CAC
We Give Wings To Your Career

JavaScript

CAC Noida is an ISO 9001:2015 certified training center with professional experience that dates back to 2005. The vision is to provide professional education merging corporate culture globally to the youth through technology resourcing and knowledge consulting with emerging technologies. Quality assurance parameters for each stage of training and development are ensured at all levels. The operating office is solely based Noida (U.P) India.

CAC Noida is the well-known JavaScript training center in Noida with high tech infrastructure and friendly environment. We provide hands on practical knowledge and full job assistance with basic as well as advanced level

CAC Noida is one of the best JavaScript training institute in Noida with 100% placement record. CAC Noida has well defined courses and modules with training sessions for developers. At CAC Noida, JavaScript training is conducted by specialist Trainers having experience of more than 10+ years.

CAC Noida is well-equipped JavaScript training center in Noida and we offer job oriented JavaScript training program keeping an eye on industry requirements and future prospects. Each and every one who is part of "CAC Noida" is important to us. Every student has the freedom to discuss and learn. We always take care that right student choose right course.

JavaScript is the one of high in demand course today and CAC Noida provides practical exposure to all the concepts, contents are well-structured to meet the industry requirements.

We are confident that JavaScript training we deliver is at a fantastic standard and are constantly striving to improve and become even better. We believe that JavaScript training should be well planned, well prepared, fit for purpose and delivered by trainers who are motivational and inspirational, trainers who can make learning interesting and will make a difference to your people and your organization.

Training Offer for JavaScript

Introduction JavaScript

- More code, less words
- Exhaustive code and repetition
- Color-coding conventions
- Code examples

JavaScript Objects

- Creating objects
- JavaScript constructors create and return object instances
- The native JavaScript object constructors
- User-defined/non-native object constructor functions
- Instantiating constructors using the new operator
- Creating shorthand or literal values from constructors

- Primitive (aka simple) values
- The primitive values null, undefined, “string”, 10, true, and false are not Objects
- How primitive values are stored/copied in JavaScript
- Primitive values are equal by value
- The string, number, and Boolean primitive values act like objects when used like Objects
- Complex (aka composite) values
- How complex values are stored/copied in JavaScript
- Complex objects are equal by reference
- Complex objects have dynamic properties
- The typeof operator used on primitive and complex values
- Dynamic properties allow for mutable objects
- All constructor instances have constructor properties that point to their constructor
- Function
- Verify that an object is an instance of a particular constructor function
- An instance created from a constructor can have its own independent properties (aka instance properties)
- The semantics of “JavaScript objects” and “Object() objects”

Working with Objects and Properties

- Complex objects can contain most of the JavaScript values as properties
- Encapsulating complex objects in a programmatically beneficial way Getting, setting, and updating an object’s properties using dot notation or bracket Notation
- Deleting object properties
- How references to object properties are resolved
- Using hasOwnProperty to verify that an object property is not from the prototype Chain
- Checking if an object contains a given property using the in operator
- Enumerate (loop over) an object’s properties using the for in loop
- Enhancing and extending objects with Underscore.js

String()

- Conceptual overview of using the String() object
- String() parameters
- String() properties and methods
- String object instance properties and methods

Number()

- Conceptual overview of using the Number() object

- Integers and floating-point numbers
- Number() parameters
- Number() properties
- Number object instance properties and methods

Boolean()

- Conceptual overview of using the Boolean() object
- Boolean() parameters
- Boolean() properties and methods
- Boolean object instance properties and methods
- Non-primitive false Boolean objects convert to true
- Certain things are false, everything else is true

Working with Primitive String, Number, and Boolean Values

- Primitive/literal values are converted to objects when properties are accessed
- You should typically use primitive string, number, and Boolean values

Null

- Conceptual overview of using the null value
- typeof returns null values as “object”

Undefined

- Conceptual overview of the undefined value
- JavaScript ECMA-262 Edition 3 (and later) declares the undefined variable in the global scope

The Head/Global Object

- Conceptual overview of the head object
- Global functions contained within the head object
- The head object vs. global properties and global variables
- Referring to the head object
- The head object is implied and typically not referenced explicitly

Object()

- Conceptual overview of using Object() objects
- Object() parameters
- Object() properties and methods
- Object() object instance properties and methods
- Creating Object() objects using “object literals”
- All objects inherit from Object.prototype

Function()

- Conceptual overview of using Function() objects
- Function() parameters
- Function() properties and methods
- Function object instance properties and methods
- Functions always return a value
- Functions are first-class citizens (not just syntax, but values)
- Passing parameters to a function
- this and arguments values are available to all functions
- The arguments.callee property
- The function instance length property and arguments.length
- Redefining function parameters
- Return a function before it is done (i.e. cancel function execution)
- Defining a function (statement, expression, or constructor)
- Invoking a function (function, method, constructor, or call() and apply())
- Anonymous functions
- Self-invoking function expression
- Self-invoking anonymous function statements
- Functions can be nested
- Passing functions to functions and returning functions from functions
- Invoking function statements before they are defined (aka function hoisting)
- A function can call itself (aka recursion)

The this Keyword

- Conceptual overview of this and how it refers to objects
- How is the value of this determined?
- The this keyword refers to the head object in nested functions
- Working around the nested function issue by leveraging the scope chain
- Controlling the value of this using call() or apply()
- Using the this keyword inside a user-defined constructor function
- The keyword this inside a prototype method refers to a constructor instance

Scope and Closures

- Conceptual overview of JavaScript scope
- JavaScript does not have block scope
- Use var inside of functions to declare variables and avoid scope gotchas
- The scope chain (aka lexical scoping)
- The scope chain lookup returns the first found value
- Scope is determined during function definition, not invocation

- Closures are caused by the scope chain

Function Prototype Property

- Conceptual overview of the prototype chain
- Why care about the prototype property?
- Prototype is standard on all Function() instances
- The default prototype property is an Object() object

Custom Library

Create Own MVC Framework

DOM/CSS Scripting

- Introduction to the Document Object Model (DOM)
- Using the getElementById method
- Modifying Page Content with the DOM
- Manipulating CSS using JavaScript
- Programmatic Access to CSS

Common Applications

- Form Validation and Testing
- Working with Regular Expressions
- User Interaction
- Local Form Processing
- Object Detection
- Creating New Windows
- Adding Content to a Window

Contact Info.

CAC – NOIDA
Address:- D-55, Sector-7, Noida

Phone:- 0120-4269814

Mobile: +91 9212091244

Email:- info@cacnoida.com

Website:- **http://cacnoida.com/**