Advanced JavaScript

This JavaScript Coding Course will teach you everything you need to master the most widely used front-end language today.

- Covering all the new Object-Oriented features introduced in ES6, this course shows you how to build large-scale web apps
- Build apps that promote scalability, maintainability, and reusability
- Learn popular Object-Oriented programming (OOP) principles and design patterns to build robust apps
- Implement Object-Oriented concepts in a wide range of front-end architectures
- Master JavaScript's OOP features, including the one's provided by ES6 specification
- Identify and apply the most common design patterns such as Singleton, Factory, Observer, Model-View-Controller, and Mediator Patterns
- Understand the SOLID principles and their benefits
- Use the acquired OOP knowledge to build robust and maintainable code
- Design applications using a modular architecture based on SOLID principles

"Any application that can be written in JavaScript will eventually be written in JavaScript."
–Atwood's Law, by Jeff Atwood

Prerequisites

This is an advanced course. You should be at the level of HTML5 / JavaScript programming before you do this course

Intended Audience:

Anybody who wants to understand and remember how to program with JavaScript using the best techniques and the most recent standards, including OO.

Further Training:

Angular

NodeJS.

Course Material

Provided

Course Contents

DAY 1

A Refresher of Objects

- Object literals
- Object constructors
- Object prototypes
- Using classes

OOP Principles

- OOP principles
- Is JavaScript Object Oriented?
- Abstraction and modeling support
- OOP principles support
- JavaScript OOP versus classical OOP

Working with Encapsulation and Information Hiding

- Encapsulation and information hiding
- Convention-based approach
- Privacy levels using closure
- A meta-closure approach
- Property descriptors
- Information hiding in ES6 classes

DAY 2

Inheriting and Creating Mixins

- Why inheritance?
- Objects and prototypes
- ES6 inheritance
- Controlling inheritance
- Implementing multiple inheritance
- Creating and using mixins

Defining Contracts with Duck Typing

- Managing dynamic typing
- Contracts and interfaces
- Duck typing
- Duck typing and polymorphism

Advanced Object Creation

- Creating objects
- Design patterns and object creation
- Creating a singleton
- An object factory
- The builder pattern
- Comparing factory and builder patterns
- Recycling objects with an object pool

DAY 3

Presenting Data to the User

- Managing user interfaces
- Presentation patterns
- The Model-View-Controller pattern
- The Model-View-Presenter pattern
- The Model-View-ViewModel pattern
- A MV* pattern comparison

Data Binding

- What is data binding?
- Implementing data binding
- The publish/subscribe pattern
- Using proxies

DAY 4

Asynchronous Programming and Promises

- Is JavaScript asynchronous?
- Writing asynchronous code
- Introducing Promises
- Using Generators

Organizing Code

- The global scope
- Creating namespaces
- The module pattern
- Module loading
- ECMAScript 6 modules

DAY 5

SOLID Principles

- Principle of OOP design
- The Single Responsibility Principle
- The Open/Closed Principle
- The Liskov Substitution Principle
- The Interface Segregation Principle
- The Dependency Inversion Principle

Modern Application Architectures

- From scripts to applications
- From old-style to Single Page Applications
- The Zakas/Osmani architecture
- Cross-cutting features and AOP
- Isomorphic applications

Duration and pricing

Pricing Group A

Certificate

Read about our certificates

Bookings

You can download the course registration form on our home page or by clicking here

Brochure

You may download a pdf copy of this page by clicking on the pdf icon at the top of the page.

Questions

Please email us

Schedule

On the calendar below. If your browser doesn't display the calendar below, please click on this link or try using Google Chrome, alternatively please enquire via our Contact Us page.

Android Programming

This Android Training Course will help you build your first working application quick-quick. You'll learn hands-on how to structure your app, design interfaces, create a database, make your app work on various smartphones and tablets, and if you want to pass the international exam AND-401, in Android, the syllabus is covered.

Prerequisites

Before attempting Android Programming on its own, you should have completed at least Beginner Java, if not Advanced Java or be very familiar with OO and advanced topics in Java or a similar OO language .

Alignment

AND-401 Android Certification. The exam is excluded, but we cover and support you in full if you want to sit for the international exam.

Course Contents

DAY 1

Welcome to Android

- The Android platform dissected
- Your development environment
- Install Java
- Build a basic app
- Activities and layouts from 50,000 feet
- Building a basic app (continued)
- Building a basic app (continued)
- You’ve just created your first Android app
- Android Studio creates a complete folder structure for you
- Useful files in your project
- Edit code with the Android Studio editors
- Run the app in the Android emulator
- Creating an Android Virtual Device
- Run the app in the emulator
- You can watch progress in the console
- What just happened?
- Refining the app

- What’s in the layout?
- activity_main.xml has two elements
- The layout file contains a reference to a string, not the string itself
- Let’s look in the strings.xml file

Interactive Apps

- You’re going to build a Beer Adviser app
- Create the project
- We’ve created a default activity and layout
- Adding components with the design editor
- activity_find_beer.xml has a new button
- Changes to the XML…
- …are reflected in the design editor
- Use string resources rather than hardcoding the text
- Change the layout to use the string resources
- Add values to the spinner
- Get the spinner to reference a string-array
- We need to make the button do something
- Make the button call a method
- What activity code looks like
- Add an onClickFindBeer() method to the activity
- onClickFindBeer() needs to do something
- Once you have a View, you can access its methods
- Update the activity code
- The first version of the activity
- Building the custom Java class
- Enhance the activity to call the custom Java class so that we can get REAL advice
- Activity code version 2
- What happens when you run the code

Multiple Activities and Intents

- Apps can contain more than one activity
- Here’s the app structure
- Create the project

- Create the second activity and layout
- Welcome to the Android manifest file
- Use an intent to start the second activity
- What happens when you run the app
- Pass text to a second activity
- Update the text view properties
- putExtra() puts extra information in an intent
- Update the CreateMessageActivity code
- Get ReceiveMessageActivity to use the information in the intent
- What happens when the user clicks the Send Message button
- How Android apps work
- What happens when the code runs
- How Android uses the intent filter
- You need to run your app on a REAL device
- Change the code to create a chooser

DAY 2

The Activity LifeCycle

- How do activities really work?
- The Stopwatch app
- The stopwatch layout code
- Add code for the buttons
- The runTimer() method
- Handlers allow you to schedule code
- The full runTimer() code
- The full StopwatchActivity code
- Rotating the screen changes the device configuration
- From birth to death: the states of an activity
- The activity lifecycle: from create to destroy
- How do we deal with configuration changes?
- What happens when you run the app
- There’s more to an activity’s life than create and destroy

- The activity lifecycle: the visible lifetime
- The updated StopwathActivity code
- What happens when you run the app
- But what if an app is only partially visible?
- The activity lifecycle: the foreground lifetime
- Stop the stopwatch if the activity’s paused
- The complete activity code
- Your handy guide to the lifecycle methods

The User Interface

- Three key layouts: relative, linear, and grid
- Positioning views relative to the parent layout
- Positioning views relative to other views
- Attributes for positioning views relative to other views
- RelativeLayout: a summary
- LinearLayout displays views in a single row or column
- Let’s change up a basic linear layout
- Adding weight to one view
- Adding weight to multiple views
- Using the android:gravity attribute: a list of values
- More values you can use with the android:layout
- gravity attribute
- The full linear layout code
- LinearLayout: a summary
- GridLayout displays views in a grid
- Adding views to the grid layout
- Let’s create a new grid layout
- Row 0: add views to specific rows and columns
- Row 1: make a view span multiple columns
- Row 2: make a view span multiple columns
- The full code for the grid layout
- GridLayout: a summary
- Layouts and GUI components have a lot in common
- Playing with views

List Views and Adapters

- Every app starts with ideas
- Categorize your ideas: top-level, category, and detail/edit activities
- Navigating through the activities
- Use ListViews to navigate to data
- We’re going to build the Starbuzz app
- The drink detail activity
- The Starbuzz app structure
- The top-level layout contains an image and a list
- The full top-level layout code
- Get ListViews to respond to clicks with a Listener
- The full TopLevelActivity code
- How to create a list activity
- Connect list views to arrays with an array adapter
- Add the array adapter to DrinkCategoryActivity
- What happens when you run the code
- How we handled clicks in TopLevelActivity
- The full DrinkCategoryActivity code
- A detail activity displays data for a single record
- Update the views with the data
- The DrinkActivity code

DAY 3

Fragments

- The Workout app structure
- The Workout class
- How to add a fragment to your project
- What fragment code looks like
- Activity states revisited
- The fragment lifecycle
- Your fragment inherits the lifecycle methods
- How to create a list fragment
- The updated WorkoutListFragment code
- Wiring up the list to the detail
- Using fragment transaction

- The updated MainActivity code
- The WorkoutDetailFragment code
- The phone and tablet app structures
- The different folder options
- The MainActivity phone layout
- The full DetailActivity code
- The revised MainActivity code

Nested Fragments

- Creating nested fragments
- The StopwatchFragment code
- The StopwatchFragment layout
- getFragmentManager() creates transactions at the activity lavel
- Nested fragments need nested transactions
- The full WorkoutDetailFragment code
- Why does the app crash if you press a button?
- the StopwatchFragment layout code
- Make the fragment implement OnClickListener
- Attach the OnClickListener to the buttons
- The StopwatchFragment code
- The WorkoutDetailFragment code

DAY 4

Action Bars

- Great apps have a clear structure
- Different types of navigation
- Let’s start with the action bar
- The Android support libraries
- Your project may include support libraries
- We’ll get the app to use up to date themes
- Apply a theme in AndroidManifest.xml
- Define styles in style resource files
- Set the default theme in styles.xml
- What happens when you run the app

- Adding action items to the action bar
- The menu resource file
- The menu showAsAction attribute
- Add a new action item
- Create OrderActivity
- Start OrderActivity with the Create Order action item
- The full MainActivity.java code
- Sharing content on the action bar
- Specify the content with an intent
- The full MainActivity.java code
- Enabling Up navigation
- Setting an activity’s parent
- Adding the Up button

Navigation Drawers

- The Pizza app revisited
- Navigation drawers deconstructed
- The Pizza app structure
- Create TopFragment
- Create PizzaFragment
- Create PastaFragment
- Create StoresFragment
- Add the DrawerLayout
- The full code for activity_main.xml
- Initialize the drawer’s list
- Changing the action bar title
- Closing the navigation drawer
- The updated MainActivity.java code
- Using an ActionBarDrawerToggle
- Modifying action bar items at runtime
- The updated MainActivity.java code
- Enable the drawer to open and close
- Syncing the ActionBarDrawerToggle state
- The updated MainActivity.java code
- Dealing with configuration changes
- Reacting to changes on the back stack

- Adding tags to fragments

SQLite Databases

- Back to Starbuzz
- Android uses SQLite databases to persist data
- Android comes with SQLite classes
- The current Starbuzz app structure
- The SQLite helper manages your database
- The SQLite helper
- Create the SQLite helper
- Inside a SQLite database
- You create tables using Structured Query Language (SQL)
- Insert data using the insert() method
- Update records with the update() method
- Multiple conditions
- The StarbuzzDatabaseHelper code
- What the SQLite helper code does
- What if you need to change the database?
- SQLite databases have a version number
- Upgrading the database: an overview
- How the SQLite helper makes decisions
- Upgrade your database with onUpgrade()
- Downgrade your database with onDowngrade()
- Let’s upgrade the database
- Upgrading an existing database
- Renaming tables
- The full SQLite helper code
- The SQLite helper code (continued)
- What happens when the code runs

DAY 5

Cursors and Asynch Tasks

- The current DrinkActivity code
- Specifying table and columns
- Applying multiple conditions to your query

- Using SQL functions in queries
- Navigating cursors
- The DrinkActivity code
- Add favorites to DrinkActivity
- The DrinkActivity code
- The new top-level activity code
- The revised TopLevelActivity.java code
- The onPreExecute() method
- The doInBackgound() method
- The onProgressUpdate() method
- The onPostExecute() method
- The AsyncTask class
- The DrinkActivity.java code

Services

- The started service app
- The IntentService from 50,000 feet
- How to log messages
- The full DelayedMessageService code
- The full DelayedMessageService.java code
- How you use the notification service
- Getting your notification to start an activity
- Send the notification using the notification service
- The full code for DelayedMessageService.java
- The steps needed to create the OdometerService
- Define the Binder
- The Service class has four key methods
- Add the LocationListener to the service
- Registering the LocationListener
- The full OdometerService.java code
- Update AndroidManifest.xml
- Update MainActivity’s layout
- Create a ServiceConnection
- Bind to the service when the activity starts
- Display the distance traveled
- The full MainActivity.java code

Material Design

- Welcome to Material Design
- The Pizza app structure
- Create the CardView
- The full card_captioned_image.xml code
- Create the basic adapter
- Define the adapter’s ViewHolder
- Create the ViewHolders
- Each card view displays an image and a caption
- Add the data to the card views
- The full code for CaptionedImagesAdapter.java
- Create the recycler view
- Add the RecyclerView to the layout
- The PizzaMaterialFragment.java code
- A RecyclerView uses a layout manager to arrange its views
- Specifying the layout manager
- The full PizzaMaterialFragment.java code
- Get MainActivity to use the new PizzaMaterialFragment
- What happens when the code runs
- Create PizzaDetailActivity
- What PizzaDetailActivity.java needs to do
- The code for PizzaDetailActivity.java
- Getting a RecyclerView to respond to clicks
- Add the interface to the adapter
- Implement the listener in PizzaMaterialFragment.java
- Bring the content forward
- The full code for fragment_top.xml
- The full code for TopFragment.java

Duration and pricing

Price Group A

Certificate

1. Upon completion of this course we will issue you with

attendance certificate to certify your attendance.
2. You may sit for our competency assessment test and on passing you will obtain our competency certificate.
3. Our competency assessment can be booked and taken by someone who has not attended the course at a cost of R950.

Bookings
You can download the course registration form on our home page or by clicking here

Brochure
You may download a pdf copy of this page by clicking on the pdf icon at the top of the page.

Questions
Please email us

Schedule
On the calendar below. If your browser doesn't display the calendar below, please click on this link or try using Google Chrome, alternatively please enquire via our Contact Us page.

Angular

Our **Angular Training Course** is intended for front-end developers who are familiar with Javascript and want to master Angular to move a gear up. Angular is the leading framework for building dynamic JavaScript applications that take advantage of the capabilities of modern browsers and devices. The course is updated for Angular 6 .

Prerequisites

JavaScript

Object – Orientation as taught in Beginner Java / Beginner Python

Further training

Node.JS

MEAN Stack

Flutter / Dart

Intended Audience

This course is for developers with basic familiarity with HTML, CSS, Javascript and object-oriented programming like Java Beginner, Python Beginner, C# Beginner. No TypeScript or AngularJS experience needed. Developers who found that JavaScript alone is inadequate to develop a rich, fully-fledged front-end. Our Angular Training Course starts with the MVC pattern and the many benefits that can be gained from separating your logic and presentation code. We then start from the nuts-and-bolts and building up to the most advanced and sophisticated features in Angular, going in-depth.

After this course

You should be able to :

- Develop a state-of-the-art web app connecting to a Firebase back-end or any REST-based API back-end like Java, .NET, PHP or Python.
- Become a front-end (Angular) developer

DAY 1

Introduction

- What is Angular?
- Overview of Angular Apps
    - Modules
    - Components
    - Services
    - Directives
- Setting Up
    - Installing Node
    - Installing TypeScript
    - Installing Typings
    - Installing Angular CLI
    - TypeScript Editor
    - Chrome Browser
- Creating a New Project with Angular CLI
    - Project File Review
- Editing Our First Angular Component

Creating and Using Components

- Creating Our First Component
- Using Our Created Component
    - app.module.ts
- Component Templates
    - Displaying a line with *ngFor
- Services
- Dependency Injection
    - providers

Bindings

- Property Binding
- CSS Class Binding
- Binding to User Input Events
- Example Application

DAY 2

Working with Components

- Input Properties
- Styles
- Example Application

Conditional Rendering, Pipes and Ng-Content

- ngIf
    - ngIfElse
- ngSwitch
- Pipes
- Custom Pipes
- ng-content
    - Multiple Insertion Points

Template Driven Forms

- Create the User Model Class
- Revising app.module.ts
- Creating an initial HTML Form Template
- Using *ngFor to Display Options
- Two-way data binding with ngModel
    - Show and Hide Validation Error messages
- Showing Specific Validation Errors
- Submit the form with ngSubmit
- Getting Submitted Values

DAY 3

Model Driven Forms

- Building a Basic Login Form
- Creating Controls Explicitly
    - Implementing Validation
    - Submitting the Form
    - app.component.ts
- Using Formbuilder
- Implementing Custom Validation
    - login.component.ts
    - login.component.html

- Validating Upon Form Submit
    - login.component.html

Introduction To Observables

- Obervables
- Creating an Observable from DOM events
- Observable Operators
    - filter operator
    - debounceTime Operator
    - distinctUntilChanged Operator

Getting Data From RESTful APIs with Observables

- GitHubRESTful API
- Getting Data
- Dependency Injection
- ngOnInit
- Showing a Loader Icon
- Implementing a GitHub Results Display page

DAY 4

Routing

- Enabling Routing
- Setting Up Our Routes
- Router Outlet and Links
    - Router Outlet
    - Router Links
    - Adding a new Router Link
- Route Parameters
    - Specifying Route Parameters
    - Retrieving Route Parameters
- Imperative Navigation
- Route Guards
    - RouteGuard CanActivate
    - RouteGuard CanDeactivate
    - app.routing.ts

    - AppModule

Structuring Large Apps With Modules

- NgModule
- Restructuring
- Restructuring GitHubModule
- Restructuring LoginModule
- Restructuring AppModule
    - app.module.ts
- Restructuring Routes
    - app.routing.ts
    - app.module.ts

DAY 5

C.R.U.D. with Firebase

- About Firebase
- Using Firebase
- Adding Firebase to our Angular App
    - app.module.ts
    - app.component.ts
- Working with a Firebase Database
- Displaying List of Users
- Adding a user
- AngularFirestore Add
    - dirty tracking
- Retrieving a Single User and Deletion
- Deleting a User
- Populating the Form on Edit
- Updating a user
- Working with multiple user signup and authentication

Own Project

==================

Duration and pricing

In Pricing Group A

Certificate

Read about our **certificates here**.

Bookings

You can download the course registration form on our home page or by clicking here

Brochure
You may download a pdf copy of this page by clicking on the pdf icon at the top of the page.

Questions
Please email us

Schedule
On the calendar below. If your browser doesn't display the calendar below, please click on this link or try using Google Chrome, alternatively please enquire via our Contact Us page.

ASP.NET CORE MVC

ASP.NET CORE MVC

Develop cloud-ready web applications using Microsoft's latest framework, ASP.NET Core MVC

Prerequisites / Further Training

- Introduction to Programming
- Beginner C#.NET (C# Fundamentals and Object – Orientation)

- Advanced C#.NET (Building practical Apps)
- SQL Querying (Database design and querying)
- *ASP.NET Core MVC (Web App Development)*
- Entity Framework

Also have a look at our .NET Bootcamp and save

Intended Audience

This course is useful for C# or ASP.NET Forms programmer who wants to learn the MVC (Model View Controller) methodology.

This course covers all the topics ot the Microsoft Exam for ASP.NET MVC. After completing this course you should be able to prepare for the Microsoft exam and understand all the concepts to pass quickly, just by doing a few mock exams.

Further Training

You may consider doing the certification exam courses:

MCPD – Web Apps

Course Material

Included in the course price.

Course Contents

Day1:

A First MVC Application

- Adding the First Controller
- Understanding Routes
- Rendering Web Pages
- Creating and Rendering a View
- Adding Dynamic Output
- Creating a Simple Data-Entry Application
- Setting the Scene
- Designing a Data Model

- Linking Action Methods
- Building the Form
- Setting the Start URL
- Handling Forms
- Adding Validation
- Styling the Content

The MVC Pattern

- The History of MVC
- Understanding the MVC Pattern
- Understanding the Domain Model
- The ASP.NET Implementation of MVC
- Comparing MVC to Other Patterns
- Building Loosely Coupled Components
- Using Dependency Injection
- Using a Dependency Injection Container
- Getting Started with Automated Testing
- Understanding Unit Testing
- Understanding Integration Testing

Essential Language Features

- Using Automatically Implemented Properties
- Using Object and Collection Initializers
- Using Extension Methods
- Applying Extension Methods to an Interface
- Creating Filtering Extension Methods
- Using Lambda Expressions
- Using Automatic Type Inference
- Using Anonymous Types
- Performing Language Integrated Queries
- Understanding Deferred LINQ Queries
- Using Async Methods
- Applying the async and await Keywords

Working with Razor

- Preparing the Example Project

- Defining the Model
- Defining the Controller
- Creating the View
- Working with the Model Object
- Working with Layouts
- Creating the Layout
- Applying a Layout
- Using a View Start File
- Shared Layouts
- Using Razor Expressions
- Inserting Data Values
- Setting Attribute Values
- Using Conditional Statements
- Enumerating Arrays and Collections
- Dealing with Namespaces

Essential Tools for MVC

- Creating the Model Classes
- Adding the Controller
- Adding the View
- Using Ninject
- Understanding the Problem
- Adding Ninject to the Visual Studio Project
- Getting Started with Ninject
- Setting up MVC Dependency Injection
- Creating Chains of Dependency
- Specifying Property and Constructor Parameter Values
- Using Conditional Binding
- Setting the Object Scope
- Unit Testing with Visual Studio
- Creating the Unit Test Project
- Creating the Unit Tests
- Running the Unit Tests (and Failing)
- Implementing the Feature
- Testing and Fixing the Code
- Using Moq

- Understanding the Problem
- Adding Moq to the Visual Studio Project
- Adding a Mock Object to a Unit Test
- Creating a More Complex Mock Object

Day2:

A Real Application

- Creating the Visual Studio Solution and Projects
- Installing the Tool Packages
- Adding References Between Projects
- Setting Up the DI Container
- Starting the Domain Model
- Creating an Abstract Repository
- Making a Mock Repository
- Displaying a List of Products
- Adding a Controller
- Adding the Layout, View Start File and View
- Setting the Default Route
- Preparing a Database
- Creating the Database
- Defining the Database Schema
- Adding Data to the Database
- Creating the Entity Framework Context
- Creating the Product Repository
- Adding Pagination
- Displaying Page Links
- Improving the URLs
- Styling the Content
- Installing the Bootstrap Package
- Applying Bootstrap Styles to the Layout
- Creating a Partial View

Navigation

- Adding Navigation Controls
- Filtering the Product List

- Refining the URL Scheme
- Building a Category Navigation Menu
- Correcting the Page Count
- Building the Shopping Cart
- Defining the Cart Entity
- Adding the Add to Cart Buttons
- Implementing the Cart Controller
- Displaying the Contents of the Cart

The Shopping Cart

- Using Model Binding
- Creating a Custom Model Binder
- Completing the Cart
- Removing Items from the Cart
- Adding the Cart Summary
- Submitting Orders
- Extending the Domain Model
- Adding the Checkout Process
- Implementing the Order Processor
- Registering the Implementation
- Completing the Cart Controller
- Displaying Validation Errors
- Displaying a Summary Page

Mobile Interface

- Putting Mobile Web Development in Context
- Doing Nothing (Or As Little As Possible)
- Using Responsive Design
- Creating a Responsive Header
- Creating a Responsive Product List
- Creating Mobile Specific Content
- Creating a Mobile Layout
- Creating the Mobile Views

Administration Section

- Adding Catalog Management

- Creating a CRUD Controller
- Creating a New Layout
- Implementing the List View
- Editing Products
- Creating New Products
- Deleting Products

Day3:

Security & Finishing Touches

- Securing the Administration Controller
- Creating a Basic Security Policy
- Applying Authorization with Filters
- Creating the Authentication Provider
- Creating the Account Controller
- Creating the View
- Image Uploads
- Extending the Database
- Enhancing the Domain Model
- Creating the Upload User Interface Elements
- Saving Images to the Database
- Implementing the GetImage Action Method
- Displaying Product Images

Overview of MVC Projects

- Working with Visual Studio MVC Projects
- Creating the Project
- Understanding MVC Conventions
- Debugging MVC Applications
- Preparing the Example Project
- Launching the Visual Studio Debugger
- Causing the Visual Studio Debugger to Break
- Using Edit and Continue
- Using Browser Link

URL Routing

- Creating the Controllers
- Creating the View
- Setting the Start URL and Testing the Application
- Introducing URL Patterns
- Creating and Registering a Simple Route
- Using the Simple Route
- Defining Default Values
- Using Static URL Segments
- Defining Custom Segment Variables
- Using Custom Variables as Action Method Parameters
- Defining Optional URL Segments
- Defining Variable-Length Routes
- Prioritizing Controllers by Namespaces
- Constraining Routes
- Constraining a Route Using a Regular Expression
- Constraining a Route to a Set of Specific Values
- Constraining a Route Using HT T P Methods
- Using T ype and Value Constraints
- Defining a Custom Constraint
- Using Attribute Routing
- Enabling and Applying Attribute Routing
- Creating Routes with Segment Variables
- Applying Route Constraints
- Using a Route Prefix

Advanced Routing Features

- Simplifying the Routes
- Adding the Optimization Package
- Updating the Unit Test Project
- Generating Outgoing URLs in Views
- Using the Routing System to Generate an Outgoing URL
- Targeting Other Controllers
- Passing Extra Values
- Specifying HT ML Attributes
- Generating Fully Qualified URLs in Links
- Generating URLs (and Not Links)

- Generating Outgoing URLs in Action Methods
- Generating a URL from a Specific Route
- Customizing the Routing System
- Creating a Custom RouteBase Implementation
- Creating a Custom Route Handler
- Working with Areas
- Creating an Area
- Populating an Area
- Resolving the Ambiguous Controller Issue
- Creating Areas with Attributes
- Generating Links to Actions in Areas
- Routing Requests for Disk Files
- Configuring the Application Server
- Defining Routes for Disk Files
- Bypassing the Routing System
- URL Schema Best Practices
- Make Your URLs Clean and Human-Friendly
- GET and POST : Pick the Right One

Controllers and Actions

- Introducing the Controller
- Creating a Controller with IController
- Creating a Controller by Deriving from the Controller Class
- Receiving Request Data
- Getting Data from Context Objects
- Using Action Method Parameters
- Producing Output
- Understanding Action Results
- Returning HT ML by Rendering a View
- Passing Data from an Action Method to a View
- Performing Redirections
- Returning Errors and HTTP Codes

Day4:

Filters

- Using Filters
- Introducing the Filter Types
- Applying Filters to Controllers and Action Methods
- Using Authorization Filters
- Applying the Custom Authorization Filter
- Using the Built-in Authorization Filter
- Using Authentication Filters
- Understanding the IAuthenticationFilter Interface
- Implementing the Authentication Check
- Combining Authentication and Authorization Filters
- Handling the Final Challenge Request
- Using Exception Filters
- Creating an Exception Filter
- Applying the Exception Filter
- Using a View to Respond to an Exception
- Using the Built-in Exception Filter
- Using Action Filters
- Implementing the OnActionExecuting Method
- Implementing the OnActionExecuted Method
- Using Result Filters
- Using the Built-in Action and Result Filter Class
- Using Other Filter Features
- Filtering Without Attributes
- Using Global Filters
    - Ordering Filter Execution
    - Overriding Filters

Controller Extensibility

- Creating a Custom Controller Factory
- Dealing with the Fallback Controller
- Instantiating Controller Classes
- Implementing the Other Interface Methods
- Registering a Custom Controller Factory
- Working with the Built-in Controller Factory
- Prioritizing Namespaces
- Customizing DefaultControllerFactory Controller

Instantiation
- Creating a Custom Action Invoker
- Using the Built-in Action Invoker
- Using a Custom Action Name
- Using Action Method Selection
- Improving Performance with Specialized Controllers
- Using Sessionless Controllers
- Using Asynchronous Controllers

Views

- Creating a Custom View Engine
- Creating a Custom IView
- Creating an IViewEngine Implementation
- Registering a Custom View Engine
- Testing the View Engine
- Working with the Razor Engine
- Preparing the Example Project
- Understanding Razor View Rendering
- Configuring the View Search Locations
- Adding Dynamic Content to a Razor View
- Using Layout Sections
- Using Partial Views
- Using Child Actions

Helper Methods

- Creating Custom Helper Methods
- Creating an Inline Helper Method
- Creating an External Helper Method
- Managing String Encoding in a Helper Method
- Using the Built-In Form Helper Methods
- Creating Form Elements
- Specifying the Route Used by a Form
- Using Input Helpers
- Creating Select Elements

Templated Helper Methods

- Using Templated Helper Methods
- Generating Label and Display Elements
- Using Whole-Model Templated Helpers
- Using Model Metadata
- Using Metadata to Control Editing and Visibility
- Using Metadata for Labels
- Using Metadata for Data Values
- Using Metadata to Select a Display Template
- Applying Metadata to a Buddy Class
- Working with Complex Type Properties
- Customizing the Templated View Helper System
- Creating a Custom Editor Template
- Creating a Generic Template
- Replacing the Built-in Templates

Day5:

URL and Ajax Helper Methods

- Creating Basic Links and URLs
- Using MVC Unobtrusive Ajax
- Creating the Synchronous Form View
- Preparing the Project for Unobtrusive Ajax
- Creating an Unobtrusive Ajax Form
- Preparing the Controller
- Creating the Ajax Form
- Understanding How Unobtrusive Ajax Works
- Setting Ajax Options
- Ensuring Graceful Degradation
- Providing the User with Feedback While Making an Ajax Request
- Prompting the User Before Making a Request
- Creating Ajax Links
- Ensuring Graceful Degradation for Links
- Working with Ajax Callbacks
- Working with JSON
- Adding JSON Support to the Controller
- Processing JSON in the Browser

- Preparing Data for Encoding
- Detecting Ajax Requests in the Action Method

Model Binding

- Understanding Model Binding
- Using the Default Model Binder
- Binding to Simple Types
- Binding to Complex Types
- Binding to Arrays and Collections
- Manually Invoking Model Binding
- Dealing with Binding Errors
- Customizing the Model Binding System
- Creating a Custom Value Provider
- Creating a Custom Model Binder
- Registering the Custom Model Binder

Model Validation

- Explicitly Validating a Model
- Displaying Validation Errors to the User
- Displaying Validation Messages
- Displaying Property-Level Validation Messages
- Using Alternative Validation Techniques
- Performing Validation in the Model Binder
- Specifying Validation Rules Using Metadata
- Defining Self-Validating Models
- Performing Client-Side Validation
- Enabling Client-Side Validation
- Using Client-Side Validation
- Understanding How Client-Side Validation Works
- Performing Remote Validation

Bundles

- Profiling Script and Style Sheet Loading
- Using Script and Style Bundles
- Adding the NuGet Package
- Defining the Bundles

- Applying Bundles
- Optimizing the JavaScript and CSS Files

Web API and Single-page Applications

- Understanding Single-page Applications
- Creating the Model
- Adding the NuGet Packages
- Adding the Controller
- Adding the Layout and Views
- Setting the Start Location and Testing the Example Application
- Using Web API
- Creating the Web API Controller
- Testing the API Controller
- Understanding How the API Controller Works
- Understanding API Controller Action Selection
- Mapping HT T P Methods to Action Methods
- Using Knockout for Single-page Applications
- Adding the JavaScript Libraries to the Layout
- Implementing the Summary
- Implementing the Create Feature
- Completing the Application
- Simplify the Home Controller
- Manage Content Visibility

——————————————————————————————-

Duration and pricing

• *Full-time* over 5 days (R9995)

• *Part-time* over 4 weeks (2 nights per week, 3 hour sessions) (R11995)

• *Part-time* over 8 Saturdays, 3 hour sessions (R11995)

• Please note : For *part-time* courses we do not have a fixed schedule and you will be placed on a waiting list until we get a group of 4+ together. Please book with no dates on the bookings form. This will automatically put you on the waiting list. We will confirm with you as soon as we have a part-time group together.

• Distance-learning over up to 3 months (R9995)

• International exams are not included in the course price.

• Prices exclude Vat for Vat-registered companies

Certificate

1. Upon completion of this course we will issue you with attendance certificate to certify your attendance and / or completion of the prescribed minimum examples.
2. You may sit for our competency assessment test and on passing you will obtain our competency certificate.
3. Our competency assessment can be booked and taken by someone who has not attended the course at a cost of R2950 including the course material and guidance session.

Bookings

You can download the course registration form on our home page or by clicking here

Brochure

You may download a pdf copy of this page by clicking on the pdf icon at the top of the page.

Questions

Please email us

Schedule

On the calendar below. If your browser doesn’t display the calendar below, please click on this link or try using Google Chrome, alternatively please enquire via our Contact Us page.

C Programming

This C Programming Course covers the C Programming language comprehensively. C is popular with amateur and commercial programmers alike. Learn to use this powerful and popular language, and get an idea of the kinds of environments in which you will most likely find yourself in when in a C coding job.

Prerequisites

You should not be a complete beginner for this course. If you cannot pass this test, you must do Intro To Programming first.

Course Material

Provided

Course Contents

DAY 1:

Overview and background of the C programming language

- Design Features
- Advantages
- Shortcomings
- What Computers Do
- High-level Computer Languages and Compilers
- Language Standards
- The First ANSI/ISO C Standard
- The C99 Standard
- The C11 Standard
- Commentary
- Programming Mechanics

- Object Code Files, Executable Files, and Libraries
- Unix System
- The GNU Compiler Collection and the LLVM Project
- Linux Systems
- Command-Line Compilers for the PC
- Integrated Development Environments (Windows)
- The Windows/Linux Option
- C on the Macintosh

Introducing C

- A Simple Example of C
- The Example Explained
- Pass 1: Quick Synopsis
- Pass 2: Program Details
- The Structure of a Simple Program
- Tips on Making Your Programs Readable
- Taking Another Step in Using C
- Documentation
- Multiple Declarations
- Multiplication
- Printing Multiple Values
- While You’re at It—Multiple Functions
- Introducing Debugging
- Syntax Errors
- Semantic Errors
- Program State
- Keywords and Reserved Identifiers

Data

- A Sample Program
- What’s New in This Program?
- Data Variables and Constants
- Data: Data-Type Keywords
- Integer Versus Floating-Point Types
- The Integer
- The Floating-Point Number

- Basic C Data Types
- The int Type
- Other Integer Types
- Using Characters: Type char
- The _Bool Type
- Portable Types: stdint.h and inttypes.h
- Types float, double, and long double
- Complex and Imaginary Types
- Beyond the Basic Types
- Type Sizes
- Using Data Types
- Arguments and Pitfalls
- One More Example: Escape Sequences
- What Happens When the Program Runs
- Output

Character Strings and Formatted Input/Output

- Program
- Character Strings: An Introduction
- Type char Arrays and the Null Character
- Using Strings
- The strlen() Function
- Constants and the C Preprocessor
- The const Modifier
- Manifest Constants on the Job
- Exploring and Exploiting printf() and scanf()
- The printf() Function
- Using printf()
- Conversion Specification Modifiers for printf()
- What Does a Conversion Specification Convert?
- Using scanf()
- The * Modifier with printf() and scanf()
- Usage Tips for printf()

DAY 2

Operators, Expressions, and Statements

- Introducing Loops
- Fundamental Operators
- Assignment Operator: =
- Addition Operator: +
- Subtraction Operator: –
- Sign Operators: – and +
- Multiplication Operator: *
- Division Operator: /
- Operator Precedence
- Precedence and the Order of Evaluation
- Some Additional Operators
- The sizeof Operator and the size_t Type
- Modulus Operator: %
- Increment and Decrement Operators: ++ and –
- Decrementing: –
- Precedence
- Expressions and Statements
- Compound Statements (Blocks)
- Type Conversions
- The Cast Operator
- Function with Arguments
- A Sample Program

C Control Statements: Looping

- Revisiting the while Loop
- Program Comments
- C-Style Reading Loop
- The while Statement
- Terminating a while Loop
- When a Loop Terminates
- while: An Entry-Condition Loop
- Syntax Points
- Which Is Bigger: Using Relational Operators and

Expression
- What Is Truth?
- What Else Is True?
- Troubles with Truth
- The New _Bool Type
- Precedence of Relational Operators
- Indefinite Loops and Counting Loops
- The for Loop
- Using for for Flexibility
- More Assignment Operators: +=, -=, *=, /=, %=
- The Comma Operator
- Zeno Meets the for Loop
- An Exit-Condition Loop: do while
- Which Loop?
- Nested Loops
- Program Discussion
- A Nested Variation
- Introducing Arrays
- Using a for Loop with an Array
- A Loop Example Using a Function Return Value
- Program Discussion
- Using Functions with Return Values

C Control Statements: Branching and Jumps

- The if Statement
- Adding else to the if Statement
- Another Example: Introducing getchar() and putchar()
- The ctype.h Family of Character Functions
- Multiple Choice else if
- Pairing else with if
- More Nested ifs
- Let’s Get Logical
- Alternate Spellings: The iso646.h Header File
- Precedence
- Order of Evaluation
- Ranges

- A Word-Count Program
- The Conditional Operator: ?:
- Loop Aids: continue and break
- The continue Statement
- The break Statement
- Multiple Choice: switch and break
- Using the switch Statement
- Reading Only the First Character of a Line
- Multiple Labels
- switch and if else
- The goto Statement
- Avoiding goto

Character Input/Output and Input Validation

- Single-Character I/O: getchar() and putchar()
- Buffers
- Terminating Keyboard Input
- Files, Streams, and Keyboard Input
- The End of File
- Redirection and Files
- Unix, Linux, and Windows Command Prompt Redirection
- Creating a Friendlier User Interface
- Working with Buffered Input
- Mixing Numeric and Character Input
- Input Validation
- Analyzing the Program
- The Input Stream and Numbers
- Menu Browsing
- Tasks
- Toward a Smoother Execution
- Mixing Character and Numeric Input

DAY 3

Functions

- Reviewing Functions

- Creating and Using a Simple Function
- Analyzing the Program
- Function Arguments
- Defining a Function with an Argument: Formal Parameters
- Prototyping a Function with Arguments
- Calling a Function with an Argument: Actual Arguments
- The Black-Box Viewpoint
- Returning a Value from a Function with return
- Function Types
- ANSI C Function Prototyping
- The Problem
- The ANSI C Solution
- No Arguments and Unspecified Arguments
- Hooray for Prototypes
- Recursion
- Recursion Revealed
- Recursion Fundamentals
- Tail Recursion
- Recursion and Reversal
- Recursion Pros and Cons
- Compiling Programs with Two or More Source Code Files
- Unix
- Linux
- DOS Command-Line Compilers
- Windows and Apple IDE Compilers
- Using Header Files
- Finding Addresses: The & Operator
- Altering Variables in the Calling Function
- Pointers: A First Look
- The Indirection Operator: *
- Declaring Pointers
- Using Pointers to Communicate Between Functions

Arrays and Pointers

- Arrays
- Initialization

- Designated Initializers (C99)
- Assigning Array Values
- Array Bounds
- Specifying an Array Size
- Multidimensional Arrays
- Initializing a Two-Dimensional Array
- More Dimensions
- Pointers and Arrays
- Functions, Arrays, and Pointers
- Using Pointer Parameters
- Comment: Pointers and Arrays
- Pointer Operations
- Protecting Array Contents
- Using const with Formal Parameters
- More About const
- Pointers and Multidimensional Arrays
- Pointers to Multidimensional Arrays
- Pointer Compatibility
- Functions and Multidimensional Arrays
- Variable-Length Arrays (VLAs)
- Compound Literals

Character Strings and String Functions

- Representing Strings and String I/O
- Defining Strings Within a Program
- Pointers and Strings
- String Input
- Creating Space
- The Unfortunate gets() Function
- The Alternatives to gets()
- The scanf() Function
- String Output
- The puts() Function
- The fputs() Function
- The printf() Function
- The Do-It-Yourself Option

- String Functions
- The strlen() Function
- The strcat() Function
- The strncat() Function
- The strcmp() Function
- The strcpy() and strncpy() Functions
- The sprintf() Function
- Other String Functions
- A String Example: Sorting Strings
- Sorting Pointers Instead of Strings
- The Selection Sort Algorithm
- The ctype.h Character Functions and Strings
- Command-Line Arguments
- Command-Line Arguments in Integrated Environments
- Command-Line Arguments with the Macintosh
- String-to-Number Conversions

DAY 4

Storage Classes, Linkage, and Memory Management

- Storage Classes
- Scope
- Linkage
- Storage Duration
- Automatic Variables
- Register Variables
- Static Variables with Block Scope
- Static Variables with External Linkage
- Static Variables with Internal Linkage
- Multiple Files
- Storage-Class Specifier Roundup
- Storage Classes and Functions
- Which Storage Class?
- A Random-Number Function and a Static Variable
- Roll ’Em
- Allocated Memory: malloc() and free()

- The Importance of free()
- The calloc() Function
- Dynamic Memory Allocation and Variable-Length Arrays
- Storage Classes and Dynamic Memory Allocation
- ANSI C Type Qualifiers
- The const Type Qualifier
- The volatile Type Qualifier
- The restrict Type Qualifier
- The _Atomic Type Qualifier (C11)
- New Places for Old Keywords

File Input/Output

- Communicating with Files
- What Is a File?
- The Text Mode and the Binary Mode
- Levels of I/O
- Standard Files
- Standard I/O
- Checking for Command-Line Arguments
- The fopen() Function
- The getc() and putc() Functions
- End-of-File
- The fclose() Function
- Pointers to the Standard Files
- A Simple-Minded File-Condensing Program
- File I/O: fprintf(), fscanf(), fgets(), and fputs()
- The fprintf() and fscanf() Functions
- The fgets() and fputs() Functions
- Adventures in Random Access: fseek() and ftell()
- How fseek() and ftell() Work
- Binary Versus Text Mode
- Portability
- The fgetpos() and fsetpos() Functions
- Behind the Scenes with Standard I/O
- Other Standard I/O Functions
- The int ungetc(int c, FILE *fp) Function

- The int fflush() Function
- The int setvbuf() Function
- Binary I/O: fread() and fwrite()
- The size_t fwrite() Function
- The size_t fread() Function
- The int feof(FILE *fp) and int ferror(FILE *fp) Functions
- An fread() and fwrite() Example
- Random Access with Binary I/O

Structures and Other Data Forms

- Sample Problem: Creating an Inventory of Books
- Setting Up the Structure Declaration
- Defining a Structure Variable
- Initializing a Structure
- Gaining Access to Structure Members
- Initializers for Structures
- Arrays of Structures
- Declaring an Array of Structures
- Identifying Members of an Array of Structures
- Program Discussion
- Nested Structures
- Pointers to Structures
- Declaring and Initializing a Structure Pointer
- Member Access by Pointer
- Telling Functions About Structures
- Passing Structure Members
- Using the Structure Address
- Passing a Structure as an Argument
- More on Structure Features
- Structures or Pointer to Structures?
- Character Arrays or Character Pointers in a Structure 627
- Structure, Pointers, and malloc()
- Compound Literals and Structures (C99)
- Flexible Array Members (C99)

- Anonymous Structures (C11)
- Functions Using an Array of Structures
- Saving the Structure Contents in a File
- A Structure-Saving Example
- Program Points
- Structures: What Next?
- Unions: A Quick Look
- Using Unions
- Anonymous Unions (C11)
- Enumerated Types
- enum Constants
- Default Values
- Assigned Values
- enum Usage
- Shared Namespaces
- typedef: A Quick Look
- Fancy Declarations
- Functions and Pointers

DAY 5

Bit Fiddling

- Binary Numbers, Bits, and Bytes
- Binary Integers
- Signed Integers
- Binary Floating Point
- Other Number Bases
- Octal
- Hexadecimal
- C’s Bitwise Operators
- Bitwise Logical Operators
- Usage: Masks
- Usage: Turning Bits On (Setting Bits)
- Usage: Turning Bits Off (Clearing Bits)
- Usage: Toggling Bits
- Usage: Checking the Value of a Bit

- Bitwise Shift Operators
- Programming Example
- Another Example
- Bit Fields
- Bit-Field Example
- Bit Fields and Bitwise Operators
- Alignment Features (C11)

The C Preprocessor and the C Library

- First Steps in Translating a Program
- Manifest Constants: #define
- Tokens
- Redefining Constants
- Using Arguments with #define
- Creating Strings from Macro Arguments: The # Operator
- Preprocessor Glue: The ## Operator
- Variadic Macros: … and __VA_ARGS__
- Macro or Function?
- File Inclusion: #include
- Header Files: An Example
- Uses for Header Files
- Other Directives
- The #undef Directive
- Being Defined—The C Preprocessor Perspective
- Conditional Compilation
- Predefined Macros
- #line and #error
- #pragma
- Generic Selection (C11)
- Inline Functions (C99)
- _Noreturn Functions (C11)
- The C Library
- Gaining Access to the C Library
- Using the Library Descriptions
- The Math Library
- A Little Trigonometry

- Type Variants
- The tgmath.h Library (C99)
- The General Utilities Library
- The exit() and atexit() Functions
- The qsort() Function
- The Assert Library
- Using assert
- _Static_assert (C11)
- memcpy() and memmove() from the string.h Library
- Variable Arguments: stdarg.h

Advanced Data Representation

- Exploring Data Representation
- Beyond the Array to the Linked List
- Using a Linked List
- Afterthoughts
- Abstract Data Types (ADTs)
- Getting Abstract
- Building an Interface
- Using the Interface
- Implementing the Interface
- Getting Queued with an ADT
- Defining the Queue Abstract Data Type
- Defining an Interface
- Implementing the Interface Data Representation
- Testing the Queue
- Simulating with a Queue
- The Linked List Versus the Array
- Binary Search Trees
- A Binary Tree ADT
- The Binary Search Tree Interface
- The Binary Tree Implementation
- Trying the Tree
- Tree Thoughts

Duration and pricing

In Price Group A

Certificate

Read about our certificates here

Bookings

You can download the course registration form on our home page or by clicking here

Brochure

You may download a pdf copy of this page by clicking on the pdf icon at the top of the page.

Questions

Please email us

Schedule

On the calendar below. If your browser doesn't display the calendar below, please click on this link or try using Google Chrome, alternatively please enquire via our Contact Us page.

C# 6 and .NET Core 1.0

C# 6 and .NET Core 1.0

Prerequisites / Further Training

- Introduction to Programming
- Beginner C#.NET (C# Fundamentals and Object – Orientation)

- *Advanced C#.NET (Building practical Apps)*
- SQL Querying (Database design and querying)
- ASP.NET CORE MVC (Web App Development)
- ASP.NET Web API (More Web App Development)
- Entity Framework

Also have a look at our .NET Bootcamp and save

Intended Audience

This course provides a comprehensive foundation in the C# programming language and the core aspects of the .NET platform plus overviews of technologies built on top of C# and .NET (ADO.NET and Entity Framework, Windows Communication Foundation [WCF], Windows Presentation Foundation [WPF], and ASP.NET [WebForms, MVC, WebAPI]). Once you digest the information presented in course, you'll be in a perfect position to apply this knowledge to your specific programming assignments, and you'll be well equipped to explore the .NET universe further.

Day 1 topics are covered in review mode as these are covered in detail on the C#.NET Beginner Course – it is required and therefore assumed that you are up to speed on at least 66.7% of day 1 topics.

After this course you should be able to

- Understand the .NET Core platform and C# 6.
- Discover the ins and outs of the leading .NET technology.
- Find complete coverage of XAML, .NET Core and Visual Studio 2015 together with an overview of the new Windows Runtime
- Students will gain a solid foundation of object-oriented development techniques, attributes and reflection, generics and collections as well as advanced topics such as CIL opcodes and emitting dynamic assemblies.

Alignment

On the same level as Microsoft : 20483B . View the MCPD Course Schedules

Further Training

OO Analysis and Design
Design Patterns
ASP.NET Forms

Course Material

Up to date course material is provided

Course info

DAY 1:

.NET CORE AND WINDOWS RUNTIME

- .NET Compiler Platform
- Testing
- Diagnostics and Application Insights
- Tasks and Parallel Programming
- Task Synchronization

DAY 2:

- Files and Streams
- Continues
- Security
- Networking
- Composition
- XML and JSON
- Localization

DAY 3 : WINDOWS APPS

- Core XAML
- Styling XAML Apps

- Patterns with XAML Apps
- Windows Apps: User Interfaces
- Advanced Windows Apps
- Windows Desktop Applications with WPF
- Creating Documents with WPF
- Deploying Windows Apps

DAY 4:

- WEP APPS AND SERVICES
- ADO.NET
- Entity Framework Core
- Windows Services
- ASP.NET Core

DAY 5:

- ASP.NET MVC
- ASP.NET Web API
- WebHooks and SignalR
- Windows Communication Foundation
- Deploying Websites and Services

Duration and pricing

- Full-time over 5 days (R9995)
- Part-time over 4 weeks (2 nights per week, 3 hour sessions) (R11995)
- Part-time over 8 Saturdays, 3 hour sessions (R11995)
- Please note : For part-time courses we do not have a fixed schedule and you will be placed on a waiting list until we get a group of 4+ together. Please book with no dates on the bookings form. This will automatically put you on the waiting list. We will confirm with you as soon as we have a part-time group together.
- “Distance Learning” over up to 3 months

(R9995)
- International exams are not included in the course price.
- Prices exclude Vat for Vat-registered companies

Certificate

- Upon completion of this course we will issue you with attendance certificate to certify your attendance and / or completion of the prescribed minimum examples.
- You may sit for our competency assessment test and on passing you will obtain our competency certificate.
- Our competency assessment can be booked and taken by someone who has not attended the course at a cost of R2950.

Bookings

You can download the course registration form on our home page or by clicking here

Brochure

You may download a pdf copy of this page by clicking on the pdf icon at the top of the page.

Questions

Please email us

Schedule

On the calendar below. If your browser doesn't display the calendar below, please click on this link or try using Google Chrome, alternatively please enquire via our Contact Us page.

C#.NET Advanced

C#.NET 6 on .NET 4.6

Prerequisites / Further Training

- Introduction to Programming
- Beginner C#.NET (C# Fundamentals and Object – Orientation)
- *Advanced C#.NET (Building practical Apps)*
- SQL Querying (Database design and querying)
- ASP.NET MVC (Web App Development)
- ASP.NET Web API (More Web App Development)
- Entity Framework

Also have a look at our .NET Bootcamp and save

Intended Audience

This course provides a comprehensive foundation in the C# programming language and the core aspects of the .NET platform plus overviews of technologies built on top of C# and .NET (ADO.NET and Entity Framework, Windows Communication Foundation [WCF], Windows Presentation

Foundation [WPF], and ASP.NET [WebForms, MVC, WebAPI]). Once you digest the information presented in course, you'll be in a perfect position to apply this knowledge to your specific programming assignments, and you'll be well equipped to explore the .NET universe further.

Day 1 topics are covered in review mode as these are covered in detail on the C#.NET Beginner Course – it is required and therefore assumed that you are up to speed on at least 66.7% of day 1 topics.

After this course you should be able to

- Understand the .NET 4.6 platform and C# 6.
- Discover the ins and outs of the leading .NET technology.
- Find complete coverage of XAML, .NET 4.6 and Visual Studio 2015 together with an overview of the new Windows Runtime
- Students will gain a solid foundation of object-oriented development techniques, attributes and reflection, generics and collections as well as advanced topics such as CIL opcodes and emitting dynamic assemblies.

Alignment

On the same level as Microsoft : 20483B . View the MCPD Course Schedules

Further Training

OO Analysis and Design
Design Patterns
ASP.NET Forms

Course Material

Up to date course material is provided

Course info

Day 1	

Day 1 topics are covered in review mode as these are covered in detail on the C#.NET Beginner Course – it is required and assumed that you are up to speed on at least 66.7% of day 1 topics.

Overview of C#.NET Beginner

- An Initial Look at the .NET Platform
- Introducing the Building Blocks of the .NET Platform (the CLR, CTS, and CLS)
- Additional .NET-Aware Programming Languages
- An Overview of .NET Assemblies
- Understanding the Common Type System
- Understanding the Common Language Specification
- Understanding the Common Language Runtime
- The Assembly/Namespace/Type Distinction
- Exploring an Assembly Using ildasm.exe
- The Platform-Independent Nature of .NET

- Building C# Applications on the Windows OS
- Building .NET Applications Beyond the Windows OS

- The Anatomy of a Simple C# Program
- An Interesting Aside: Some Additional Members of the System.Environment Class
- The System.Console Class
- System Data Types and Corresponding C# Keywords
- Working with String Data
- Narrowing and Widening Data Type Conversions
- Understanding Implicitly Typed Local Variables
- C# Iteration Constructs
- Decision Constructs and the Relational/Equality Operators

- Methods and Parameter Modifiers
- Understanding C# Arrays
- Understanding the enum Type
- Understanding the Structure (aka Value Type)

- Understanding Value Types and Reference Types
- Understanding C# Nullable Types

Encapsulation

- Introducing the C# Class Type
- Understanding Constructors
- The Role of the this Keyword
- Understanding the static Keyword
- Defining the Pillars of OOP
- C# Access Modifiers
- The First Pillar: C#’s Encapsulation Services
- Understanding Automatic Properties
- Understanding Object Initialization Syntax
- Working with Constant Field Data
- Understanding Partial Classes

Inheritance / Polymorphism

- The Basic Mechanics of Inheritanc
- Revising Visual Studio Class Diagrams
- The Second Pillar of OOP: The Details of Inheritance
- Programming for Containment/Delegation
- The Third Pillar of OOP: C#’s Polymorphic Support
- Understanding Base Class/Derived Class Casting Rules
- The Master Parent Class: System.Object

Exceptions

- Ode to Errors, Bugs, and Exceptions
- The Role of .NET Exception Handling
- The Simplest Possible Example
- Configuring the State of an Exception
- System-Level Exceptions (System.SystemException)
- Application-Level Exceptions (System.ApplicationException)
- Processing Multiple Exceptions
- Debugging Unhandled Exceptions Using Visual Studio

Interfaces

- Understanding Interface Types
- Defining Custom Interfaces
- Implementing an Interface
- Invoking Interface Members at the Object Level
- Interfaces As Parameters
- Interfaces As Return Values
- Arrays of Interface Types
- Implementing Interfaces Using Visual Studio
- Explicit Interface Implementation
- Designing Interface Hierarchies
- The IEnumerable and IEnumerator Interfaces
- The ICloneable Interface
- The IComparable Interface

Day 2

The core C# Advanced Course starts here

Collections and Generic

- The Motivation for Collection Classes
- The Problems of Nongeneric Collections
- The Role of Generic Type Parameters
- The System.Collections.Generic Namespace
- The System.Collections.ObjectModel Namespace
- Creating Custom Generic Methods
- Creating Custom Generic Structures and Classes
- Constraining Type Parameters

Delegates / events / lambdas

- Understanding the .NET Delegate Type
- The Simplest Possible Delegate Example

Sending Object State Notifications Using Delegates

- Understanding Generic Delegates
- Understanding C# Events

- Understanding C# Anonymous Methods
- Understanding Lambda Expressions

Advanced language features

- Understanding Indexer Methods
- Understanding Operator Overloading
- Understanding Custom Type Conversions
- Understanding Extension Methods
- Understanding Anonymous Types
- Working with Pointer Types

LINQ to Objects

- LINQ-Specific Programming Constructs
- Understanding the Role of LINQ
- Applying LINQ Queries to Primitive Arrays
- Returning the Result of a LINQ Query
- Applying LINQ Queries to Collection Objects
- Investigating the C# LINQ Query Operators
- The Internal Representation of LINQ Query Statements

Day 3

Object Lifetime

- Classes, Objects, and References
- The Basics of Object Lifetime
- The Role of Application Roots
- Understanding Object Generations
- Concurrent Garbage Collection Prior to .NET 4.0
- Background Garbage Collection Under .NET 4.0 and Beyond
- The System.GC Type
- Building Finalizable Objects
- Building Disposable Objects
- Building Finalizable and Disposable Types
- Understanding Lazy Object Instantiation

Assembly configuration

- Defining Custom Namespaces
- The Role of .NET Assemblies
- Understanding the Format of a .NET Assembly
- Building and Consuming Custom Class Library
- Understanding Private Assemblies
- Understanding Shared Assemblies
- Consuming a Shared Assembly
- Configuring Shared Assemblies
- Understanding Publisher Policy Assemblies
- Understanding the <codeBase> Element
- The System.Configuration Namespace
- The Configuration File Schema Documentation

Reflection and attributes

- The Necessity of Type Metadata
- Understanding Reflection
- Building a Custom Metadata Viewer
- Dynamically Loading Assemblies
- Reflecting on Shared Assemblies
- Understanding Late Binding
- Understanding the Role of .NET Attributes
- Building Custom Attributes
- Assembly-Level Attributes
- Reflecting on Attributes Using Early Binding
- Reflecting on Attributes Using Late Binding
- Putting Reflection, Late Binding, and Custom Attributes in Perspective
- Building an Extendable Application

Dynamic types / DLR

- The Role of the C# dynamic Keyword
- The Role of the Dynamic Language Runtime
- Simplifying Late-Bound Calls Using Dynamic Types
- Simplifying COM Interoperability Using Dynamic Data
- COM Interop Using C# Dynamic Data

Processes, appdomains

- The Role of a Windows Process
- Interacting with Processes Under the .NET Platform
- Understanding .NET Application Domains
- Interacting with the Default Application Domain
- Creating New Application Domains
- Understanding Object Context Boundaries
- Summarizing Processes, AppDomains, and Context

CIL

- Motivations for Learning the Grammar of CIL
- Examining CIL Directives, Attributes, and Opcodes
- Pushing and Popping: The Stack-Based Nature of CIL
- Understanding Round-Trip Engineering
- Understanding CIL Directives and Attributes
- .NET Base Class Library, C#, and CIL Data Type Mappings
- Defining Type Members in CIL
- Examining CIL Opcodes
- Building a .NET Assembly with CIL
- Understanding Dynamic Assemblies

Day 4

Threading

- The Process/AppDomain/Context/Thread Relationship
- A Brief Review of the .NET Delegate
- The Asynchronous Nature of Delegates
- Invoking a Method Asynchronously
- The System.Threading Namespace
- The System.Threading.Thread Class
- Manually Creating Secondary Threads
- The Issue of Concurrency
- Programming with Timer Callbacks
- Understanding the CLR ThreadPool
- Parallel Programming Using the Task Parallel LibraryParallel LINQ Queries (PLINQ)

- Asynchronous Calls with the async Keyword

File IO / Object Serialization

- Exploring the System.IO Namespace
- The Directory(Info) and File(Info) Types
- Working with the DirectoryInfo Type
- Working with the Directory Type
- Working with the DriveInfo Class Type
- Working with the FileInfo Class
- Working with the File Type
- The Abstract Stream Class
- Working with StreamWriters and StreamReaders
- Working with StringWriters and StringReaders
- Working with BinaryWriters and BinaryReaders
- Watching Files Programmatically
- Understanding Object Serialization
- Configuring Objects for Serialization
- Choosing a Serialization Formatter
- Serializing Objects Using the BinaryFormatter
- Serializing Objects Using the SoapFormatter
- Serializing Objects Using the XmlSerializer
- Serializing Collections of Objects
- Customizing the Soap/Binary Serialization Process

ADO.NET Part 1

- A High-Level Definition of ADO.NET
- Understanding ADO.NET Data Providers
- Additional ADO.NET Namespaces
- The Types of the System.Data Namespace
- Abstracting Data Providers Using Interfaces
- Creating the AutoLot Database
- The ADO.NET Data Provider Factory Model
- Understanding the Connected Layer of ADO.NET
- Working with Data Readers
- Building a Reusable Data Access Library
- Creating a Console UI–Based Front End

- Understanding Database Transactions

ADO.NET Part 2

- Understanding the Disconnected Layer of ADO.NET
- Understanding the Role of the DataSet
- Working with DataColumns
- Working with DataRows
- Working with DataTables
- Binding DataTable Objects to Windows Forms GUIs
- Working with Data Adapters
- Adding Disconnected Functionality to AutoLotDAL.dll
- Multitabled DataSet Objects and Data Relationships
- The Windows Forms Database Designer Tools
- Isolating Strongly Typed Database Code into a Class Library
- Programming with LINQ to DataSet

Entity Framework

- Understanding the Role of the Entity Framework
- Code First from an Existing Database
- Using the Model Classes in CodeHandling Database Changes
- AutoLotDAL Version 4
- Test-Driving AutoLotDAL
- Entity Framework Migrations
- Revisiting the Transaction Test
- Concurrency
- Interception
- ObjectMaterialized and SavingChanges Events
- Deploying to SQL Server

LINQ to XML

- A Tale of Two XML APIs
- Members of the System.Xml.Linq Namespace
- Working with XElement and XDocument
- Manipulating an In-Memory XML Document

Day 5

Introducing Windows Communication Foundation

- A Potpourri of Distributed Computing APIs
- The Role of WCF
- Investigating the Core WCF Assemblies
- The Visual Studio WCF Project Templates
- The Basic Composition of a WCF Application
- The ABCs of WCF
- Building a WCF Service
- Hosting the WCF Service
- Building the WCF Client Application
- Simplifying Configuration Settings
- Using the WCF Service Library Project Template
- Hosting the WCF Service Within a Windows Service
- Invoking a Service Asynchronously from the Client
- Designing WCF Data Contracts

Windows Presentation Foundation

- The Motivation Behind WPF
- The Various Flavors of WPF
- Investigating the WPF Assemblies
- Building a WPF Application Without XAML
- Building a WPF Application Using Only XAML
- Transforming Markup into a .NET Assembly
- Understanding the Syntax of WPF XAML
- Building a WPF Application Using Code-Behind Files
- Building WPF Applications Using Visual Studio
- Building a Custom XAML Editor with Visual Studio

Programming with WPF Controls

- A Survey of the Core WPF Controls
- A Brief Review of the Visual Studio WPF Designer
- Controlling Content Layout Using Panels
- Building a Window’s Frame Using Nested Panels
- Understanding WPF Commands

- Understanding Routed Events
- A Deeper Look at WPF APIs and Controls
- Building the Ink API Tab
- Introducing the Documents API
- Building the Documents Tab
- Introducing the WPF Data-Binding Model
- Understanding the Role of Dependency Properties
- Building a Custom Dependency Property

WPF Graphics Rendering Services

- Understanding WPF’s Graphical Rendering Services
- Rendering Graphical Data Using Shapes
- WPF Brushes and Pens
- Applying Graphical Transformations
- Working with the Visual Studio Transform Editor
- Rendering Graphical Data Using Drawings and Geometries
- Working with Vector Images
- Rendering Graphical Data Using the Visual Layer

WPF Resources, Animations, Styles, and Templates

- Understanding the WPF Resource System
- Working with Object (Logical) Resources
- Understanding WPF’s Animation Services
- Authoring Animations in XAML
- Understanding the Role of WPF Styles
- Logical Trees, Visual Trees, and Default Templates
- Building a Control Template with the Trigger Framework

Notifications, Commands, Validation, and MVVM

- Introducing Model-View-ViewModel
- The WPF Binding Notification System
- Validation
- Using Data Annotations
- Creating Custom Commands
- Fully Implementing MVVM
- Updating AutoLotDAL for MVVM

- Full MVVM Example

- **Duration and pricing**

- *Full-time* over 5 days (R9995)
- *Part-time* over 4 weeks (2 nights per week, 3 hour sessions) (R11995)
- *Part-time* over 8 Saturdays, 3 hour sessions (R11995)
- Please note : For *part-time* courses we do not have a fixed schedule and you will be placed on a waiting list until we get a group of 4+ together. Please book with no dates on the bookings form. This will automatically put you on the waiting list. We will confirm with you as soon as we have a part-time group together.
- Distance-learning over up to 3 months (R9995)
- International exams are not included in the course price.
- Prices exclude Vat for Vat-registered companies

Certificate

1. Upon completion of this course we will issue you with attendance certificate to certify your attendance and / or completion of the prescribed minimum examples.
2. You may sit for our competency assessment test and on passing you will obtain our competency certificate.
3. Our competency assessment can be booked and taken by someone who has not attended the course at a cost of R2950.

Bookings

You can download the course registration form on our home page or by clicking here

Brochure

You may download a pdf copy of this page by clicking on the pdf icon at the top of the page.

Questions

Please email us

Schedule

On the calendar below. If your browser doesn’t display the calendar below, please click on this link or try using Google Chrome, alternatively please enquire via our Contact Us page.

C#.NET Beginner

Prerequisites / Further Training

You should not be a complete beginner for this course. If you cannot pass this test, you must do Intro to Programming with Java Script or Intro to Programming with Python first.

- *Beginner C#.NET (C# Fundamentals and Object – Orientation)*
- Advanced C#.NET (Building practical Apps)
- SQL Querying (Database design and querying)
- ASP.NET MVC (Web App Development)
- ASP.NET Web API (More Web App Development)
- Entity Framework

Also have a look at our .NET Bootcamp and save

Intended Audience

This course is for anyone at the Intro to Programming level who want to learn how to program in C# using the .NET framework

After this course you should be able to

Create a fun arcade game and build games and other projects

Further Training

Recommended – C# Advanced

Alignment

Together, the Beginner and Advanced C#.NET are on the same level as Microsoft : 10266A .

Course Material

Up to date course material is provided

Course info
Start Building With c#: Build something cool, fast!

- Why you should learn C#
- C# and the Visual Studio IDE
- Debugging

It's all Just Code

- Anatomy of a program
- Two classes can be in the same namespace
- Variables
- C# uses familiar math symbols
- Use the debugger to see your variables change
- Loops
- if/else statements make decisions
- Build an app from the ground up

Day 2

Objects: Get Oriented!

- Classes
- Methods

- Objects
- An example solution
- Properties
- Instances
- Naming
- Class diagrams
- There’s an easier way to initialize objects
- A few ideas for designing intuitive classes

Types and References

- The variable’s type determines what kind of data it can store
- Casting
- C# does some casting automatically
- When you call a method, the arguments must be compatible with the types of the parameters
- References are like labels for your object
- If there aren’t any more references, your object gets garbage-collected
- Multiple references and their side effects
- Arrays can contain a bunch of reference variables, too
- Objects use references to talk to each other
- Controls are objects, just like any other object

Day 3

Encapsulation

- Building an Event – planning app
- Each option should be calculated individually
- It’s easy to accidentally misuse your objects
- Encapsulation means keeping some of the data in a class private
- Use encapsulation to control access to your class’s methods and fields
- Private fields and methods can only be accessed from inside the class

- A few ideas for encapsulating classes
- Encapsulation keeps your data pristine
- Properties make encapsulation easier
- Automatic properties
- Use a constructor to initialize private fields

Inheritance

- Anoter event-planning typ of class
- When your classes use inheritance, you only need to write your code once
- Build up your class model by starting general and getting more specific
- Let’s design a zoo simulator
- Use inheritance to avoid duplicate code in subclasses
- Create the class hierarchy
- Every subclass extends its base class
- Use a colon to inherit from a base class
- We know that inheritance adds the base class fields, properties, and methods to the subclass…
- Method overriding
- Use the override and virtual keywords to inherit behavior
- A subclass can access its base class using the base keyword
- When a base class has a constructor, your subclass needs one, too

Day 4

Interfaces and Abstract Classes

- An interface tells a class that it must implement certain methods and properties
- Use the interface keyword to define an interface
- Upcasting works with both objects and interfaces
- Downcasting lets you turn your appliance back into a coffee maker

- Upcasting and downcasting work with interfaces, too
- There’s more than just public and private
- Access modifiers change visibility
- Some classes should never be instantiated
- An abstract class is like a cross between a class and an interface
- An abstract method doesn’t have a body
- Polymorphism means that one object can take many different forms

Enums and Collections

- Enums let you work with a set of valid values
- Enums let you represent numbers with names
- Lists make it easy to store collections of…anything
- Lists are more flexible than arrays
- Lists shrink and grow dynamically
- Generics can store any type
- Collection initializers are similar to object initializers
- Lists are easy, but SORTING can be tricky
- IComparable helps your list sort
- Use IComparer tells your List how to sort
- IComparer can do complex comparisons
- Overriding a ToString() method lets an object describe itself
- You can upcast an entire list using IEnumerable
- You can build your own overloaded methods
- Use a dictionary to store keys and values
- A queue is FIFO—First In, First Out
- A stack is LIFO—Last In, First Out

Day 5

Reading and Writing Files

- Use Stream.Read() to read bytes from a stream
- StreamWriter

- StreamReader
- You can read and write serialized files manually, too
- Use a BinaryWriter to write binary data
- C# can use byte arrays to move data around
- NET uses Unicode to store characters and text
- Serialization
- Use a switch statement to choose the right option
- Writing files usually involves making a lot of decisions
- Avoid filesystem errors with using statements
- IDisposable makes sure your objects are disposed of properly
- Use file dialogs to open and save files (all with just a few lines of code)
- Use the built-in File and Directory classes to work with files and directories
- Dialog boxes are objects, too
- Dialog boxes are just another WinForms control
- Reading and writing using two objects
- .NET streams to read and write data

Duration and pricing

- *Full-time* over 5 days (R9995)
- *Part-time* over 4 weeks (2 nights per week, 3 hour sessions) (R11995)
- *Part-time* over 8 Saturdays, 3 hour sessions (R11995)
- Please note : For *part-time* courses we do not have a fixed schedule and you will be placed on a waiting list until we get a group of 4+ together. Please book with no dates on the bookings form. This will automatically put you on the waiting list. We will confirm with you as soon as we have a part-time group together.
- Distance-learning over up to 3 months (R9995)
- International exams are not included in the course price.
- Prices exclude Vat for Vat-registered companies
- **Monthly payment options for Distance Learning / Self**

Study

If you want to pay the course on a monthly basis, we divide the course in 4 sections – one per month. You then have to complete a quarter of the course per month. The payments are as follows:

- R3333 registration fee and covering the first month and section 1
- R3333- month 2 and section 2
- R3333 – month 3 the last section
- **Certificate**
    1. Upon completion of this course we will issue you with attendance certificate to certify your attendance and / or completion of the prescribed minimum examples.
    2. You may sit for our competency assessment test and on passing you will obtain our competency certificate.
    3. Our competency assessment can be booked and taken by someone who has not attended the course at a cost of R2950, including the course material and a guidance session.

Bookings

You can download the course registration form on our home page or by clicking here

Brochure

You may download a pdf copy of this page by clicking on the pdf icon at the top of the page.

Questions

Please email us

Schedule

On the calendar below. If your browser doesn't display the calendar below, please click on this link or try using Google Chrome, alternatively please enquire via our Contact Us page.

C++ Beginner

C++ Beginner

Learn one of the most popular and widely-used languages in the world. Learn Object-Orientation and if you are interested in the international exam, our syllabus covers it.

Prerequisites

You should not be a complete beginner for this course. If you cannot pass this test, you must do Intro To Programming first.

Alignment

C++ Certified Associate Programmer (CAP) from the C++ Institute

Further Training

C++ Advanced

Course Material

Included

Course Contents

DAY 1:

Setting Out to C++

- C++ Initiation
- C++ Statements
- More C++ Statements
- Functions
- Summary
- Chapter Review
- Programming Exercises

Dealing with Data

- Simple Variables
- The const Qualifier
- Floating-Point Numbers
- C++ Arithmetic Operators

Compound Types

- Introducing Arrays
- Strings
- Introducing the string Class
- Introducing Structures
- Unions
- Enumerations
- Pointers and the Free Store
- Pointers, Arrays, and Pointer Arithmetic
- Combinations of Types
- Array Alternatives

Loops and Relational Expressions

- Introducing for Loops
- The while Loop
- The do while Loop
- The Range-Based for Loop (C++11)
- Loops and Text Input
- Nested Loops and Two-Dimensional Arrays
- Summary
- Chapter Review
- Programming Exercises

DAY 2:

Branching Statements and Logical Operators

- The if Statement
- Logical Expressions
- The cctype Library of Character Functions
- The ?: Operator
- The switch Statement
- The break and continue Statements
- Number-Reading Loops
- Simple File Input/Output

Functions: C++’s Programming Modules

- Function Review
- Function Arguments and Passing by Value
- Functions and Arrays
- Functions and Two-Dimensional Arrays
- Functions and C-Style Strings
- Functions and Structures
- Functions and string Class Objects
- Functions and array Objects
- Recursion
- Pointers to Functions

Adventures in Functions

- C++ Inline Functions
- Reference Variables
- Default Arguments
- Function Overloading
- Function Templates

Memory Models and Namespaces

- Separate Compilation
- Storage Duration, Scope, and Linkage
- Namespaces

DAY 3:

Objects and Classes

- Procedural and Object-Oriented Programming
- Abstraction and Classes
- Class Constructors and Destructors
- Knowing Your Objects: The this Pointer
- An Array of Object
- Class Scope
- Abstract Data Types

Working with Classes

- Operator Overloading
- Time on Our Hands: Developing an Operator
- Overloading Example
- Introducing Friends
- Overloaded Operators: Member Versus Nonmember
- Functions
- More Overloading: A Vector Class
- Automatic Conversions and Type Casts for Classes

Classes and Dynamic Memory Allocation

- Dynamic Memory and Classes
- The New, Improved String Class
- Things to Remember When Using new
- in Constructors
- Observations About Returning Objects
- Using Pointers to Objects
- Reviewing Techniques
- A Queue Simulation

Class Inheritance

- Beginning with a Simple Base Class
- Inheritance: An Is-a Relationship
- Polymorphic Public Inheritance

- Static and Dynamic Binding
- Access Control: protected
- Abstract Base Classes
- Inheritance and Dynamic Memory Allocation
- Class Design Review

DAY 4:

Reusing Code in C++

- Classes with Object Members
- Private Inheritance
- Multiple Inheritance
- Class Templates

Friends, Exceptions, and More

- Friends
- Nested Classes
- Exceptions
- Runtime Type Identification
- Type Cast Operators

The string Class and the Standard

- Template Library
- The string Class
- Smart Pointer Template Classes
- The Standard Template Library
- Generic Programming
- Function Objects (a.k.a. Functors)
- Algorithms
- Other Libraries

Input, Output, and Files

- An Overview of C++ Input and Output
- Output with cout
- Input with cin

- File Input and Output
- Incore Formatting

DAY 5:

Visiting with the New C++ Standard

- C++11 Features Revisited
- Move Semantics and the Rvalue Reference
- New Class Features
- Lambda Functions
- Wrappers
- Variadic Templates
- More C++11 Features
- Language Change

PROJECT

—

Duration and pricing

In Price Group A

Certificate

Read about our certificates here

Bookings

You can download the course registration form on our home page or by clicking here

Brochure

You may download a pdf copy of this page by clicking on the pdf icon at the top of the page.

Questions

Please email us

Schedule

On the calendar below. If your browser doesn't display the calendar below, please click on this link or try using Google Chrome, alternatively please enquire via our Contact Us page.

Design Patterns

Django

Intended Audience

This Django Training Course is for Python programmers who want to learn the best and most-widely used framework for developing Python-based Web applications.

Prerequisites

Advanced Python

After this course you should be able to

Quickly start a new Django project and establish a strong foundation for a growing site

- Define how your data is organized and create a SQLite database to manage it
- Quickly produce HTML with Django templates
- Create dynamic webpages with Django's URL patterns and views, including function views, class-based views, and generic views

- Enable efficient, reliable data input with Django Forms and custom form validations
- Understand the Model-View-Controller (MVC) architecture, compare it to Model-Template-Views, and gain a holistic understanding of Django’s structure
- Write as little code as possible, simplify code reuse, and mitigate software decay by adhering to the Don’t Repeat Yourself paradigm.
- Dive into Django source code to troubleshoot problems
- Extend site functionality with Django’s contributed library
- Protect your site with user authentication and permissions
- Avoid security pitfalls such as SQL Injection, XSS, and CSRF
- Optimize site performance
- Develop complete Python-based Web applications from start to finish in Django.

Course Material

Supplied

Course Contents

DAY 1

- Set Up
    - The Command Line
    - Install Python 3 on Mac OS X (click here for Windows or Linux)
    - Install Python 3 on Windows
    - Install Python 3 on Linux
    - Virtual Environments
    - Install Django
    - Install Git
    - Text Editors

- Conclusion
- Hello World app
    - Initial Set Up
    - Create an app
    - Views and URLConfs
    - Hello, world!
    - Git
    - Bitbucket
    - Conclusion
- Pages app
    - Initial Set Up
    - Templates
    - Class-Based Views
    - URLs
    - Add an About Page
    - Extending Templates
    - Tests
    - Git and Bitbucket
    - Local vs Production
    - Heroku
    - Additional Files
    - Deploy
    - Conclusion

DAY 2

- Message Board app
    - Initial Set Up
    - Create a database model
    - Activating models
    - Django Admin
    - Views/Templates/URLs
    - Adding new posts
    - Tests
    - Bitbucket
    - Heroku configuration

    - Heroku deployment
    - Conclusion
- Blog app
    - Initial Set Up
    - Database Models
    - Admin
    - URLs
    - Views
    - Templates
    - Static files
    - Individual blog pages
    - Tests
    - Git
    - Conclusion
- Forms
    - Forms
    - Update Form
    - Delete View
    - Tests
    - Conclusion

DAY 3

- User Accounts
    - Log in
    - Updated homepage
    - Log out link
    - Sign up
    - Bitbucket
    - Heroku config
    - Heroku deployment
    - Conclusion
- Custom User Model
    - Set Up
    - Custom User Model
    - Forms

  - Superuser
  - Conclusion
- User Authentication
  - Templates
  - URLs
  - Admin
  - Conclusion

DAY 4

- Bootstrap
  - Pages app
  - Tests
  - Bootstrap
  - Sign Up Form
  - Next Steps
- Password Change and Reset
  - Password Change
  - Customizing password change
  - Password reset
  - Custom Templates
  - Conclusion
- Email
  - SendGrid
  - Custom emails
  - Conclusion

DAY 5

- Newspaper app
  - Articles app
  - URLs and Views
  - Edit/Delete
  - Create page
  - Conclusion

- Permissions and Authorization
    - Improved CreateView
    - Authorizations
    - Mixins
    - LoginRequiredMixin
    - UpdateView and DeleteView
    - Conclusion
- Comments
    - Model
    - Admin
    - Template
    - Conclusion

Duration and pricing

- In Pricing Group A

Certificate

About Our Certificates

Schedule

On the calender on this page below.
If your browser doesn't display the calendar below, please click on this link or try using Google Chrome, alternatively please enquire via our 'Contact Us' page.

Bookings

You can download the course registration form on our home page or by clicking here

Brochure

You may download a pdf copy of this page by clicking here.

Questions

Please email us

We are a member of the Python Software Foundation

Django REST API

This Django REST API Training Course will teach you how to build multiple RESTFul web API’s of increasing complexity from scratch using Django and Django REST Framework

Intended Audience

This Django REST API Training Course is for Django Developers who want to learn how to write RESTFul API’s or how to convert their current websites into RESTFul API’s.

Prerequisites

Django

After this course you should be able to

- Build your own web API from scratch properly using modern best practices.
- Extend any existing Django website into a web API with a minimal amount of code.

Course Material

Supplied

Course Contents

DAY 1

- Introduction
    - Prerequisites
    - Why APIs
    - Django REST Framework
    - Why this book
    - Conclusion
- Web APIs
    - World Wide Web
    - URLs
    - Internet Protocol Suite
    - HTTP Verbs
    - Endpoints
    - HTTP
    - Status Codes
    - Statelessness
    - REST
    - Conclusion

DAY 2

- Library Website and API
    - Traditional Django
    - First app
    - Models
    - Admin
    - Views
    - URLs
    - Webpage
    - Django REST Framework
    - URLs
    - Views
    - Serializers
    - cURL

        - Browsable API
        - Conclusion
- Todo API
        - Initial Set Up
        - Models
        - Django REST Framework
        - URLs
        - Serializers
        - Views
        - Consuming the API
        - Browsable API
        - CORS
        - Conclusion

DAY 3

- Todo React Front-end
        - Install Node
        - Install React
        - Mock data
        - Django REST Framework + React
        - Conclusion
- Blog API
        - Initial Set Up
        - Model
        - Tests
        - Django REST Framework
        - URLs
        - Serializers
        - Views
        - Browsable API
        - Conclusion

DAY 4

- Permissions
    - Create a new user
    - Add log in to the browsable API
    - AllowAny
    - View-Level Permissions
    - Project-Level Permissions
    - Custom permissions
    - Conclusion
- User Authentication
    - Basic Authentication
    - Session Authentication
    - Token Authentication
    - Default Authentication
    - Implementing token authentication
    - Endpoints
    - Django-Rest-Auth
    - User Registration
    - Tokens
    - Conclusion

DAY 5

- Viewsets and Routers
    - User endpoints
    - Viewsets
    - Routers
    - Conclusion
- Schemas and Documentation
    - Schemas
    - Documentation
    - Django REST Swagger
    - Conclusion

Duration and pricing

- In Pricing Group A

Certificate

About Our Certificates

Schedule

On the calender on this page below.
If your browser doesn’t display the calendar below, please click on this link or try using Google Chrome, alternatively please enquire via our ‘Contact Us’ page.

Bookings

You can download the course registration form on our home page or by clicking here

Brochure

You may download a pdf copy of this page by clicking here.

Questions

Please email us

We are a member of the Python Software Foundation