NUTECH COMPUTER TRAINING INSTITUTE

1682 E. GUDE DRIVE #102, ROCKVILLE, MD 20850

WEB: www.NUTECHTRAINING.COM TEL: 301-610-9300

MCSD Web Applications – Course Outlines

70-480 Programming in HTML5 with JavaScript and CSS3

Toptic 1. Implement and manipulate document structures and objects

- Create the document structure
    - Structure the UI by using semantic markup, including for search engines and screen readers (Section, Article, Nav, Header, Footer, and Aside); create a layout container in HTML
- Write code that interacts with UI controls
    - Programmatically add and modify HTML elements; implement media controls; implement HTML5 canvas and SVG graphics
- Apply styling to HTML elements programmatically
    - Change the location of an element; apply a transform; show and hide elements
- Implement HTML5 APIs
    - Implement storage APIs, AppCache API, and Geolocation API
- Establish the scope of objects and variables
    - Define the lifetime of variables; keep objects out of the global namespace; use the “this” keyword to reference an object that fired an event; scope variables locally and globally
- Create and implement objects and methods
    - Implement native objects; create custom objects and custom properties for native objects using prototypes and functions; inherit from an object; implement native methods and create custom methods

Toptic 2. Implement program flow:

- Iterate across collections and array items; manage program decisions by using switch statements, if/then, and operators; evaluate expressions

- Raise and handle an event
  - Handle common events exposed by DOM (OnBlur, OnFocus, OnClick); declare and handle bubbled events; handle an event by using an anonymous function
- Implement exception handling
  - Set and respond to error codes; throw an exception; request for null checks; implement try-catch-finally blocks
- Implement a callback
  - Receive messages from the HTML5 WebSocket API; use jQuery to make an AJAX call; wire up an event; implement a callback by using anonymous functions; handle the “this” pointer
- Create a web worker process
  - Start and stop a web worker; pass data to a web worker; configure timeouts and intervals on the web worker; register an event listener for the web worker; limitations of a web worker

Toptic 3. Access and secure data:

- Validate user input by using HTML5 elements
  - Choose the appropriate controls based on requirements; implement HTML input types and content attributes (for example, required) to collect user input
- Validate user input by using JavaScript
  - Evaluate a regular expression to validate the input format; validate that you are getting the right kind of data type by using built-in functions; prevent code injection
- Consume data
  - Consume JSON and XML data; retrieve data by using web services; load data or get data from other sources by using XMLHTTPRequest
- Serialize, deserialize, and transmit data
  - Binary data; text data (JSON, XML); implement the jQuery serialize method; Form.Submit; parse data; send data by using XMLHTTPRequest; sanitize input by using URI/form encoding

Toptic 4. Use CSS3 in applications:

- Style HTML text properties
  - Apply styles to text appearance (color, bold, italics); apply styles to text font (WOFF and @font-face, size); apply styles to text alignment, spacing, and indentation; apply styles to text hyphenation; apply styles for a text drop shadow
- Style HTML box properties
  - Apply styles to alter appearance attributes (size, border and rounding border corners, outline, padding, margin); apply styles to alter graphic effects (transparency, opacity, background image, gradients, shadow, clipping); apply styles to establish and change an element's position (static, relative, absolute, fixed)
- Create a flexible content layout
  - Implement a layout using a flexible box model; implement a layout using multi-column; implement a layout using position floating and exclusions; implement a layout using grid alignment; implement a layout using regions, grouping, and nesting
- Create an animated and adaptive UI
  - Animate objects by applying CSS transitions; apply 3-D and 2-D transformations; adjust UI based on media queries (device adaptations for output formats, displays, and representations); hide or disable controls
- Find elements by using CSS selectors and jQuery
  - Choose the correct selector to reference an element; define element, style, and attribute selectors; find elements by using pseudo-elements and pseudo-classes

70-486 Developing ASP.NET MVC Web Applications

Toptic 1. Design the application architecture:

- Plan the application layers

- Plan data access; plan for separation of concerns; appropriate use of models, views, and controllers; choose between client-side and server side processing; design for scalability
- Design a distributed application
  - Design a hybrid application (on-premises versus off-premises, including Azure), plan for session management in a distributed environment, plan web farms
- Design and implement the Azure role life cycle
  - Identify and implement Start, Run, and Stop events; identify startup tasks (IIS configuration [app pool], registry configuration, third-party tools)
- Configure state management
  - Choose a state management mechanism (in-process and out of process state management), plan for scalability, use cookies or local storage to maintain state, apply configuration settings in web.config file, implement sessionless state (for example, QueryString)
- Design a caching strategy
  - Implement page output caching (performance oriented), implement data caching, implement HTTP caching, implement Azure caching
- Design and implement a WebSocket strategy
  - Read and write string and binary data asynchronously (long-running data transfers), choose a connection loss strategy, decide a strategy for when to use WebSockets, implement SignalR
- Design HTTP modules and handlers
  - Implement synchronous and asynchronous modules and handlers, choose between modules and handlers in IIS

Toptic 2. Design the user experience:

- Apply the user interface design for a web application
  - Create and apply styles by using CSS, structure and lay out the user interface by using HTML, implement dynamic page content based on a design
- Design and implement UI behavior
  - Implement client validation, use JavaScript and the DOM to control application behavior, extend objects by using prototypal inheritance, use AJAX to make partial page updates, implement the UI by using JQuery
- Compose the UI layout of an application

- Implement partials for reuse in different areas of the application, design and implement pages by using Razor templates (Razor view engine), design layouts to provide visual structure, implement master/application pages
- Enhance application behavior and style based on browser feature detection
  - Detect browser features and capabilities; create a web application that runs across multiple browsers and mobile devices; enhance application behavior and style by using vendor-specific extensions, for example, CSS
- Plan an adaptive UI layout
  - Plan for running applications in browsers on multiple devices (screen resolution, CSS, HTML), plan for mobile web applications

Toptic 3. Develop the user experience:

- Plan for search engine optimization and accessibility
  - Use analytical tools to parse HTML, view and evaluate conceptual structure by using plugs-in for browsers, write semantic markup (HTML5 and ARIA) for accessibility (for example, screen readers)
- Plan and implement globalization and localization
  - Plan a localization strategy; create and apply resources to UI, including JavaScript resources; set cultures; create satellite resource assemblies
- Design and implement MVC controllers and actions
  - Apply authorization attributes, global filters, and authentication filters; specify an override filter; implement action behaviors; implement action results; implement model binding
- Design and implement routes
  - Define a route to handle a URL pattern, apply route constraints, ignore URL patterns, add custom route parameters, define areas
- Control application behavior by using MVC extensibility points
  - Implement MVC filters and controller factories; control application behavior by using action results, viewengines, model binders, and route handlers
- Reduce network bandwidth
  - Bundle and minify scripts (CSS and JavaScript), compress and decompress data (using gzip/deflate; storage), plan a content delivery network (CDN) strategy

Toptic 4. Troubleshoot and debug web applications:

- Prevent and troubleshoot runtime issues
    - Troubleshoot performance, security, and errors; implement tracing, logging (including using attributes for logging), and debugging (including IntelliTrace); enforce conditions by using code contracts; enable and configure health monitoring (including Performance Monitor)
- Design an exception handling strategy
    - Handle exceptions across multiple layers, display custom error pages using global.asax or creating your own HTTPHandler or set web.config attributes, handle first chance exceptions
- Test a web application
    - Create and run unit tests (for example, use the Assert class), create mocks; create and run web tests, including using Browser Link; debug a web application in multiple browsers and mobile emulators
- Debug an Azure application
    - Collect diagnostic information by using Azure Diagnostics API and appropriately implement on demand versus scheduled; choose log types

Toptic 5. Design and implement security:

- Configure authentication
    - Authenticate users; enforce authentication settings; choose between Windows, Forms, and custom authentication; manage user session by using cookies; configure membership providers; create custom membership providers; configure ASP.NET Identity
- Configure and apply authorization
    - Create roles, authorize roles by using configuration, authorize roles programmatically, create custom role providers, implement WCF service authorization
- Design and implement claims-based authentication across federated identity stores
    - Implement federated authentication by using Azure Access Control Service; create a custom security token by using Windows Identity Foundation; handle token formats (for example, oAuth, OpenID, Microsoft Account, Google, Twitter, and Facebook) for SAML and SWT tokens
- Manage data integrity
    - Apply encryption to application data, apply encryption to the configuration sections of an application, sign application data to prevent tampering

- Implement a secure site with ASP.NET
    - Secure communication by applying SSL certificates; salt and hash passwords for storage; use HTML encoding to prevent cross-site scripting attacks (ANTI-XSS Library); implement deferred validation and handle unvalidated requests, for example, form, querystring, and URL; prevent SQL injection attacks by parameterizing queries; prevent cross-site request forgeries (XSRF)

70-487 Developing Microsoft Azure and Web Services

Toptic 1. Accessing data:

- Choose data access technologies
    - Choose a technology (ADO.NET, Entity Framework, WCF Data Services, Azure storage) based on application requirements
- Implement caching
    - Cache static data, apply cache policy (including expirations); use CacheDependency to refresh cache data; query notifications
- Implement transactions
    - Manage transactions by using the API from System.Transactions namespace; implement distributed transactions; specify transaction isolation level
- Implement data storage in Azure
    - Access data storage in Azure; choose data storage mechanism in Azure (blobs, tables, queues, SQL Database); distribute data by using the Content delivery network (CDN); handle exceptions by using retries (SQL Database); manage Azure Caching
- Create and implement a WCF Data Services service
    - Address resources; implement filtering; create a query expression; access payload formats (including JSON); use data service interceptors and service operators
- Manipulate XML data structures
    - Read filter, create, modify XML data structures; Manipulate XML data by using XMLReader, XMLWriter, XMLDocument, XPath, LINQ to XML; transform XML by using XSLT transformations

Toptic 2. Query and manipulate data by using the Entity Framework

-
    - Query, update, and delete data by using DbContext; build a query that uses deferred execution; implement lazy loading and eager loading; create and run compiled queries; query data by using Entity SQL; perform asynchronous operations using Entity Framework; map a stored procedure
- Query and manipulate data by using Data Provider for Entity Framework
    - Query and manipulate data by using Connection, DataReader, and Command from the System.Data.EntityClient namespace; perform synchronous and asynchronous operations; manage transactions (API); programmatically configure a Data Provider
- Query data by using LINQ to Entities
    - Query data by using LINQ operators (for example, project, skip, aggregate, filter, and join); log queries and database commands; implement query boundaries (IQueryable vs. IEnumerable); implement async query
- Query and manipulate data by using ADO.NET
    - Query and manipulate data by using Connection, DataReader, Command, DataAdapter, DataSet; perform synchronous and asynchronous operations; manage transactions (API)
- Create an Entity Framework data model
    - Structure the data model using table per type, table per class, table per hierarchy; choose and implement an approach to manage a data model (code first vs. model first vs. database first); implement POCO objects; describe a data model by using conceptual schema definitions, storage schema definition, mapping language (CSDL, SSDL, MSL), and Custom Code First Conventions

Toptic 3. Designing and implementing WCF Services:

- Create a WCF service
    - Create contracts (service, data, message, callback, and fault); implement message inspectors; implement asynchronous operations in the service
- Configure WCF services by using configuration settings
    - Configure service behaviors; configure service endpoints; configure bindings including WebSocket bindings; specify a service contract; expose service metadata (XSDs, WSDL, and metadata exchange endpoint); configure message compression and encoding
- Configure WCF services by using the API

- Configure service behaviors; configure service endpoints; configure binding; specify a service contract; expose service metadata (XSDs, WSDL, and metadata exchange); WCF routing and discovery features

- Secure a WCF service
  - Implement message level security, implement transport level security; implement certificates; design and implement multiple authentication modes
- Consume WCF services
  - Generate proxies by using SvcUtil; generate proxies by creating a service reference; create and implement channel factories
- Version a WCF service
  - Version different types of contracts (message, service, data); configure address, binding, and routing service versioning
- Create and configure a WCF service on Azure
  - Create and configure bindings for WCF services (Azure SDK—extensions to WCF); relay bindings to Azure using service bus endpoints; integrate with the Azure service bus relay
- Implement messaging patterns
  - Implement one way, request/reply, streaming, and duplex communication; implement Azure Service Bus and Azure Queues
- Host and manage services
  - Manage services concurrency (single, multiple, reentrant); create service hosts; choose a hosting mechanism; choose an instancing mode (per call, per session, singleton); activate and manage a service by using AppFabric; implement transactional services; host services in an Azure worker role

Toptic 4. Creating and consuming Web API-based services:

- Design a Web API
  - Define HTTP resources with HTTP actions; plan appropriate URI space, and map URI space using routing; choose appropriate HTTP method (get, put, post, delete) to meet requirements; choose appropriate format (Web API formats) for responses to meet requirements; plan when to make HTTP actions asynchronous; design and implement routes
- Implement a Web API
  - Accept data in JSON format (in JavaScript, in an AJAX callback); use content negotiation to deliver different data formats to clients; define actions and parameters to handle data binding;

use HttpMessageHandler to process client requests and server responses; implement dependency injection, along with the dependency resolver, to create more flexible applications; implement action filters and exception filters to manage controller execution; implement asynchronous and synchronous actions; implement streaming actions; implement SignalR; test Web API web services

- Secure a Web API
    - Implement HTTPBasic authentication over SSL; implement Windows Auth; prevent cross-site request forgery (XSRF); design, implement, and extend authorization and authentication filters to control access to the application; implement Cross Origin Request Sharing (CORS); implement SSO by using OAuth 2.0; configure multiple authentication modes on a single endpoint
- Host and manage Web API
    - Host Web API in an ASP.NET app; self-host a Web API in your own process (a Windows service) including Open Web Interface for .NET (OWIN); host services in an Azure worker role; restrict message size; configure the host server for streaming
- Consume Web API web services
    - Consume Web API services by using HttpClient synchronously and asynchronously; send and receive requests in different formats (JSON/HTML/etc.); request batching

Toptic 5. Deploying web applications and services:

- Design a deployment strategy
    - Create an IIS install package; deploy to web farms; deploy a web application by using XCopy; automate a deployment from TFS or Build Server
- Choose a deployment strategy for an Azure web application
    - Perform an in-place upgrade and VIP swap; configure an upgrade domain; create and configure input and internal endpoints; specify operating system configuration; deploy applications using Azure Web Site
- Configure a web application for deployment
    - Switch from production/release mode to debug mode; use SetParameters to set up an IIS app pool; set permissions and passwords; enable and monitor ASP.NET App Suspend; configure WCF endpoints (including HTTPS protocol mapping), bindings, and behaviors; transform web.config by using XSLT (for example, across development, test, and production/release environments); configure Azure configuration settings

- Manage packages by using NuGet
    - Create and configure a NuGet package; install and update an existing NuGet package; connect to a local repository cache for NuGet, set up your own package repository
- Create, configure, and publish a web package
    - Create an IIS InstallPackage; configure the build process to output a web package; apply pre- and post- condition actions to ensure that transformations are correctly applied; include appropriate assets (web content, certificates)
- Share assemblies between multiple applications and servers
    - Prepare the environment for use of assemblies across multiple servers (interning); sign assemblies by using a strong name; deploy assemblies to the global assembly cache; implement assembly versioning; create an assembly manifest; configure assembly binding redirects (for example, from MVC4 to MVC5)