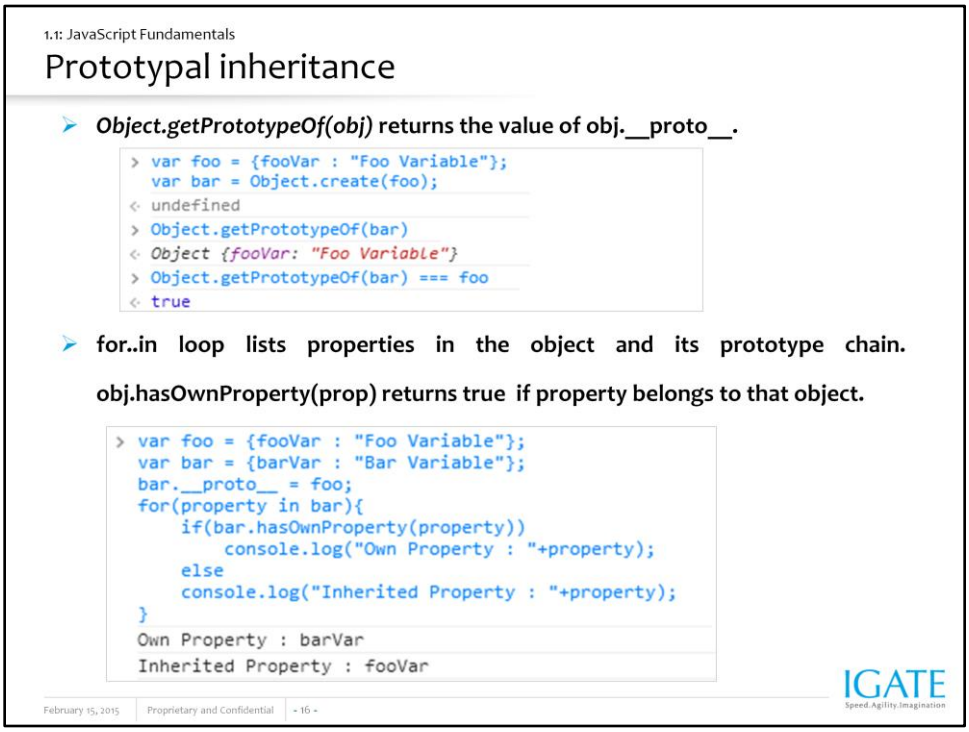
Instructor Notes:

Add instructor notes here.



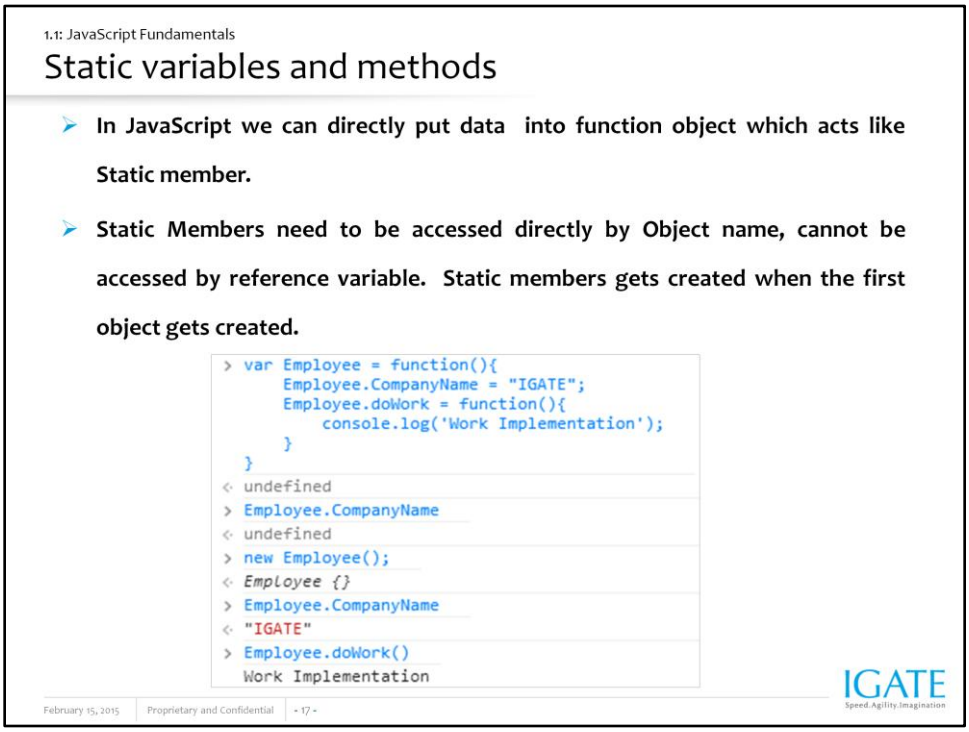
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.



In JavaScript :

private variables are declared with the 'var' keyword inside the object, and can only be accessed by private functions and privileged methods.

private functions are declared inline inside the object's constructor (or alternatively may be defined via var functionName=function(){...}) and may only be called by privileged methods (including the object's constructor).

privileged methods are declared with this.methodName=function(){...} and may invoked by code external to the object.

public properties are declared with this.variableName and may be read/written from outside the object.

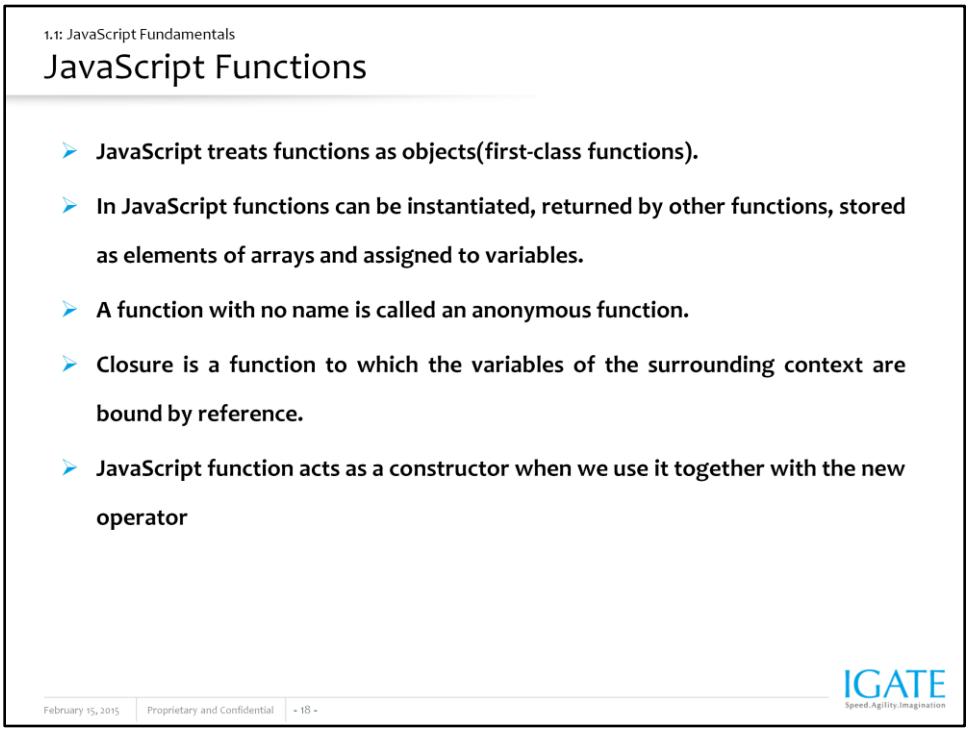
public methods are defined by Classname.prototype.methodName = function(){...} and may be called from outside the object.

prototype properties are defined by Classname.prototype.propertyName = someValue

static properties are defined by Classname.propertyName = someValue

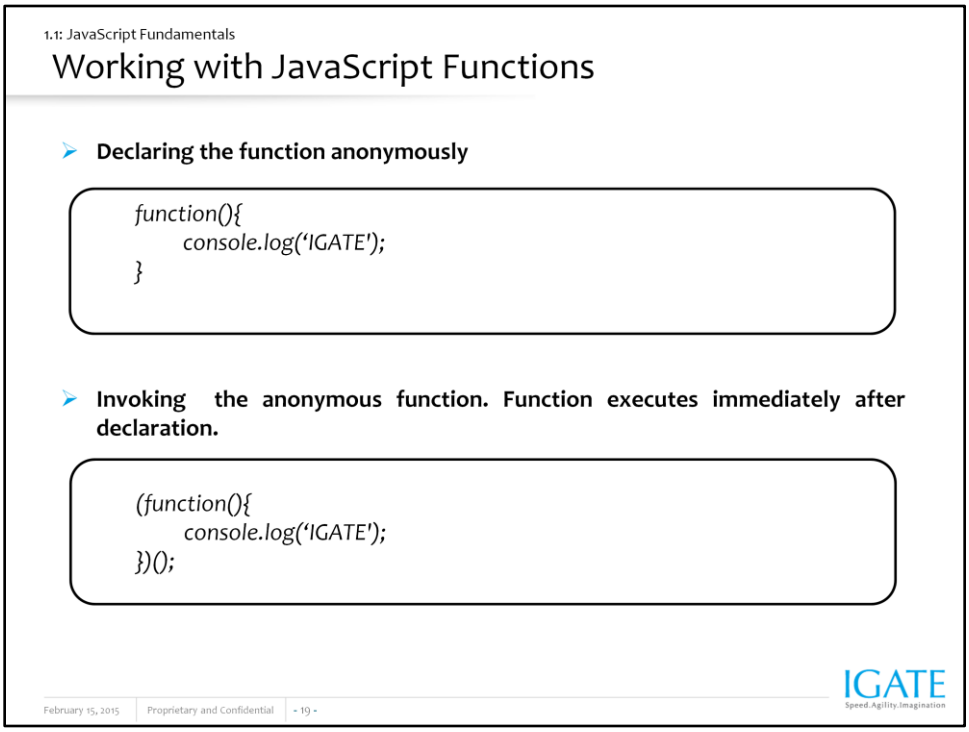
Instructor Notes:

Add instructor notes here.



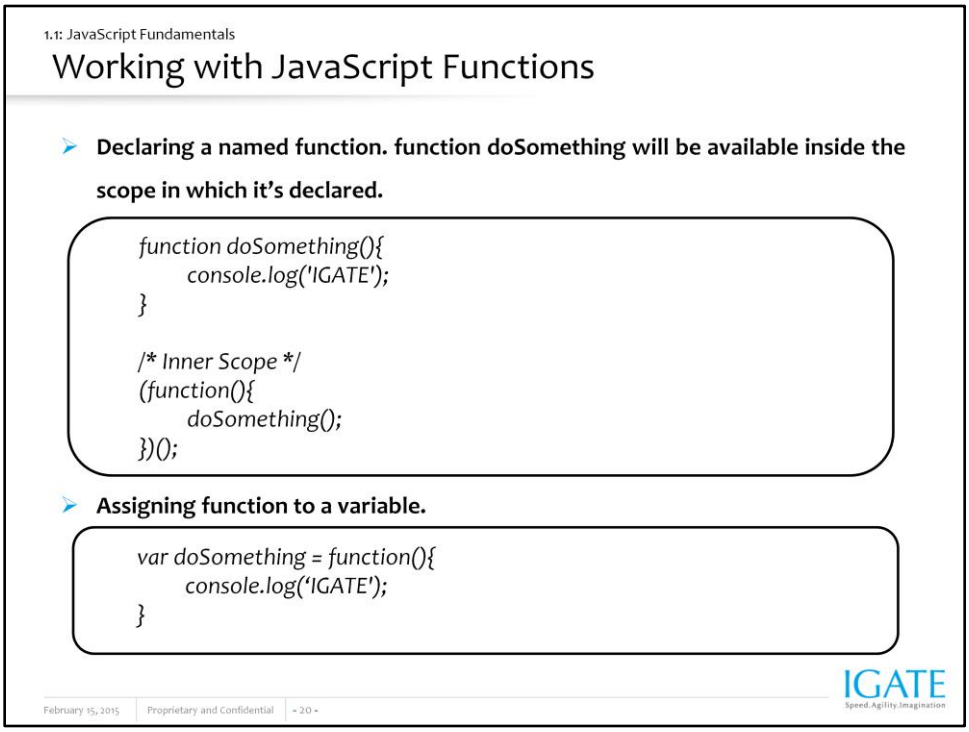
Instructor Notes:

Add instructor notes here.



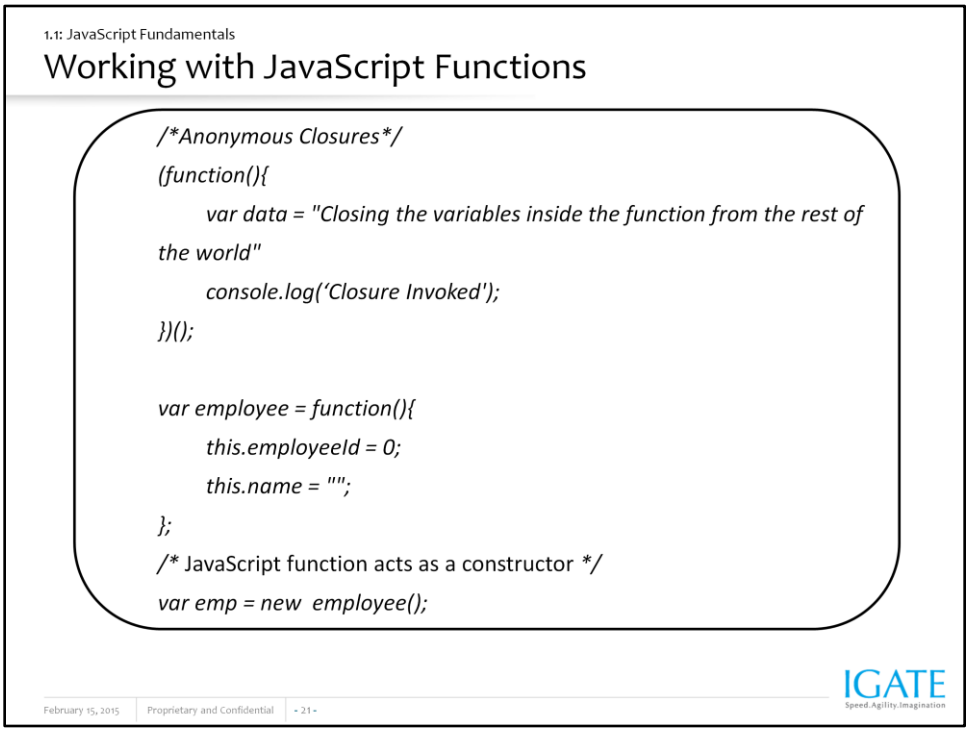
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.

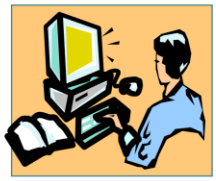


Instructor Notes:

Add instructor notes here.

Demo

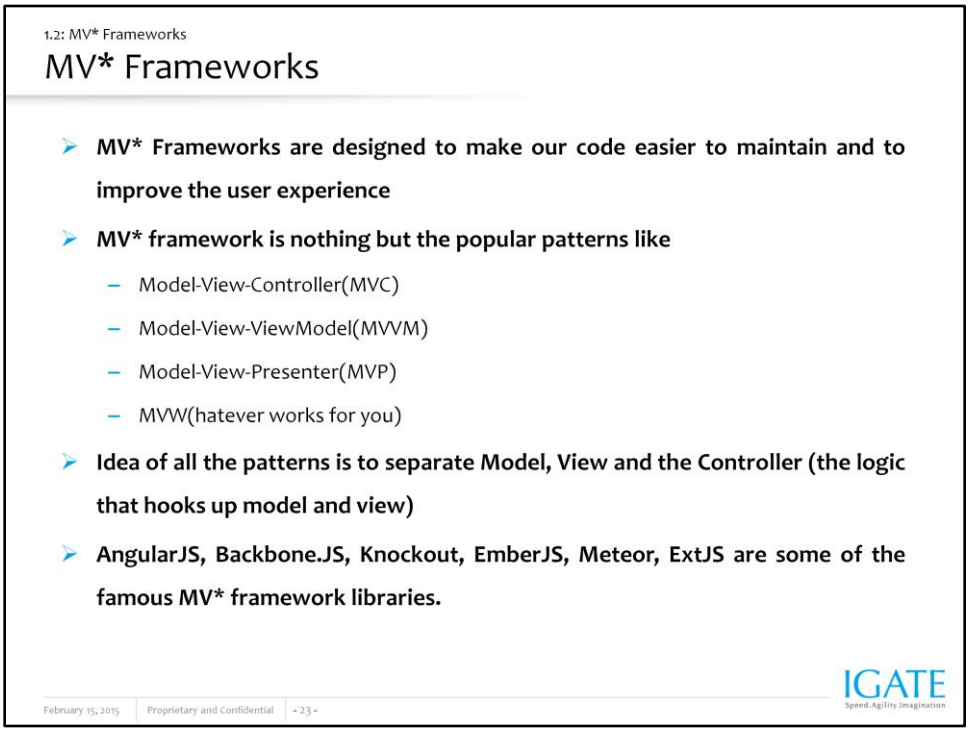
- Working-with-Javascript-Functions
- Closure-Demo



IGATE
Speed.Agility.Imagination

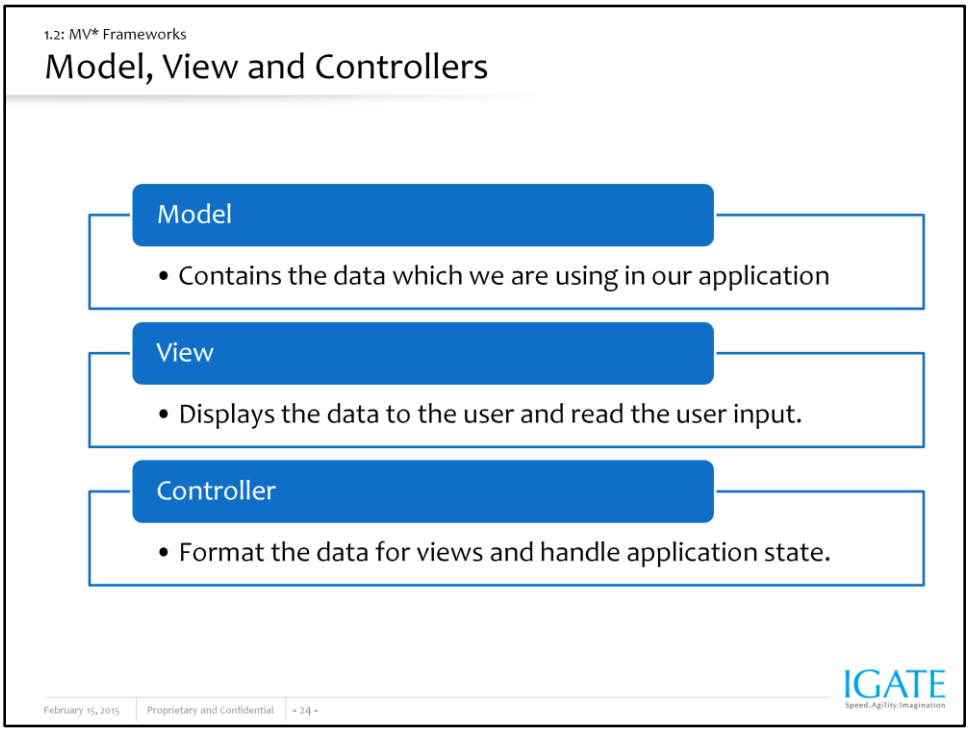
Instructor Notes:

Add instructor notes here.



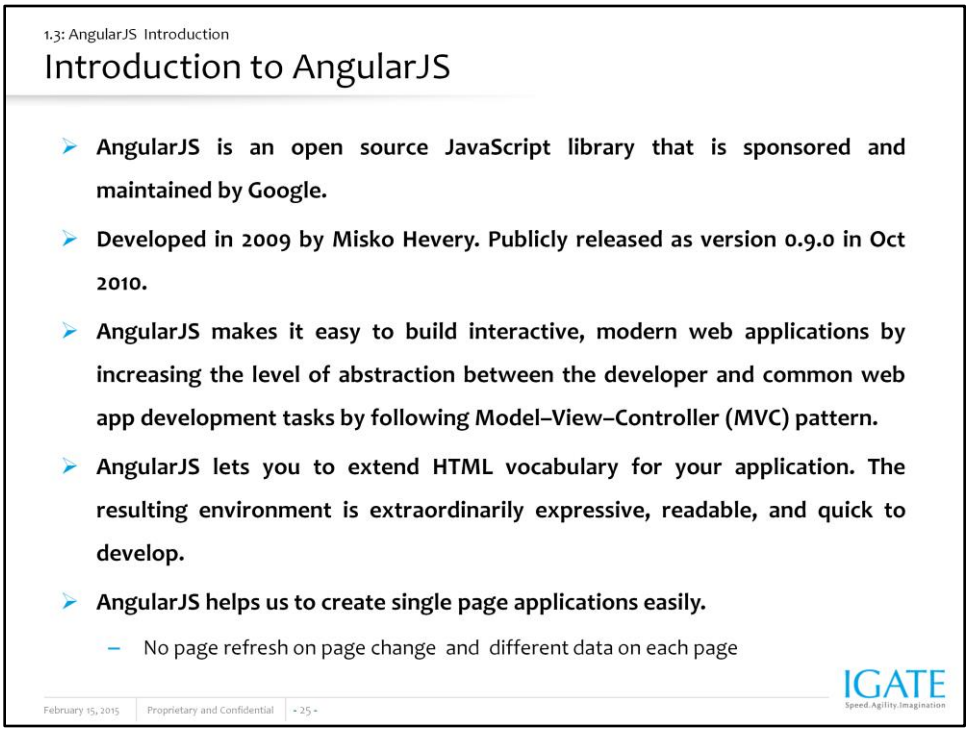
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.



AngularJS is a client-side technology, written entirely in JavaScript.

It works with the long-established technologies of the web (HTML, CSS, and JavaScript) to make the development of web apps easier and faster than ever before.

It is a framework that is primarily used to build single-page web applications. AngularJS makes it easy to build interactive, modern web applications by increasing the level of abstraction between the developer and common web app development tasks.

The AngularJS team describes it as a "structural framework for dynamic web apps."

Single page applications can be done with just JavaScript and AJAX calls, Angular will make this process easier

Instructor Notes:

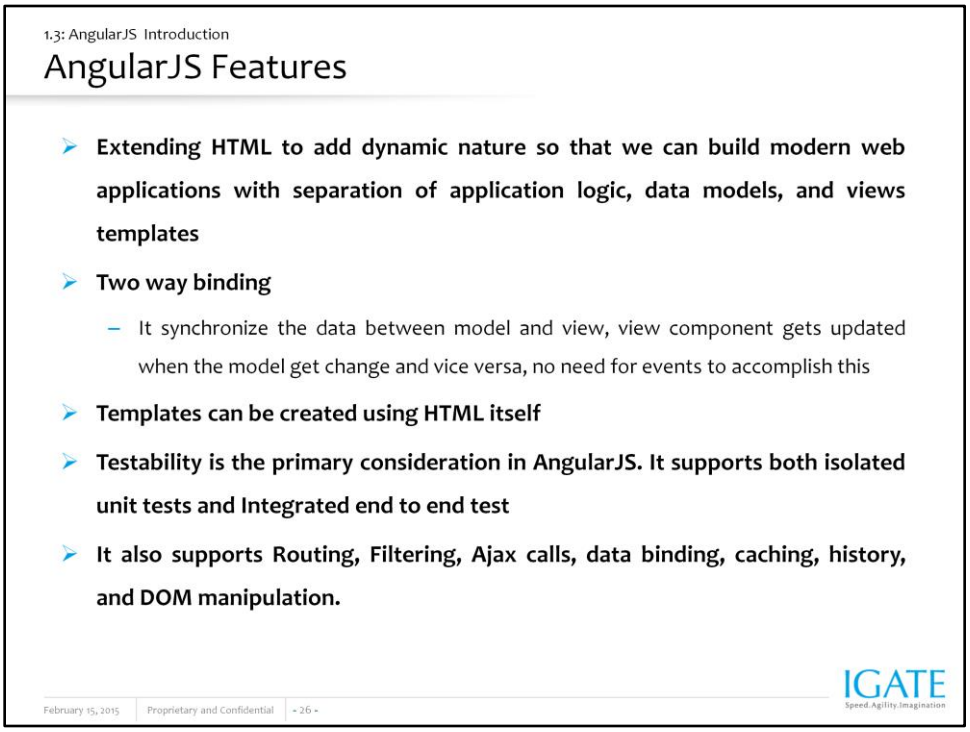
Use the demos mentioned below to demonstrate AngularJS features

•AngularJs-PhoneBookApp

•AngularJs-Service

•AngularJs-Filters

•AngularJs-Routing



HTML is designed for static webpages, but developing a dynamic site with HTML we need to do many tricks to achieve what we want. With AngularJS extending the HTML it is really simple to make a dynamic site in a proper MVC structure.

With AngularJS we can achieve
Create a template and reuse it in application multiple times.
•Can bind data to any element in two ways
•Can directly call the code-behind code in your html.
•Easily validate forms and input fields before submitting it.
•Can control complete DOM structure show/hide, changing everything with AngularJS properties.

In other JavaScript frameworks, we are forced to extend from custom JavaScript objects and manipulate the DOM from the outside. For instance, using jQuery, to add a button in the DOM, we'll have to know where we're putting the element and insert it in the appropriate place.

```
var $btn = $("<button>jQuery Button</button>");
$("#target").append($btn);
```

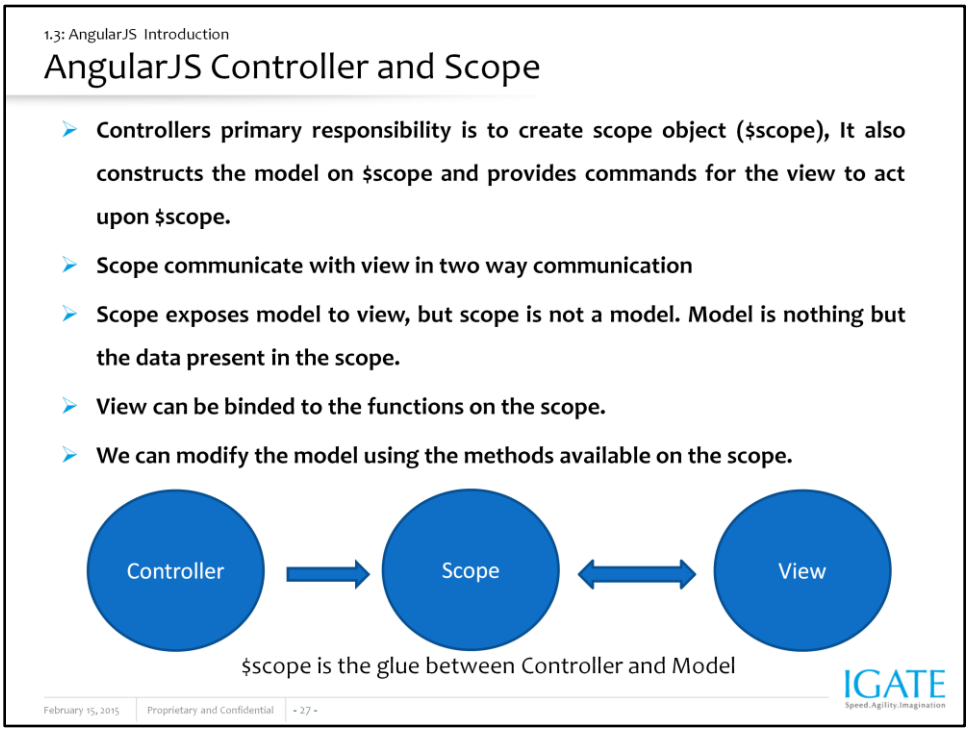
AngularJS, on the other hand, augments HTML to give it native Model-View-Controller (MVC) capabilities. It enables the developer, to encapsulate a portion of entire page as one application, rather than forcing the entire page to be an AngularJS application.

This distinction is particularly beneficial, if the web application already includes another framework or if there is a need to make a portion of the page dynamic

Instru while the rest operates as a static page or is controlled by another JavaScript framework.

Instructor Notes:

Add instructor notes here.



Scopes are a core fundamental of any Angular app. They are used all over the framework. $scope object is where we define the business functionality of the application, the methods in our controllers, and properties in the views.

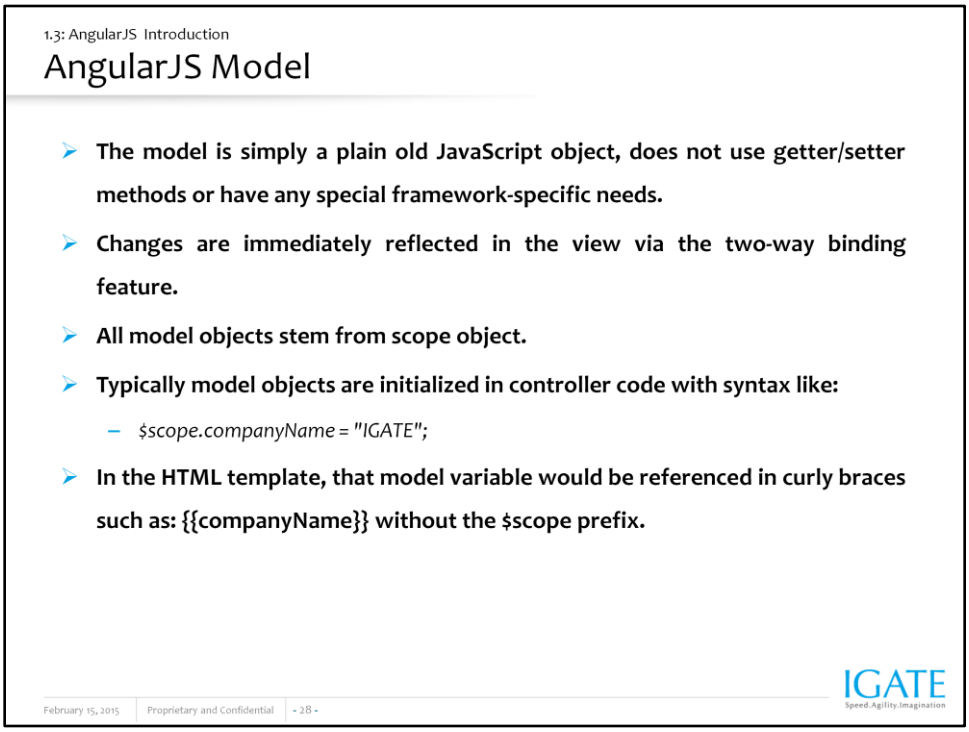
Scopes are the source of truth for the application state. Because of this live binding, we can rely on the $scope to update immediately when the view modifies it, and we can rely on the view to update when the $scope changes.

$scopes in AngularJS are arranged in a hierarchical structure so that we can reference properties on parent $scopes.

It is ideal to contain the application logic in a controller and the working data on the scope of the controller.

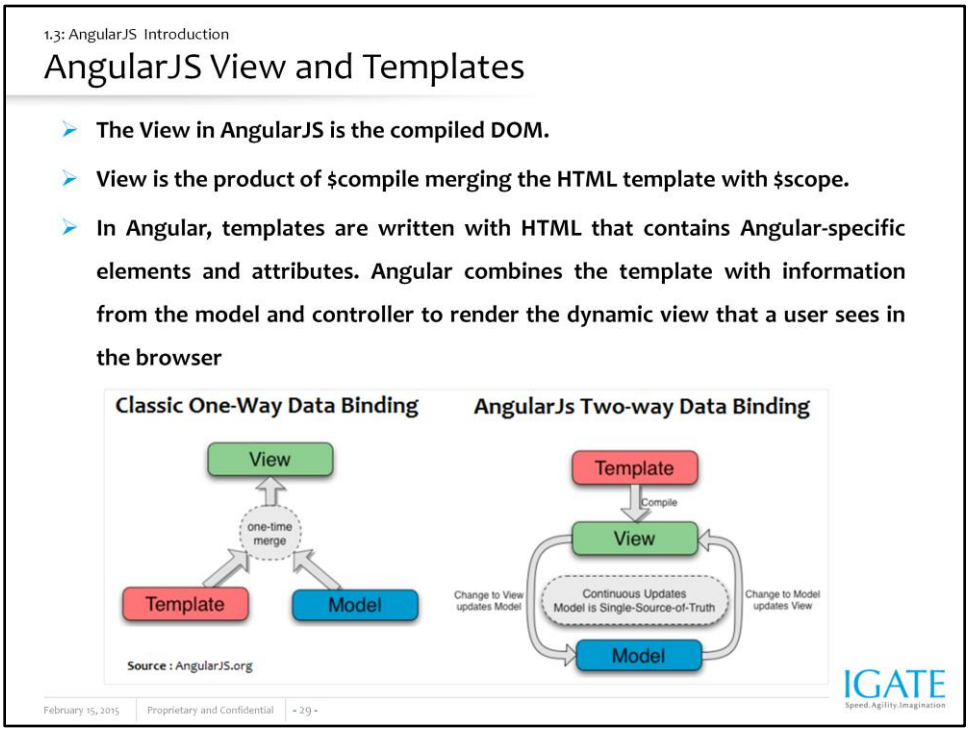
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.



Most other templating systems consume a static string template and combine it with data, resulting in a new string. The resulting text is then innerHTMLed into an element. This means that any changes to the data need to be re-merged with the template and then innerHTMLed into the DOM.

Angular is different. The Angular compiler consumes the DOM, not string templates. The result is a linking function, which when combined with a scope model results in a live view. The view and scope model bindings are transparent. The developer does not need to make any special calls to update the view. And because innerHTML is not used, you won't accidentally clobber user input. Furthermore, Angular directives can contain not just text bindings, but behavioral constructs as well.

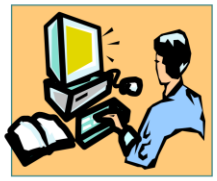
The Angular approach produces a stable DOM. The DOM element instance bound to a model item instance does not change for the lifetime of the binding. This means that the code can get hold of the elements and register event handlers and know that the reference will not be destroyed by template data merge.

Instructor Notes:

Add instructor notes here.

Demo

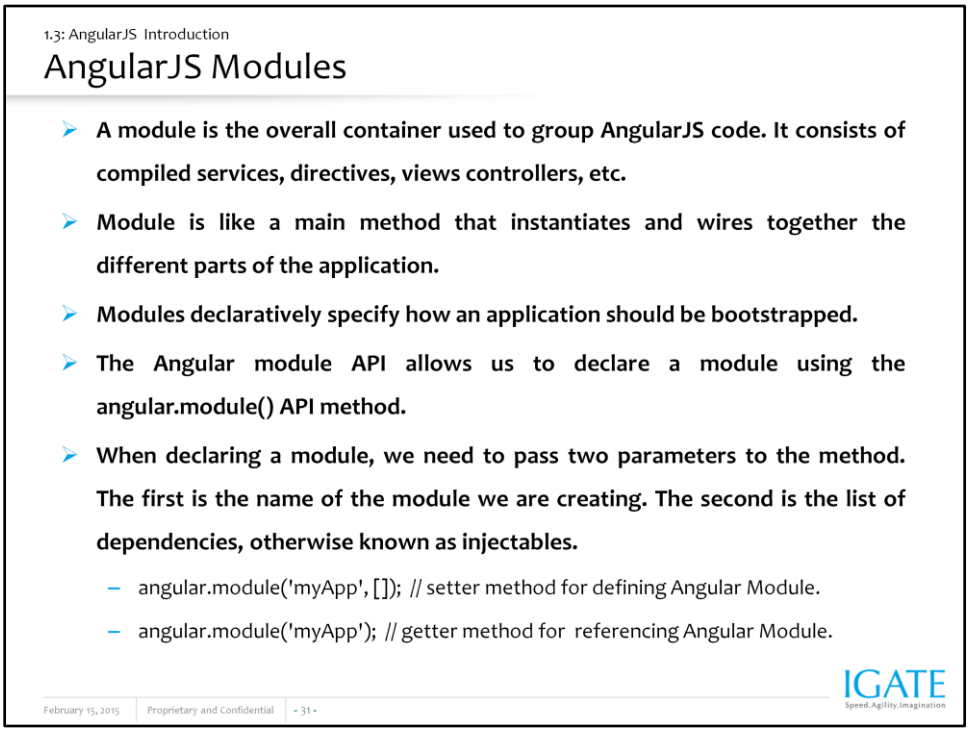
- **AngularJs-MVC**



IGATE
Speed.Agility.Imagination

Instructor Notes:

Add instructor notes here.



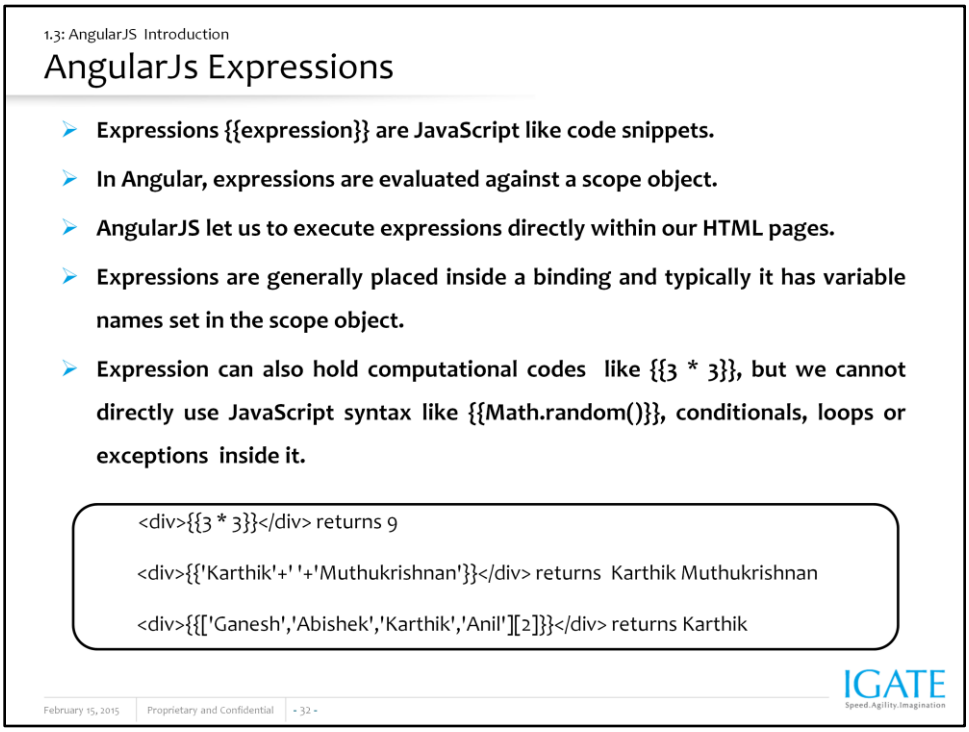
Advantages of Modules

➢Keeping our global namespace clean
➢Making tests easier to write and keeping them clean so as to more easily target isolated functionality
➢Making it easy to share code between applications
➢Allowing our app to load different parts of the code in any order

When writing large applications, we can create several different modules to contain our logic. Creating a module for each piece of functionality gives us the advantage of isolation to write and test.

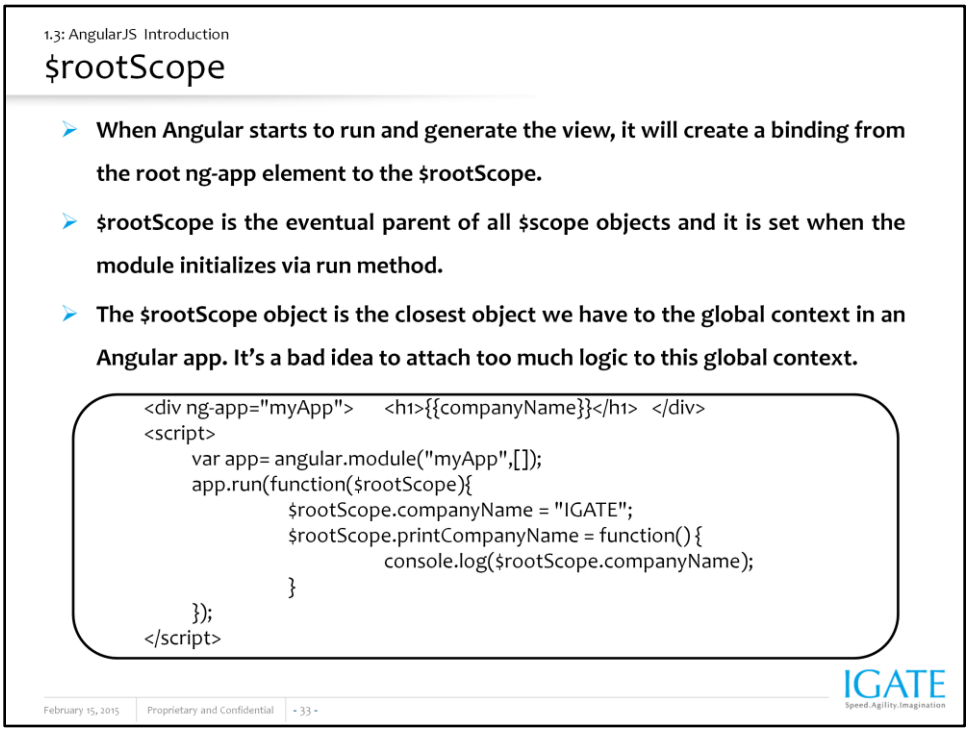
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.



Run blocks - get executed after the injector is created and are used to kickstart the application. Only instances and constants can be injected into run blocks. This is to prevent further system configuration during application run time.

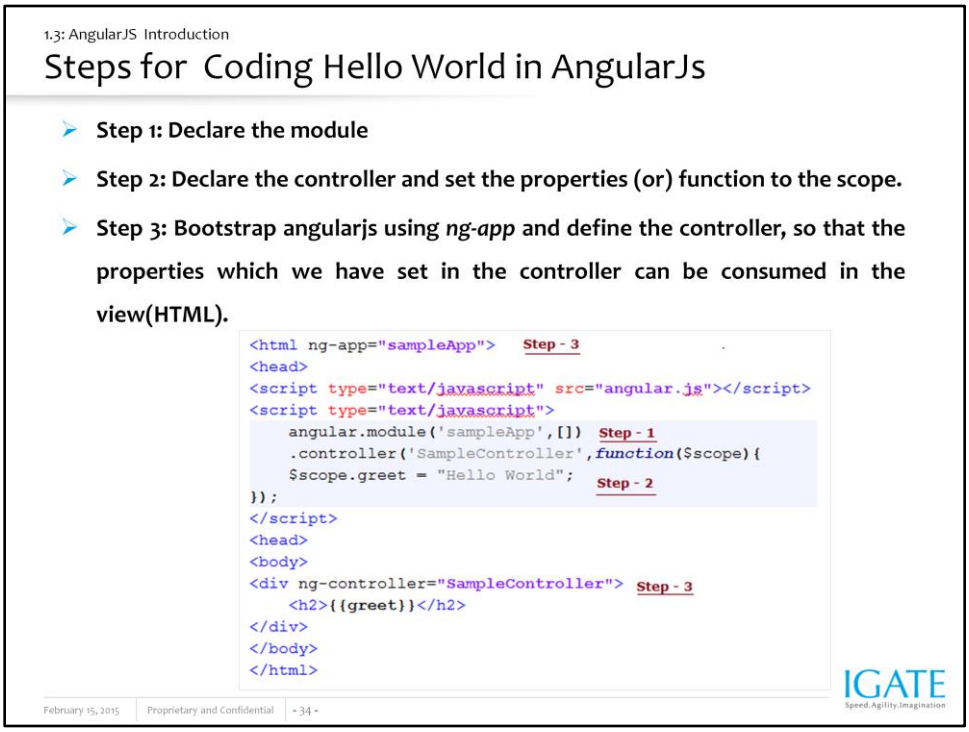
Run blocks are the closest thing in Angular to the main method. It is executed after all of the service have been configured and the injector has been created.

Variables set at the root-scope are available to the controller scope via prototypical inheritance.

```
<div ng-app="myApp" ng-controller="myCntrl">
  {{company}}
</div>

var app= angular.module("myApp",[]);
app.run(function($rootScope){
        $rootScope.companyName = "Microsoft";
        $rootScope.printCompanyName = function() {

        console.log($rootScope.companyName);
        }
});

app.controller("myCntrl",function($scope,$rootScope){
        console.debug($rootScope.companyName);
        $rootScope.companyName= "IGATE"; //Changing Value
        $scope.company = $rootScope.companyName;
});
```

Instructor Notes:

Add instructor notes here.

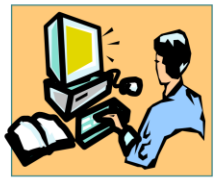


Instructor Notes:

Add instructor notes here.

Demo

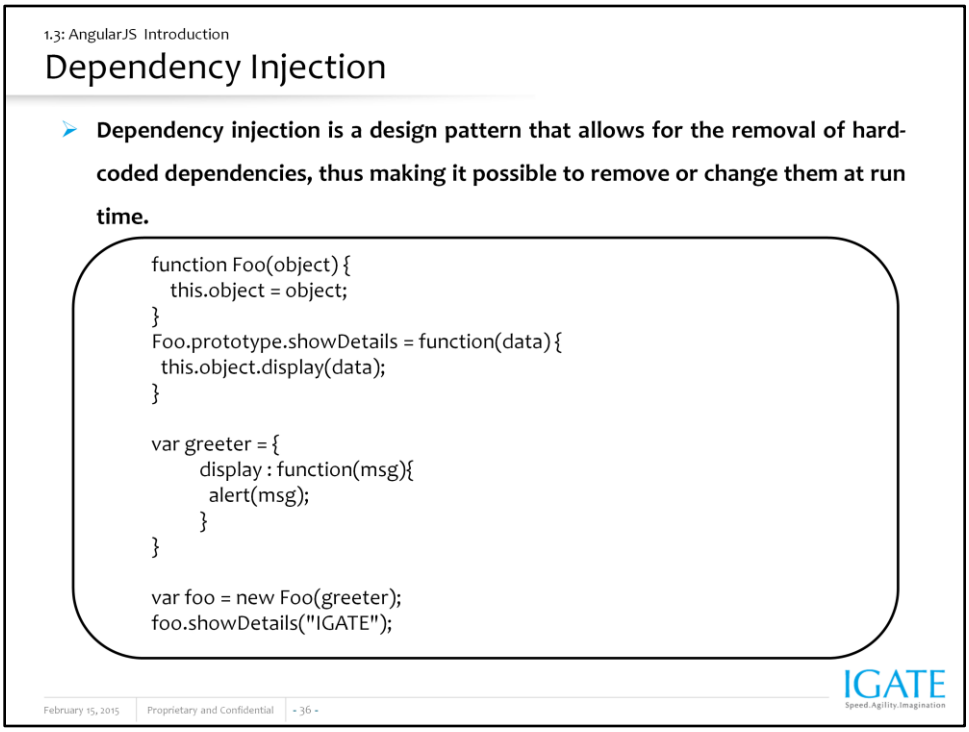
- AngularJs-Modules



IGATE
Speed.Agility.Imagination

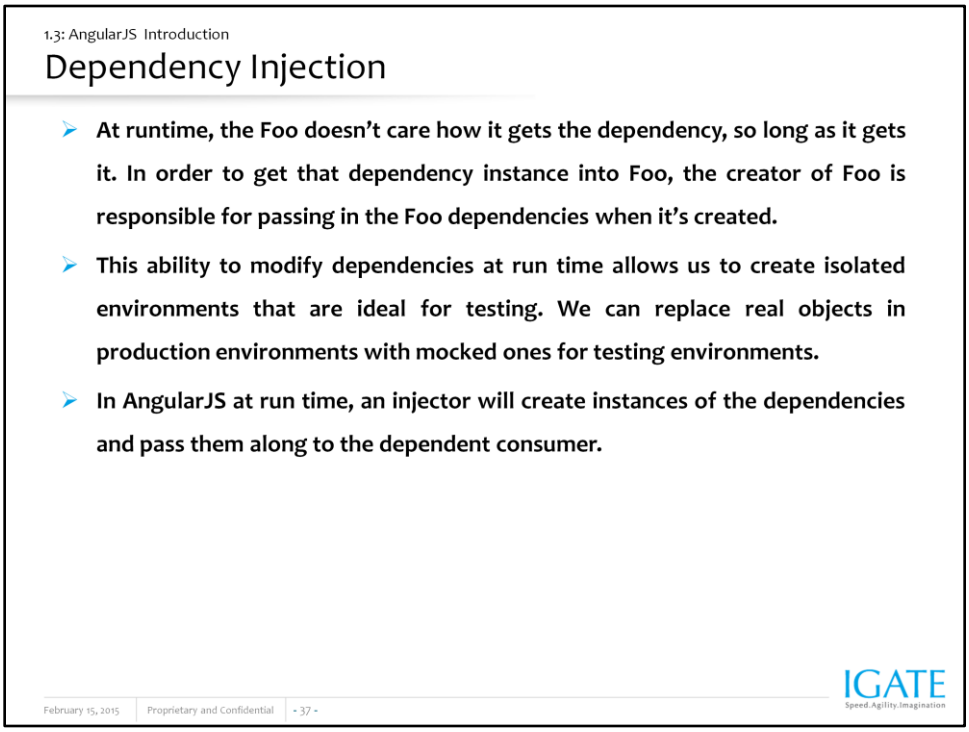
Instructor Notes:

Add instructor notes here.



Instructor Notes:

Add instructor notes here.

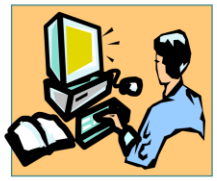


Instructor Notes:

Add instructor notes here.

Demo

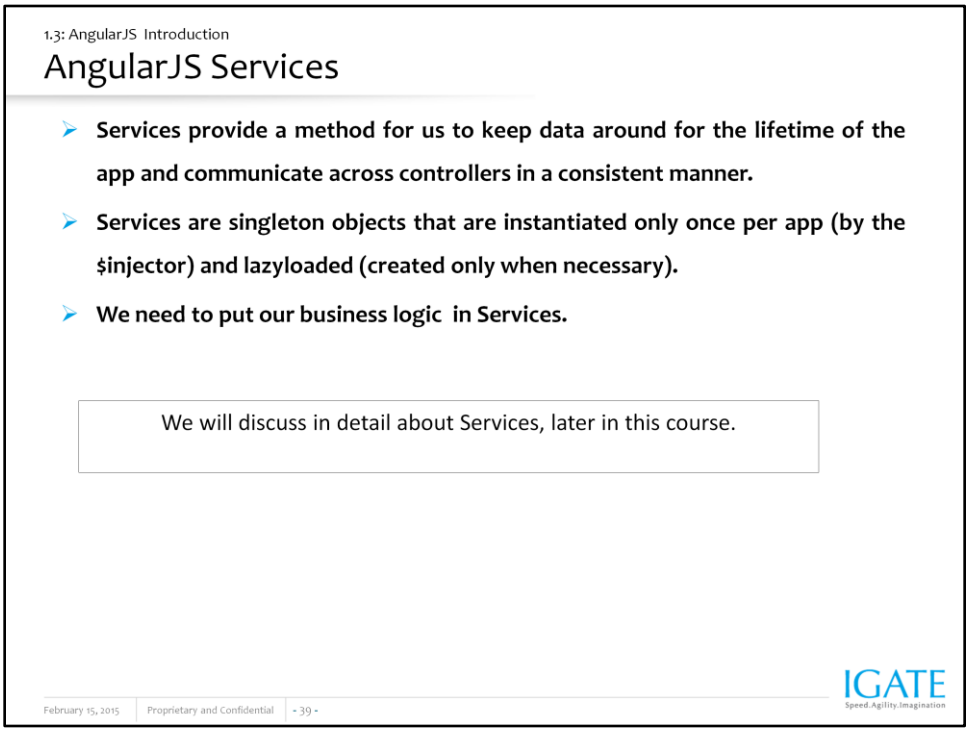
- **Dependency Demo**



IGATE
Speed.Agility.Imagination

Instructor Notes:

Add instructor notes here.



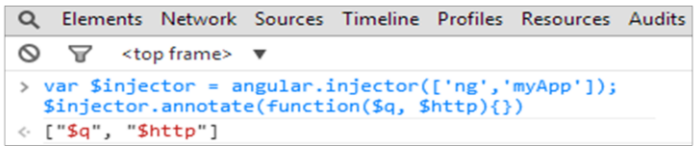
Instructor Notes:

Add instructor notes here.

1.3: AngularJS Introduction

injector Service

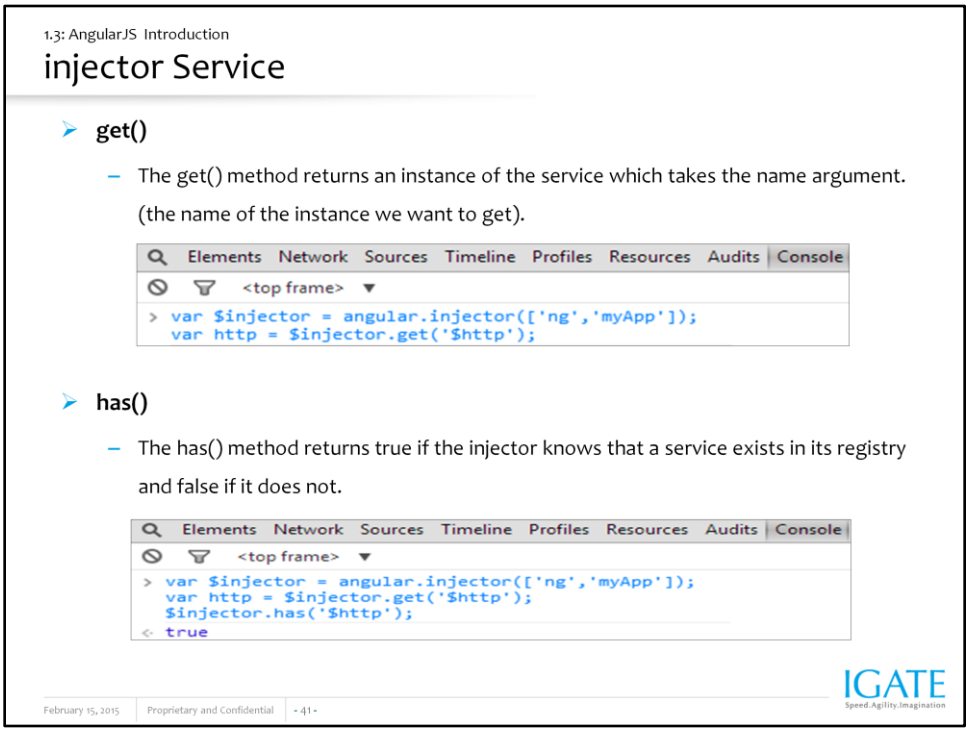
- **Angular uses the injector for managing lookups and instantiation of dependencies. We will very rarely work directly with injector service**
- **injector is responsible for handling all instantiations of our Angular components, including app modules, directives, controllers, etc.**
- **injector is responsible for instantiating the instance of the object and passing in any of its required dependencies. Injector API has following methods.**
- **annotate()**
  - The annotate() function returns an array of service names that are to be injected into the function when instantiated.



IGATE Speed.Agility.Imagination

Instructor Notes:

Add instructor notes here.



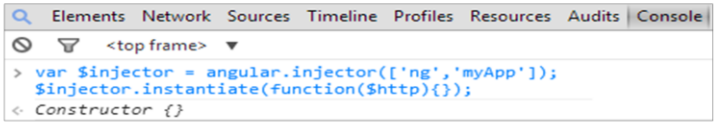
Instructor Notes:

Add instructor notes here.

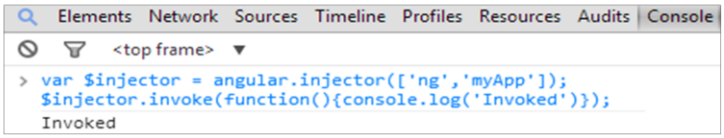
1.3: AngularJS Introduction

injector Service

- **instantiate()**
  - The instantiate() method creates a new instance of the JavaScript type. It takes a constructor and invokes the new operator with all of the arguments specified.

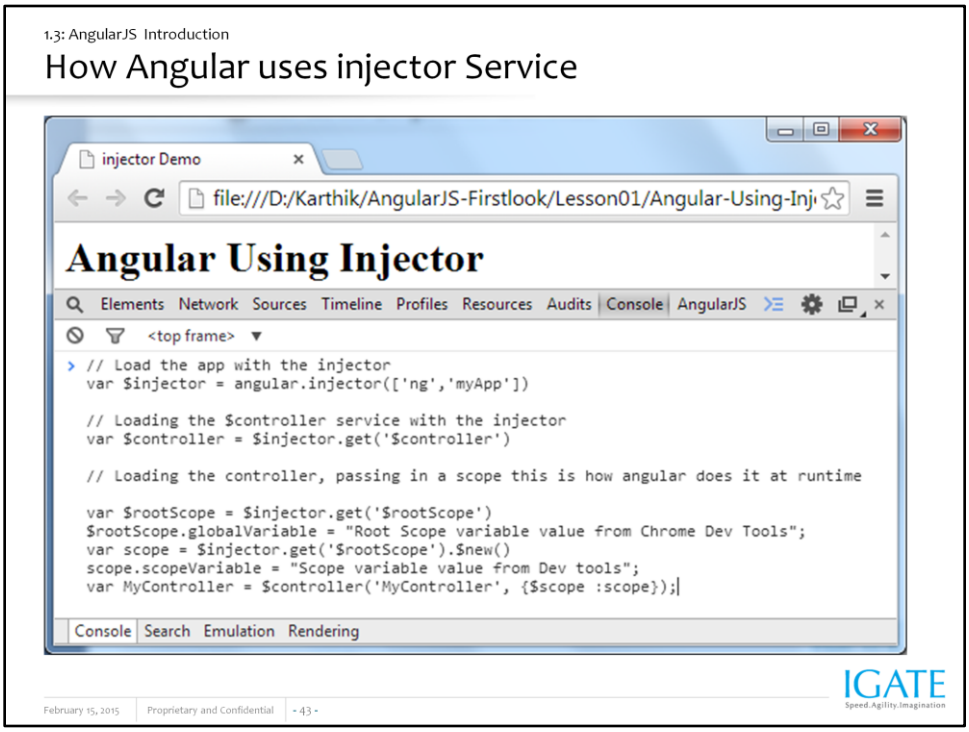


- **invoke()**
  - The invoke() method invokes the method and adds the method arguments from the $injector. The invoke() method returns the value that the fn function returns



Instructor Notes:

Add instructor notes here.



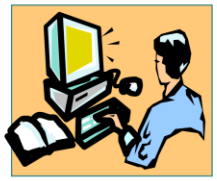

```
// Load the app with the injector
var $injector = angular.injector(['ng','myApp'])

// Loading the $controller service with the injector
var $controller = $injector.get('$controller')

// Loading the controller, passing in a scope this is how angular does
it at runtime

var $rootScope = $injector.get('$rootScope')
$rootScope.globalVariable = "Root Scope variable value from
Chrome Dev Tools";
var scope = $injector.get('$rootScope').$new()
scope.scopeVariable = "Scope variable value from Dev tools";
var MyController = $controller('MyController', {$scope :scope});
```

Instructor Notes:

Run this demo by typing the contents shown in Previous slide using Chrome Developer tools

Demo

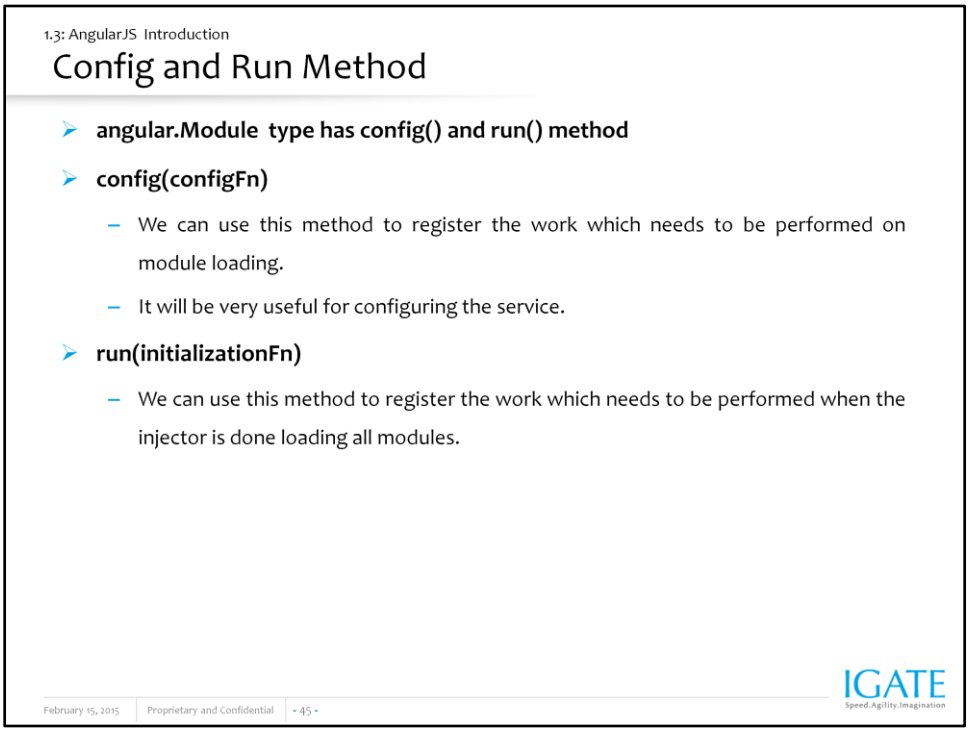
- **Angular-Using-Injector**
- **Injector-Demo**



IGATE
Speed.Agility.Imagination

Instructor Notes:

Add instructor notes here.

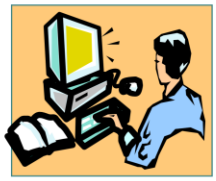


Instructor Notes:

Run this demo by typing the contents shown in Previous slide using Chrome Developer tools

Demo

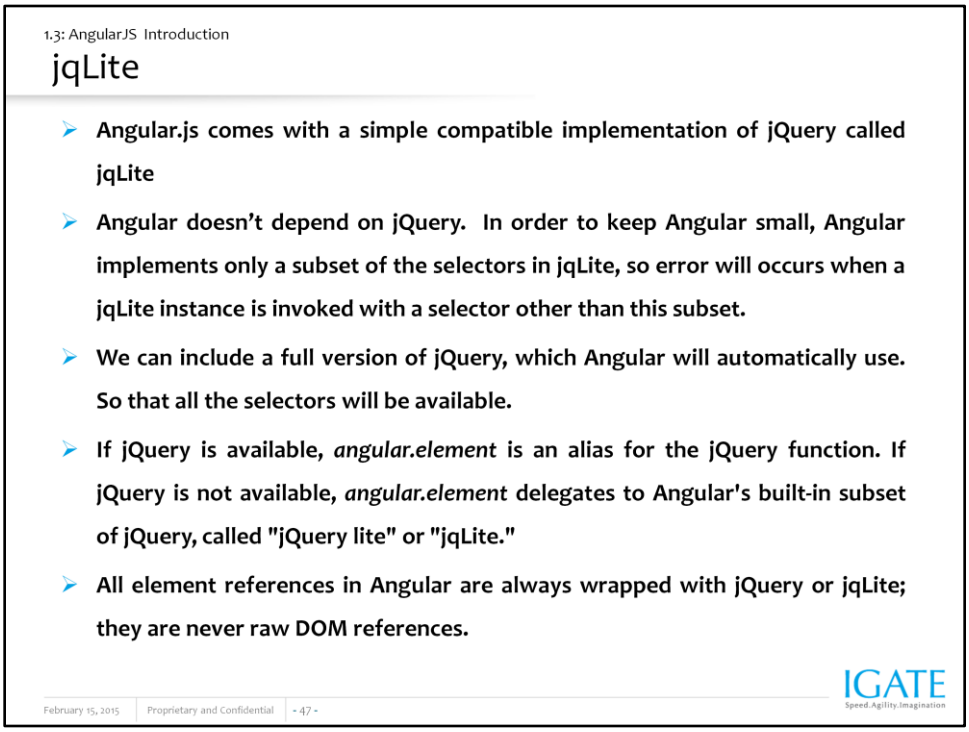
- **Config-Demo**



IGATE
Speed.Agility.Imagination

Instructor Notes:

Add instructor notes here.

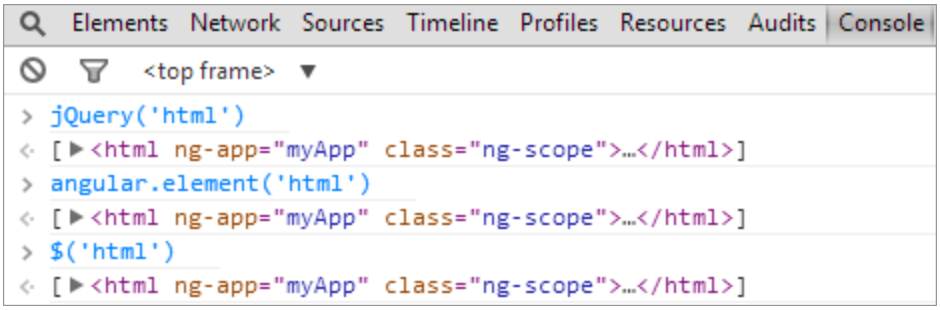


Angular's jqLite

jqLite provides only the following jQuery methods:

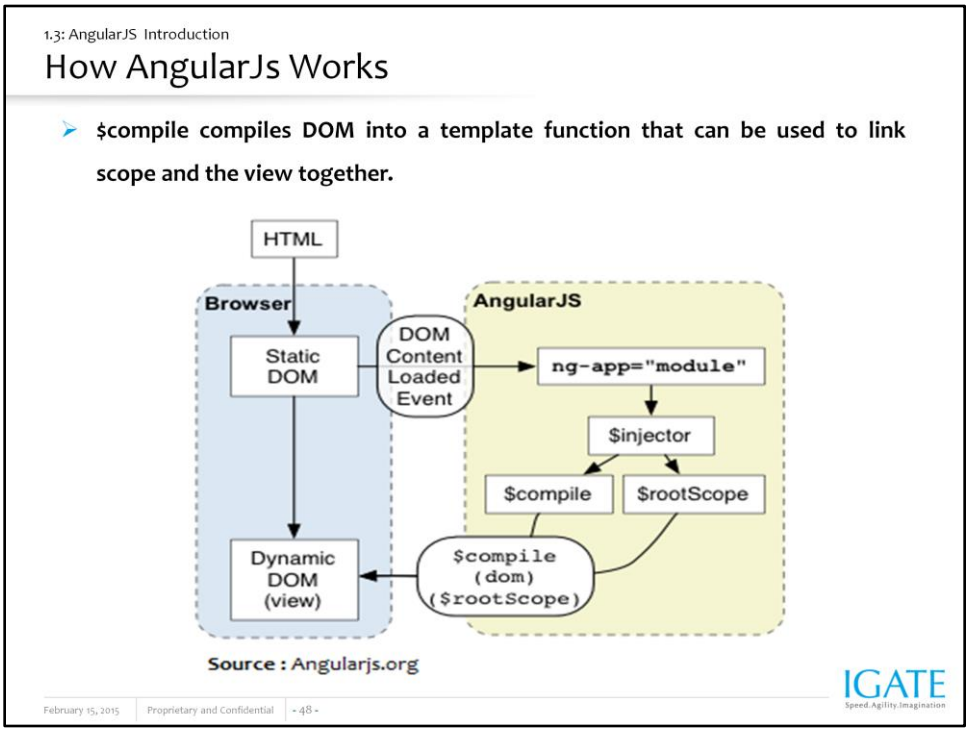
addClass(), after(), append(), attrclone(), contents(), css(), data(), detach(), empty(), eq(), hasClass(), html(), prepend(), prop(), ready(), remove(), removeAttr(), removeClass(), removeData(), replaceWith(), text(), toggleClass(), val() & wrap()

children(), parent(), next() - Does not support selectors,
find() - Limited to lookups by tag name,
bind() - Does not support namespaces, selectors or eventData,
on() - Does not support namespaces, selectors or eventData,
off() - Does not support namespaces or selectors,
one() - Does not support namespaces or selectors
unbind() - Does not support namespaces
triggerHandler() - Passes a dummy event object to handlers.



Instructor Notes:

Add instructor notes here.



Angular initializes automatically upon DOMContentLoaded event or when the angular.js script is evaluated if at that time document.readyState is set to 'complete'.

At this point Angular looks for the ng-app directive which designates your application root.

If the ng-app directive is found then Angular will:

•Load the module associated with the directive.
•Create the application injector
•Compile the DOM treating the ng-app directive as the root of the compilation.

This allows you to tell it to treat only a portion of the DOM as an Angular application.

Instructor Notes:

Add instructor notes here.

Summary

- JavaScript objects are also associative arrays
- In JavaScript, the inheritance is prototype-based.
- JavaScript treats functions as objects(first-class functions).
- JavaScript function acts as a constructor when we use it together with the new operator.
- Angular thinks of HTML as if it had been designed to build applications instead of documents.
- Angular supports unit tests and end to end tests.
- Controller is the central component in an angular application.

Add the notes here.

Instructor Notes:

Add instructor notes here.

Summary

- Creating the scope is the primary responsibility of the controller.
- $scope is the glue between Controller and Model
- View can bind to the properties as well as functions on the scope.
- Expressions only supports a subset of JavaScript. We can create an array in expression.
- Using Dependency Injection we can replace real objects in production environments with mocked ones for testing environments

IGATE
Speed.Agility.Imagination

Add the notes here.