

CERTIFIED PROFESSIONAL DIPLOMA IN FULL STACK JAVA



ABOUT US FULL STACK JAVA

A Full Stack Java developer is proficient in both front-end and back-end development using Java and related technologies. On the front-end, they work with technologies such as HTML, CSS, JavaScript, and frameworks like Angular or React to build dynamic and responsive user interfaces. For back-end development, they utilize Java-based technologies such as Spring, Hibernate, and Java EE to create robust server-side applications. Full Stack Java developers are skilled in integrating front-end and back-end components, utilizing databases like MySQL or PostgreSQL, and deploying applications on cloud platforms or servers. Their expertise spans the entire development lifecycle, from designing the user interface to managing server infrastructure, ensuring seamless functionality and performance across the application.

BENEFITS OF FULL STACK JAVA

- **END-TO-END DEVELOPMENT**
- **IMPROVED FORECASTING**
- **FLEXIBILITY ACROSS INDUSTRIES**
- **OPTIMIZED RESOURCE ALLOCATION**
- **FUTURE-PROOF CAREER**
- **ENCOURAGES CREATIVITY AND INNOVATION**
- **COMPETITIVE ADVANTAGE**
- **FUTURE-PROOF CAREER**

TOPICS COVERED

- Web Designing (Html, CSS, JavaScript & JQuery)
- REACT JS
- JAVA

HTML

Overview of HTML5:

- Introduction HTML
- HTML Basics HTML
- Structure of HTML5
- Different between HTML and HTML5
- Elements HTML5
- Semantic HTML Attributes and Formatting Tags
- HTML Headings
- HTML Paragraph
- HTML Pre tag
- HTML Mark Tag
- HTML br and hr tag
- HTML Image with alt
- HTML Styles
- HTML Formatting
- HTML Quotations
- HTML Computer Code
- ▶ • HTML Comments & Colours HTML CSS,
- Links and Images
- HTML Lists

HTML

- HTML Table
- HTML Blocks
- HTML SVG
- HTML Classes
- HTML iframe - Google Map
- HTML Layout
- HTML Responsive
- HTML JavaScript
- HTML Forms - Labels and Inputs , Radio , Date , Time , Checkbox, Select, Textarea, Button-submit , Input-Required , Max- length and Min- length
- Html Marquee
- Html variables
- Figure Tag , Figure caption ,Video,Audio.
- HTML Canvas



CSS

- Introduction CSS3 CSS3
- Syntax CSS3 How To link css
- in html CSS3 Colours CSS3
- Backgrounds CSS3
- Background Image CSS3
- Borders CSS Padding CSS
- Margin CSS Calc method CSS
- hover CSS importants CSS
- opacity CSS scale CSS skew
- CSS rotate c CSS3 Font Style
- CSS Height/Width CSS
- Animation - Keyframes CSS3
- Gradients CSS3 Postion Fixed
- and Sticky

CSS

- CSS3 Shadows
- CSS3 Text
- CSS3 Fonts
- CSS3 2D Transforms
- CSS3 3D Transforms
- CSS Lists, CSS
- Tables
- CSS Outline
- CSS Display
- CSS Max-width
- CSS Position
- CSS Justify Contant
- CSS Float
- CSS Inline-block
- CSS Align
- CSS class
- CSS element
- CSS Navigation Bar
- CSS Dropdowns
- CSS Tooltips
- CSS3 Images
- CSS Attribute
- Selectors CSS
- Forms CSS Counters

CSS

- CSS3 Buttons
- CSS3 Pagination
- CSS3 Multiple Columns
- Border Image
- Media Queries, CSS3 Responsive

JAVA SCRIPT

- JavaScript Introduction,
- Internal, External javascript
- Data Types
- Js Operators
- If Statements
- Switch, Loops Js
- Function
- Function return
- Function with parameters
- Arrow Functions
- Calculator Projects using Function
- Object
- Arrays Js
- Array Methods
- Array Map, Filter, Reduce
- Maths Methods
- Prototypes
- Try and catch
- ▶ DOM(Document Object) getElementById, GetElementsByName, getElementByTagName, InnerHtml, InnerText property Js
- Validation – Form validation, Email validation Js
- OOPs – JS Class, Object, Constructor, Methods Js Inheritance, Encapsulation, Polymorphism, Abstraction

JQUERY

- Introduction to jQuery
- jQuery Syntax
- jQuery Selectors
- jQuery Events
- Hide, Show, Toggle,
- FadeIn, FadeOut,
- SlideDown, SlideUp, SlideToggle,
- Jquery animate
- JQuery Mouseover and Mouseleave
- JQuery Keypress, Keyup and Keydown
- JQuery Focus, blur, select, submit
- jQuery Click and dblclick
- jQuery CSS
- jQuery Effects
- jQuery HTML
- jQuery Append, prepend, before, after
- jQuery Mobile jQuery Mobile Pages
- Mobile Buttons
- Mobile Navbars
- Mobile Tables
- jQuery Mobile Forms

REACT JS

- REACTJS OVERVIEW
 - React —Features
 - React —Advantages
 - React —Limitations
- REACTJS SETUP
 - Install Global Packages Create the
 - Root Folder Add Dependencies
 - and Plugins Create the Files Set
 - Compiler, Server, and Loaders
 - index.html App.jsx and main.js
 - Running the Server
- REACTJS
 - Using JSX Nested
 - Elements Attributes
 - JavaScript Expressions



REACT JS

- Styling
- Comments
- Naming Convention
- **REACTJS —COMPONENTS**
- Stateless Example
- Stateful Example
- **REACTJS —STATE, EFFECT & REF**
- **REACTJS — PROPS OVERVIEW**
- Using Props27 Default Props
- State and Props
- **REACTJS —PROPS VALIDATION**
- Validating Props
- **REACTJS —COMPONENT API**
- Set State
- Force Update
- Find Dom Node
- **REACTJS —COMPONENT LIFE CYCLE**
- Lifecycle Methods

REACT JS

- Styling
- Comments
- Naming Convention
- **REACTJS —COMPONENTS**
- Stateless Example
- Stateful Example
- **REACTJS —STATE, EFFECT & REF**
- **REACTJS — PROPS OVERVIEW**
- Using Props27 Default Props
- State and Props
- **REACTJS —PROPS VALIDATION**
- Validating Props
- **REACTJS —COMPONENT API**
- Set State
- Force Update
- Find Dom Node
- **REACTJS —COMPONENT LIFE CYCLE**
- Lifecycle Methods

REACT JS

- **REACTJS —FORMS**
- Simple Example
- Complex Example
- **REACTJS —EVENTS**
- Simple Example
- Child Events
- **REACTJS —REFS 51**
- Using Refs..51
- **REACTJS —KEYS 53**
- Using Keys .53
- **REACTJS —ROUTER**
- Install a React Router
- Create Components
- Add a Router
- **REACTJS – REDUX**
- **CONCEPT**
- Redux Elements
- Redux Pros
- **REACTJS —USING FLUX**
- Install Redux

REACT JS

- Create Files and Folders
 - Actions
 - Reducers
 - Store
 - Root Component
 - Other Components
 - **REACTJS —ANIMATIONS**
 - Install React CSS Transitions Group
 - Add a CSS file
 - Appear Animation
 - Enter and Leave Animations
 - **REACTJS HIGHER ORDER COMPONENTS**
 - **REACTJS BEST PRACTICES**
 - **Nodejs – CRUD Operations.**
- And Many More...**

JAVA

CORE JAVA

Java Basics

- Define the scope of variables. Define the structure of a Java class. Create executable Java applications with a main method; run a Java program from the command line; Import other Java packages to make them accessible in your code. Compare and contrast the features and components of Java such as: platform independence, object orientation, encapsulation.

Working with Java Data Types

- Declare and initialize variables (including casting of primitive types). Differentiate between object reference variables and primitive variables. Develop code that uses wrapper classes

Working with Operators and Decision Constructs

- Using Operators and Decision Constructs. Use Java operators; including parentheses to override operator precedence. Test equality between Strings and other objects using == and equals (). Create if and if/else and ternary constructs. Use a switch statement

JAVA

Creating and Using Arrays

- Declare, instantiate, initialize and use a one-dimensional array: Declare, instantiate, initialize and use multi-dimensional array.

Using Loop Constructs

- Create and use while loops. Create and use for loops including the enhanced for loop. Create and use do/while loops. Compare loop constructs. Use break and continue

Java Class Design

- Implement encapsulation. Implement inheritance including visibility modifiers and composition. Implement polymorphism. Object class. Create and use singleton classes and immutable classes. static keyword on initialize blocks, variables, methods, and classes

Working with Methods and Encapsulation

- Create methods with arguments and return values. overloaded methods. Apply the static keyword to methods and fields. Create and overload constructors; including impact on default constructors. Apply access modifiers.

JAVA

Working with Inheritance

- Describe inheritance and its benefits. Develop code that demonstrates the use of polymorphism; including overriding and object type versus reference type. Determine when casting is necessary. Use super and this to access objects and constructors. Use abstract classes and interfaces

Handling Exceptions

- Differentiate among checked exceptions, unchecked exceptions, and Errors. Can create a try-catch block and determine how exceptions alter normal program flow. Describe the advantages of Exception handling. Create and invoke a method that throws an exception. "Recognize common exception classes (such as NullPointerException, ArithmeticException, ArrayIndexOutOfBoundsException, ClassCastException)"

Working with Selected classes from the Java API

- Manipulate data using the String Builder class and its methods. Creating and manipulating Strings. Create and manipulate calendar data using classes from java.time.LocalDateTime, java.time.LocalDate, java.time.LocalTime, java.time.format.
- DateTimeFormatter, java.time.Period . Declare and use an ArrayList of a given type. Write a simple Lambda expression that consumes a Lambda Predicate expression

JAVA

Generics and Collections

- Create and use a generic class. Create and use ArrayList, TreeSet, TreeMap, and ArrayDeque objects. Use java.util.Comparator and java.lang.Comparable interfaces. Collections Streams and Filters. Iterate using forEach methods of Streams and List. Describe Stream interface and Stream pipeline. Filter a collection lambda expressions. Use by using method references with Streams

Lambda Built-in Functional Interfaces

- Use the built-in interfaces included in the java.util. function package such as Predicate, Consumer, Function, and Supplier. Develop code that uses primitive versions of functional interfaces. Develop code that uses binary versions of functional interfaces. Develop code that uses the Unary Operator interface

JAVA

Java Stream AP

- Develop code to extract data from an object using `peek()` and `map()` methods including primitive versions of the `map()` method. Search for data by using search methods of the Stream classes including `findFirst`, `findAny`, `anyMatch`, `allMatch`, `noneMatch`. Develop code that uses the `Optional` class. Develop code and calculation methods. Sort a collection using Stream API. Save results to a collection using the `collect` method and group/partition data using the `Collectors` class. Use `flatMap()` methods in the Stream API

Use Java SE 8 Date/Time API

- Create and manage date-based and time-based events including a combination of date and time into a single object using `LocalDate`, `LocalTime`, `LocalDateTime`, `Instant`, `Period`, and `Duration`. Work with dates and times across time zones and manage changes resulting from daylight savings including `Format` date and times values. Define and create and manage date-based and time-based events using `Instant`, `Period`, `Duration`, and `TemporalUnit`

JAVA

Java I/O Fundamentals

- Read and write data from the console. Use `BufferedReader`, `BufferedWriter`, `File`, `FileReader`, `FileWriter`, `FileInputStream`, `FileOutputStream`, `ObjectOutputStream`, `ObjectInputStream`, and `PrintWriter` in the `java.io` package

Localization

- Read and set the locale by using the `Locale` object. Create and read a `Properties` file. Build a resource bundle for each locale and load a resource bundle in an application

ADVANCED JAVA

JDBC

- Introduction to JDBC. JDBC architecture. java. SQL Package.
- Connection, Statement, ResultSet.
- Prepared Statement. Callable Statement.
- Scrollable and Updatable ResultSet. Batch Updates. ResultSet
- MetaData. Simple Transaction Management.
- Four Levels of JDBC drivers, their pros & cons. Features of JDBC 3.0

SERVLETS

- The need for server-side programming. Introduction to Servlets. Servlet Life Cycle.
- Javax .servlet package. Servlet Config, Servlet Context, Servlet Response. Supplying initialization parameters to Servlets.
- Performing database operations in Servlets.
- Include and forward mechanisms.
- Applying filters to Servlets.

ADVANCED JAVA

- **java. servlet. http Package. Http Servlet Life Cycle. Http request methods GETvs. POST. Http Servlet**
 - Request, Http Servlet Response. Dealing with Http headers & error codes. Session Tracking, purpose .Hidden form fields, Cookies .Http Session, URL rewriting.
 - Event listeners. Web application security.

JSP

- Disadvantages of Servlets.
- Introduction to JSP. JSP Life Cycle.
- Creating dynamic Web content with JSP.
- Scripting elements: Scriptlet. Declaration. Expression.
- XML syntax for JSP elements .JSP directives page, include and tag lib.
- JSP implicit objects. JSP scopes.
- Include and forward mechanism.
- Using a Java bean in a JSP. JSP Model 1 architecture. JSP Model 2 (MVC) architecture.
- Custom Tag Development. Classic Tags, Simple Tags.
- Error Handling in a JSP.
- JSTL. (Java Standard Tag Library)
- Expression Language.Processing XML in a JSP

And Many More...

ADVANCED JAVA

JAVA FRAMEWORK

HIBERNATE

Object Relational Mapping

- Introduction to Hibernate
- Hibernate Mapping
- Collection in Hibernate
- Inheritance in Hibernate
- Hibernate Query Language(HQL)
- Hibernate Criteria API

ADVANCED JAVA

Spring

- Overview
 - Architecture
 - IoC Containers Bean Definition,
 - Scope Dependency Injection
 - with Bean Dependency Injection
 - with Bean Annotation Based
 - Configuration Java Based
 - Configuration
 - MVC with Spring
 - AOP with Spring Framework
 - Spring JDBC Framework
- And Many More...**

SKILLS DEVELOPED BY FULL STACK JAVA

- Critical Thinking
- Coding Mathematics
- Communication
- Problem Solving Risk
- Analysis

CAREER OPPORTUNITIES

- Frontend Web Developer
 - Backend Web Developer
 - UI/UX Developer
 - Web Application Developer
 - Web Analyst
 - Senior Developer
- And Many More...**

FACILITIES OFFERED

- Practical Training on Live Projects
- Complete Placement Assistance
- Interview Preparation
- Global Certification
- Fully functional labs
- Online / Offline Training
- Study Materials
- Expert Level Industry Recognized Training

OUR RECRUITERS



AND MANY MORE....