IBM Full Stack Development Coursera Course Materials

Introduction to Cloud Computing

1.1 Introduction to cloud computing

- Cloud computing is the delivery of on-demand computing resources over the internet on a pay-as-you-go basis.
- Resources are dynamically assigned and reassigned among multiple users and scaled up and down in response to users' needs.
- The origins of cloud computing can be traced back to the mainframes of the 1950s, with virtualization technologies and hypervisors serving as catalysts for the emergence of modern-day cloud computing.
- Organizations must consider their business needs, investment viability, and risk capacity to create a cloud adoption strategy that delivers desired benefits without causing business disruptions or security, compliance, or performance issues.
- Cloud adoption is growing faster than predicted. Driving this technological wave are cloud service providers with a host of services ranging from infrastructure to platform and software services. Some major cloud providers of our time include AWS, Alibaba Cloud, Google, IBM, and Microsoft Azure.

1.2 Business case for cloud computing

- The adoption of cloud technologies enables enterprises, big and small, to be agile, innovative, and competitive and to create differentiated customer experiences. The questions organizations are asking are not whether they should move to the cloud but rather what strategy they should adopt to move to the cloud.
- Some case studies that demonstrate the impact businesses have created by adopting the cloud:
    - American Airlines adopting cloud technologies to deliver customer value rapidly across its enterprise.
    - UBank leveraging cloud platform services to give more control to their developers, thereby removing barriers to innovation.
    - Bitly leveraging the scalability offered by cloud infrastructure for low-latency delivery to its geographically dispersed enterprise customers.
    - ActivTrades leveraging the infrastructure, storage, network, and security offerings on the cloud to accelerate the execution and delivery of new functions in their online trading systems to their customers.

1.3 Emerging technologies accelerated by cloud

- Emerging technologies powered by the cloud are disrupting existing business models and creating unprecedented opportunities for businesses to grow, innovate, and create value for their customers.
- Some case studies that demonstrate how the use of emerging technologies on the cloud is creating value for millions around the world:
    - The use of the Internet of Things on the cloud to combat poaching of endangered rhinos in South Africa.

    - Artificial intelligence on the cloud is being leveraged to deliver unique digital experiences to millions of fans around the world by the United States Tennis Association.
    - Blockchain on the cloud helps farmers reduce waste by building traceability and transparency in the food supply chain.
    - The use of data analytics for driving predictive maintenance solutions for a city's infrastructure by KONE.

2. Cloud Computing Models

2.1 Service models

- Cloud computing allows us to utilize technology as a service, leveraging remote resources on-demand, on a pay-as-you-go model. There are three main service models available on the cloud: Infrastructure as a service (LaaS), platform as a service (PaaS), and software as a service (SaaS).
    - With IaaS, the cloud provider manages physical resources.
    - With PaaS, the provider manages the platform infrastructure.
    - In the SaaS model, the provider hosts and manages the applications and data.
- Infrastructure as a service is a form of cloud computing that delivers fundamental computer, network, and storage resources to consumers on-demand, over the network, on a pay-as-you-go basis.
- The key components of cloud infrastructure are:
    - Physical data centers
    - Compute
    - Network
    - Storage
- Platform as a service is a cloud computing model that provides customers with a complete platform: Hardware, software, and infrastructure.
- The high level of abstraction, support services, runtime environments, rapid deployment mechanisms, and middleware capabilities distinguish PaaS clouds.
- PaaS advantages are:
    - Scalability
    - Faster time to market products and services
    - Greater agility and innovation
- Software as a service is a cloud offering that provides users access to a service provider's cloud-based software.
- SaaS characteristics are:
    - Multitenant architecture
    - Security, compliance, and maintenance
    - Customization of applications
    - Subscription model
    - Scaling
- SaaS advantages are:
    - Direct procurement of solutions
    - Improved workforce productivity and efficiency
    - Enable distribution of software costs

2.2 Deployment models

- Deployment models indicate where the infrastructure resides, who owns and manages it, and how cloud resources and services are available to users. There are four main deployment models available on the cloud: Public, private, hybrid, and community.
- In the public cloud model, the service provider owns, manages, provisions, and maintains the physical infrastructure, such as data centers, servers, networking equipment, and storage, with users accessing virtualized computing, networking, and storage resources as services.
- In the private cloud model, the provider provisions the cloud infrastructure for exclusive use by a single organization. The private cloud infrastructure can be internal to the organization and run on-premises. Or it can be on a public cloud, as in the case of virtual private clouds (VPC), and be owned, managed, and operated by the cloud provider.
- In the hybrid cloud model, an organization's on-premises private cloud and a third-party, public cloud are connected as a single, flexible infrastructure that leverages the features and benefits of both public and private clouds.
- In the community cloud model, the provider provisions the cloud infrastructure for use by a community of organizations with shared concerns. One or more organizations in the community, a third-party provider, or both are responsible for the ownership, management, and operation of this infrastructure.

3. Components of Cloud Computing

3.1 Cloud infrastructure

- Cloud infrastructure comprises data centers, storage, networking components, and computing resources.
- Virtualization is the process of creating a software-based version of physical resources, made possible by hypervisors.
- A few different types of virtual machines can be provisioned on the cloud. These include:
    - Shared or public cloud VMs are provider-managed, multi-tenant deployments that can be provisioned on-demand with predefined sizes.
    - Transient or spot VMs that use unused capacity in a cloud data center.
    - Reserved VMs that allow you to reserve capacity and guarantee resources for future deployments.
    - Dedicated hosts that offer single-tenant isolation.
- Bare metal servers are single-tenant physical servers dedicated to a single customer. Bare metal servers fulfill the demanding needs of high-performance computing (HPC) and data-intensive applications. They are ideal for applications that have a high degree of security or compliance requirements.
- Networking capabilities in the cloud are delivered as a service rather than in the form of rack-mounted devices. Cloud resources such as VMs (or VSIs), storage, network connectivity, and load balancers are deployed into subnets within virtual private clouds (VPCs). Using private and public subnets allows users to deploy multi-tier enterprise applications securely. Load balancers distribute the traffic and allow applications to be responsive.
- Containers are executable units of software in which application code, its libraries, and its dependencies are packaged in a common way, so that it can be run anywhere, from desktops to traditional IT to the cloud. Containers are more lightweight and consume fewer resources than virtual machines, helping streamline the development and deployment of cloud native applications.

3.2 Cloud storage and content delivery networks

- Cloud storage is available in four main types: Direct attached, file, block, and object storage. These storage types differ in how they can be accessed, the capacity they offer, how much they cost, the types of data they are best suited to store, and their read-write speed.
- Direct attached (or local) storage is storage presented directly to a cloud-based server and is effectively either within the host server chassis or within the same rack.
- File storage is typically presented to compute nodes as a Network File System (NFS), which means that the storage is connected to compute nodes over a standard Ethernet network.
- Block storage is presented to compute nodes using high-speed fiber connections, typically provisioned in volumes, which are mounted onto a compute node.
- Object storage is accessed via an API and doesn't need an underlying compute node.
- Object storage offers infinite capacity as you can keep adding files to it and just pay for what you use. Compared to the other storage types, object storage is the slowest in terms of read and write speeds.
- A content delivery network (CDN) is a distributed server network that accelerates internet content delivery by delivering temporarily stored or cached copies of website or media content to users based on their geographic location.

4. Emergent Trends and Practices

4.1 Hybrid multi-cloud, microservices, and serverless

- Hybrid multi-cloud is a cloud adoption strategy that allows public clouds, private clouds, and on-premises IT to interoperate seamlessly while leveraging the best cloud-based services from different public cloud providers.
- Microservices architecture is an approach in which an application is built as a collection of loosely coupled and independently deployable components or services, leading to efficient development, maintenance, and upgradation cycles.
- Serverless computing is an approach to computing that offloads responsibility for common infrastructure management tasks for application runtimes to cloud providers, allowing developers to focus their time and effort on development and testing and not have to worry about provisioning, maintaining, and scaling compute resources.

4.2 Cloud native applications, DevOps, and application modernization

- Cloud native applications are applications that are built or refactored to work in the cloud environment. These applications, developed using DevOps methodologies, consist of microservices packaged in containers that can run in any environment, making it possible to create and update features in quick iterative cycles.
- DevOps is a collaborative approach that enables development and operations teams to continuously deliver software in quick iterative cycles while reducing overhead, duplication, and rework. DevOps' tools, practices, and processes help tackle the complexities and challenges posed by the cloud, allowing solutions to be delivered and updated quickly and reliably.
- Application modernization helps organizations accelerate digital transformation, take advantage of new technologies and services, and become more responsive to changing

market dynamics. Cloud computing is one of the key enablers of application modernization.

5. Cloud Security, Monitoring, Case Studies, Jobs

5.1 Cloud security and monitoring

- Cloud security refers to the policies, technological procedures, services, and solutions designed to secure enterprise applications and data on the cloud against insider threats, data breaches, compliance issues, and organized security threats.
- Cloud security is a shared responsibility between the cloud provider and the user organization.
- Security architecture and methods for achieving continuous security need to be embedded throughout the life cycle of an application to ensure that the application runs on a safe platform, the code is free from vulnerabilities, and the operational risks are understood.
- Identity and access management, also known as access control, helps authenticate and authorize users and provides user-specific access to cloud resources, services, and applications.
- As part of their identity and access management services, most cloud providers offer users the ability to define access groups and create access policies that define permissions for users on account resources.
- Cloud encryption, often called the last line of defense, encrypts data and provides robust data access control, key management, and certificate management.
- Data needs encryption in three states:
    - **Encryption at rest:** Protecting data while it is stored.
    - **Encryption in transit:** Protecting data while it is transmitted from one location to another.
    - **Encryption in use:** Protecting data when used in memory.
- There needs to be active monitoring of all connected systems and cloud-based services to maintain visibility of all data exchanges between public, private, and hybrid cloud environments. This ensures that the cloud provides a trusted platform that can securely integrate with your enterprise data centers.

5.2 Case studies and jobs

- Businesses all over the world are realizing tangible benefits from the use of cloud technologies and services, including:
    - The Weather Company migrating to the cloud to reliably deliver critical weather data at high speed, especially during major weather events such as hurricanes and tornadoes.
    - American Airlines using the cloud platform and technologies to deliver digital self-service tools and customer value more rapidly across its enterprise.
    - Cementos Pacasmayo achieving operational excellence and insight to help drive strategic transformation and reach new markets using cloud services.
    - Welch choosing cloud storage to drive business value from hybrid cloud.
    - LiquidPower using cloud-based SAP applications to fuel business growth.
- The market size of the cloud services industry is nearly three times the growth of overall IT services, increasing the need for qualified cloud computing professionals. Some common job roles available in this domain include those of cloud software engineers, cloud integration specialists, cloud data engineers, cloud security engineers, cloud DevOps engineers, and cloud solution architects.

HTML, CSS & JS

1. Introduction to Application Development

- Front-end developers work on the parts of the website or app that the user sees and interacts with.
- Back-end developers work on the logic and functionality that keeps the website or app running and responding to users' inputs.
- Full-stack developers have both sets of skills.
- Front-end developers and back-end developers work closely together.
- Frameworks and libraries extend the functionality of coding languages such as JavaScript and Python.
- Common languages for front-end development include HTML, CSS, and JavaScript.
- Common languages and frameworks for back-end development include Python, Django, and Flask.
- Version control systems keep track of changes and resolve conflicts between them.
- Continuous Integration with Continuous Delivery/Deployment (CI/CD) is the best practice developers use to deliver frequent changes reliably.

2. HTML Overview

- HTML provides the basic structure and content for a website using tags.
- Tags represent the elements of an HTML page.
- The HTML DOM tree describes how a website is structured.
- HTML uses APIs to enhance the user experience, providing features for advanced animation, audio, and video.
- Scripting provides a more interactive user experience when browsing websites. It is recommended to not rely on scripting, as it can be disabled.
- HTML5 sandboxes help manage iframe mashups.
- HTML5 Browser Support tables describe which browsers support which HTML5 features.
- JavaScript is used to check if an element is supported by a browser.
- CSS provides consistent style and design throughout the website.
- There are two types of CSS layouts used to design websites: Fluid and fixed.

3. CCS Overview and HTML5 Elements

- CSS creates a uniform look throughout each element of each website page.
- CSS is usually coded in external style sheets and creates base styles for a website.
- CSS frameworks assist in implementing UI elements and creating dynamic web pages.
- CSS has two types of frameworks:
    - Utility-first frameworks, which provide utility classes to help in building one's own styles and layouts.
    - Component frameworks, which provide a wide selection of pre-styled components and templates that can be implemented on a website.
- Plain (Vanilla) CSS lets developers write the styles and layouts of a website.
- HTML5 elements provide structure and function to websites.
- HTML5 uses the <input> tag to allow users to input information. It has many different types, including:
    - Color
    - Date
    - Datetime-local
    - Email
    - Number

    - Range
    - Search
    - URL

4. JavaScript Programming for Web Applications

- JavaScript is a scripting language that enables developers to add dynamic content to web pages.
- JavaScript variables are declared using the var keyword and take their type from the value assigned.
- Program execution is controlled by statements like If…Then…Else, Switch, For loops, and While loops.
- JavaScript uses blocks of code called functions that can be called from anywhere in the script.
- New methods and properties can be added to an object by modifying the prototype for that object.
- Prototypes allow you to define properties and methods for all instances of a specific object.
- Client-side scripts are programs that accompany HTML documents and are used by developers to incorporate more interactive elements.
- The script tag can incorporate a script within an HTML document or call a script from an external file.
- The Document Object Model (DOM) is the programming interface between HTML or XHTML and JavaScript.
- Developers can access HTML DOM elements from JavaScript scripts using the correct DOM notation.
- APIs are often used to access HTML DOM elements in web pages.

Git and Github

1.1 Getting started with Git and GitHub

- A distributed version control system (DVCS) keeps track of changes to code, regardless of where it is stored. This allows multiple users to work on the same codebase or repository, mirroring the codebase on their own computers if needed, while the distributed version control software helps manage synchronization amongst the various codebase mirrors.
- Repositories are storage structures that:
    - Store the code
    - Track issues and changes
    - Enable you to collaborate with others
- GitHub is one of the most popular web-hosted services for Git repositories. GitLab, Bitbucket, and Beanstalk are examples of hosted version control systems.

2. Using Git Commands and Managing GitHub Projects

2.1 GitHub workflows with branches and Git commands

- Branches are used to isolate changes to code. When the changes are complete, they can be merged back into the main branch.
- Repositories can be cloned to make it possible to work locally, then sync changes back to the original.

- Repositories can be forked to be used as a base for a new project, or so that the developer can work independently.
- A pull request (PR) can be submitted to have your changes reviewed and merged.
- Large projects include people working in different roles:
    - Developer: Creates code
    - Integrator: Manages changes made by developers
    - Repository Administrator: Configures and maintains access to the repository

Frontend with React

1. Building Rich Front-End Applications with React and ES6

- React is an efficient, flexible JavaScript library for building user interfaces.
- New features introduced in JavaScript as a part of ES6 are let, const, arrow functions, promise, and class.
- The main benefits of using JSX are that you can leverage the full power of JavaScript in HTML and avoid learning or using a templating language. It allows React to show useful errors and warning messages.
- The four types of React components are Functional, Class, Pure, and High-order components.
- Functional components are most useful when the lifecycle of the component does not have to be managed.
- Class components are more versatile.

2. React Components

- State is a plain JavaScript object used by React to represent information about the component's current situation.
- Props is a shortened form for properties, and they are used to pass data between React components in a unidirectional flow from parent to child.
- You can pass data between components from parent to child using properties, from child to parent using callbacks, and between siblings.
- Components are created or mounted on the DOM; they grow by updating and then die or are unmounted on DOM. This is referred to as a component lifecycle.
- React components can be tested using Mocha, Chai, and Sinon, but preferred approaches are using Jest and React Testing Library.

3. Advanced React

- Hooks provide a way to use functionalities such as context or state without classes.
- Inputs in React can be one of two types: controlled or uncontrolled.
- Redux is a state management library often used with React to handle the state of your application.
- The Redux elements involved in updating the component properties are action, store, and reducer.
- You can interact with asynchronous data in your React Redux app using middleware.
- The data flow in the React-Redux application is unidirectional.

Backend with Node and Express

1. Introduction to Server-Side JavaScript

- Back-end technologies include various types of servers and supporting infrastructures such as programming languages, frameworks, and other hardware.
- Node.js is the server-side component of JavaScript. Using Node.js can improve application performance, and express.js is a framework that helps you build Node.js applications.
- The “require” statement can be called from anywhere in the app code, is bound dynamically, and is synchronous, whereas the “import” statement can only be called at the beginning of a file, is bound statically, and is asynchronous.
- Client-side JavaScript is used to process front-end user interface elements, and server-side JavaScript is used to enable access to different kinds of servers and web applications.
- With server-side JavaScript, Node.js applications process and route web service requests from the client.
- To make a function or a value available to Node.js applications that import your module, add a property to the implicit exports object.
- Core modules include bare minimum functionality, local modules are those that you create for your application, and the Node.js community creates third-party modules.
- A local install means only the application within the directory of the installed can access the package, whereas a global install means that any application on the machine can access the package.

2. Asynchronous I/O with Callback Programming

- Asynchronous network operations can be handled using callback functions to prevent blocking JavaScript code.
- A callback function must invoke another callback function to pass a message from the Node.js module back to the main application after the Node.js module receives a response message.
- Nested callbacks can be difficult to read and debug. Inversion of control causes trust issues when dealing with third-party code.
- Promise objects are most useful for operations that are time-consuming and can block resources.
- JSON.parse() and JSON.stringify() are two methods used to parse JSON objects.

3. Express Web Application Framework

- Developers rely on third-party packages to extend Node.js.
- You can use the npm application to manage Node.js packages in your Node.js framework installation.
- The model-view-controller MVC architecture style divides a back-end application into three parts: the model, the view, and the controller.
- REST API frameworks use HTTP methods to communicate with each other.
- Express abstracts low-level details.
- Routing can be handled at the application level or router level.
- Five types of middleware are as follows: application level, router level, error handling, built-in middleware, and third party.
- Template rendering is the ability of the server to fill in dynamic content.
- The npm jsonwebtoken package should be required in an Express application to authenticate a user.

Python for data science, AI, and development

1.1 Types

- A type is how Python represents different types of data. For instance, the three data types are int, float, and string.
- The data type int stands for an integer, float stands for float, essentially a real number, and the data type string is a sequence of characters.
- Typecasting is a process that can be used to change the type of expression in Python. The type int can be changed to float. A string containing an integer value can also be converted into int.
- Boolean is another important type in Python. A Boolean can take on two values: True or False.

1.2 Expressions and variables

- Expressions are operations Python performs. The numbers are referred to as operands, whereas the math symbols are referred to as operators.
- The addition operation can be performed using an addition symbol (+). To perform the operation of subtraction, use the subtraction sign (–). For multiplication operations, use the asterisk symbol (*), and for division operations, use the forward slash (/).
- Python follows mathematical conventions when performing mathematical expressions.
- Variables are used to store values. To assign a new value to any variable, use the assignment operator, i.e., the equal sign.

1.3 String operations

- In Python, a string is a sequence of characters. A string is contained within two quotes. A string can be spaces or digits. A string can also be special characters.
- A string can be considered as an ordered sequence. Each element in the sequence can be accessed using an index represented by the array of numbers.
- The len command is used to obtain the length of the string.
- Backslashes represent the beginning of escape sequences.
    - Backslash n (i.e. \n) represents a new line. The output is given by a new line after the backslash "n" is encountered.
    - Backslash t (i.e. \t) represents a tab. The output is given by a tab where the \t is.
    - To place a backslash in your string, use a double backslash.
- The string method "Upper" converts lowercase characters to uppercase characters.

2. Python Data Structures

2.1 List and tuples

- Lists and tuples are called compound data types and are one of the key types of data structures in Python.
- Tuples are an ordered sequence and are expressed as comma-separated elements within parentheses. Each element of a tuple can be accessed via an index.
- Tuples are immutable, which means they cannot be changed.
- A tuple can contain other tuples as well as other complex data types, which is called nesting.
- Lists are also an ordered sequence and are represented with square brackets.
- The one key difference between tuples and lists is that the lists are mutable.

2.2 Dictionaries

- Dictionaries are a type of collection in Python. The dictionary has keys and values.
    - The key is analogous to the index. They are like addresses, but they do not have to be integers. They are usually characters.
    - The values are similar to the elements in a list and contain information.
- To create a dictionary, use curly brackets. The keys are the first elements. They must be immutable and unique. Each key is followed by a value separated by a colon. The values can be immutable, mutable, and duplicates. Each key and value pair is separated by a comma.

2.3 Sets

- A set is a type of collection that is unordered and consists of unique elements.
- To define a set, place the different elements of a set in a curly bracket separated by commas. Remove the duplicate elements.
- A list can be converted to a set by using the function set, which is called type casting. Simply use the list as the input to the function set. The result will be a list converted to a set.
- A Venn diagram is a tool that uses shapes to represent sets.
- To add an item to a set, use the add method. Put the set name followed by a dot, then the add method.
- To remove an item from a set, use the remove method. Put the set name followed by a dot, then the remove method.
- The intersection of two sets is a new set containing elements that are in both of those sets. The intersection is defined in terms of "and." In Python, we use an ampersand (&) to find the intersection of the two sets.
- The union of two sets is the new set of elements that contain all the items in both sets.

3. Python Programming Fundamentals

3.1 Conditioning and branching

- Comparison operations compare some values or operands and produce a Boolean (True or False) based on some condition.
- The equality operator denoted with two equal signs can be used to determine if two values are equal.
- The inequality test uses an exclamation mark preceding the equal sign. If two operands are not equal, then the condition becomes true.
- Branching allows us to run different statements for different inputs. For instance, it is helpful to think of an if statement as a locked room. If this statement is true, you can enter the room and your program can run some predefined task. If the statement is false, your program will skip the task.
- The elif statement, short for else if, allows us to check additional conditions if the preceding condition is false. If the condition is true, the alternate expressions will be run.
- Logic operations take Boolean values and produce different Boolean values.
    - The first operation is the not operator. If the input is true, the result is false. Similarly, if the input is false, the result is true.
    - The OR operator takes in the two values and produces a new Boolean value. It only produces a false if all the Boolean values are false.
    - The AND operator takes in the two values and produces a new Boolean value. It only produces a true if all the Boolean values are true.

3.2 Loops

- The range function outputs and orders sequence as a list. If the input is a positive integer, the output is a sequence. The sequence contains the same number of elements as the input but starts at zero.
- If the range function has two inputs where the first input is smaller than the second input, the output is a sequence that starts at the first input. Then, the sequence iterates up to but not including the second number.
- Loops perform a task over and over. The two such loops are for loop and while loop.

3.3 Functions

- Functions take some input and then produce some output or change.
- The function len takes in an input of type sequences, such as a string or list, or a type collection, such as a dictionary or set, and returns the length of that sequence or collection.
- The function sum takes in an iterable like a tuple or list and returns the total of all the elements.
- The function sorted returns a new sorted list or tuple.
- The function mult multiplies two numbers. It returns a new integer if the two numbers are integers. It returns a float if we pass an integer and float. If we pass in the integer two and a string, then the string will be repeated two times.
- Global variables are those variables that are defined outside of any function having a global scope, meaning they can be accessed anywhere after they are defined.

3.4 Exception handling

- The try…except statement will first attempt to execute the code in the "try" block, but if an error occurs, it will kick out and begin searching for the exception that matches the error. Once it finds the correct exception to handle the error, it will then execute that line of code.
- Adding an else statement will provide us a notification to the console that "The file was written successfully."
- By adding a finally statement, it will tell the program to close the file no matter the end result and print "File is now closed" to our console.

3.5 Objects and classes

- Python has many different kinds of data types: integers, floats, strings, lists, dictionaries, and Booleans.
- Every object has the following: a type, internal representation, and a set of functions called methods to interact with the data. An object is an instance of a particular type.
- Building the circle class in Python:
    - First, define the class, and then initialize each instance of the class with data attributes, radius, and color using the class constructor.
    - The function init is a constructor, which is a special function that tells Python you are making a new class.
    - There are other special functions in Python to make more complex classes. The radius and color parameters are used to initialize the radius and color data attributes of the class instance.
    - The self-parameter refers to the newly created instance of the class. The parameters, radius, and color can be used in the constructors' body to access the values passed to the class constructor when the class is constructed.

4. Working with Data in Python

4.1 Reading and writing files with open

- Python's built-in open function is used to create a file object and obtain the data from a "txt" file.
    - While using the open function, the first argument is the file path. This is made up of the file name and the file directory.
    - The second parameter is the mode. Common values used include 'r' for reading, 'w' for writing, and 'a' for appending.
    - Finally, we have the file object.
- It is recommended to use a "with" statement to open a file because it automatically closes the file. The code will run everything in the indent block, and then closes the file.
- Python's open function can be used to get a file object to create a text file. We can apply method write to write data to that file.
    - While using the open function, the first argument is the file path. This is made up of the file name. If you have that file in your directory, it will be overwritten.
    - The second parameter is the mode. Set the mode parameter to "w" for writing.
    - Finally, we have the file object.

4.2 Pandas

- Pandas is a popular library for data analysis.
- This library can be imported using the "import" command followed by the library name, which will give access to a large number of pre-built classes and functions.
- A data frame is comprised of rows and columns. A data frame can be created out of a dictionary, where the keys correspond to the column labels, and the values or lists correspond to the rows.

4.3 Numpy in Python

- Numpy is a library for scientific computing and has many useful functions. It serves as the basis for Pandas and has many other advantages like speed and memory.
- A Python list is a container that allows you to store and access data, where each element is associated with an index.
- A Numpy array or ND array is similar to a list, usually fixed in size, and has elements of the same type.
- The attribute ndim represents the number of array dimensions or the rank of the array.
- The attribute shape is a tuple of integers indicating the size of the array in each dimension.
- **Vector addition:** This can be performed over two given vectors to get a new vector. The first component of this new vector is the addition of the first component of the given two vectors. Similarly, the second component of the new vector is the sum of the second components of the given two vectors. The new vector is a linear combination of the given two vectors.
- **Vector multiplication with a scalar:** To multiply a vector with a scalar, multiply each component of a given vector by the given scalar.
- **Hadamard product** is another widely used operation in data science. The Hadamard product of u and v is a new vector z. The first component of z is the product of the first element of u and v. Similarly, the second component is the product of the second element of u and v. The resultant vector consists of the entrywise product of u and v.

Django, SQL and Databases

1.1 Introduction to databases

- Structured query language, or SQL, was designed to manage data in relational databases and is useful for handling structured data.
- Data is a collection of facts in words, numbers, and pictures.
- A database is a repository of data that provides functionality for adding, modifying, and querying data.
- Relational databases store tabular data as collections of related items, with columns containing item properties.
- The basic SQL statements are CREATE TABLE, INSERT, SELECT, UPDATE, and DELETE.
- Non-relational databases provide a flexible and scalable approach to storing and retrieving data.
- Relational databases are ideal for the optimized storage, retrieval, and processing of large volumes of data.
- Relational Database Management System (RDBMS) is a mature and well-documented technology, providing flexibility, reduced redundancy, ease of backup and disaster recovery, and ACID compliance.
- An entity-relationship model is a tool for designing relational databases. Entities become tables, and attributes are translated into columns.

1.2 Basic SQL statements

- SQL is used for querying and managing data.
- SQL is useful for handling structured data or data incorporating relations among entities and variables.
- The SQL SELECT statement retrieves data from a relational database table.
- The SELECT statement is a query, and the output we get from executing this query is a result set or a result table.
- In its simplest form, the syntax for a SELECT statement is: SELECT * from TableName
- The SQL INSERT statement inserts data into a relational database table by adding rows.
- The syntax of the INSERT statement is: INSERT INTO TableNameColumnName VALUES (values)
- For the INSERT statement, the values provided in the Values clause must equal the number of column names specified in the Column Name list. This ensures that each column has a value.
- The SQL UPDATE statement is used to read and modify data.
- The syntax of the UPDATE statement is as follows: UPDATE [TableName] SET [[ColumnName]=[Value]] <WHERE [Condition]>
- The SQL DELETE statement is used to remove data from a table.
- The syntax of the DELETE statement is: DELETE FROM [TableName] <Where [Condition]>
- The WHERE clause specifies the rows in a table that will be acted on by an SQL statement such as SELECT, DELETE, or UPDATE.

2. ORM: Bridging the Gap Between the Real World and Relational Model

- The object-oriented programming (OOP) and SQL paradigms model data differently.
- Object-relational mapping, or ORM, bridges the gap between OOP and SQL.
- ORM libraries or tools can map and transfer data stored in a relational database as rows into objects or objects into rows.
- ORM allows developers to use OOP to query and manipulate data because it transfers objects into rows and rows into objects.
- Django ORM is a Python ORM component that belongs to the Django web application framework.

- Django ORM can help speed up database development because you define maps to a database table for each Django model.
- Each Django field maps to a column type.
- Django automatically creates tables once models and fields are defined.
- Django APIs can perform Create, Read, Update, and Delete (CRUD) operations on database objects.
- In a Django model, you create an object and call the model's save method to insert it into the database as a record.
- To read objects using Django Model API, you need to construct a QuerySet using a Manager on your model class.
- There are several ways to update database records in Django by updating objects.
- To delete records in a database, you call Django ORM's Delete method on a model object or query set.

3. Full-stack Django Development

- The Model-View-Controller design pattern divides application logic into three components:
    - Model accesses and manipulates data
    - View presents data in various forms
    - Controller coordinates between Model and View
- The Django Model-View-Template pattern is like MVC, except there is no Controller, and the Django server performs the controller function.
- In Django, View is a Python function that takes a web request and applies the necessary logic to generate a web response.
- Django uses a template containing static HTML elements and special Python code to generate dynamic web pages.
- When you create a Django project, Django creates some core files.
    - manage.py is a command-line interface used to interact with the Django project
    - settings.py contains the settings and configurations for your Django project
    - urls.py contains the URL and routing definitions of your Django app
- You start building a Django admin site by creating an admin user.
- You can then log in as a superuser and register your models to the admin site so you can manage them.
- You can customize the admin form and add search and filters.
- A Django View takes a web request such as HTTP GET, POST, DELETE, or UPDATE and returns a web response. The web response can be a string, JSON/XML file, HTML page, or an error status indicating client or server-side errors.
- You create templates in Django to specify how your data will be presented. A Django template combines static HTML elements with Django template tags and variables to describe how the dynamic parts will be inserted. These work together to generate an HTML page rendered in a user's web browser.

4. Consolidate and Deploy Your Django App

- Both function-based and class-based views are Python functions.
- When you build a class-based view, you define a class subclassing the Django View base class. Then, you access some standard methods such as Get or Post. Next, you implement your logic to handle HTTP requests.
- To speed up development and solve common tasks, Django provides some built-in view classes called generic-based views for developers to reuse.
- Authentication is validating users' identities using credentials such as username and password.

- After users are authenticated, authorization will check the users' access permissions for resources such as databases.
- In Django, a user model is created to handle authentication and to work with other models, such as groups and permissions, to handle authorization.
- Developers can extend the user model to define application-specific users, such as instructors or learners inherited from the user model.
- Bootstrap, a free web front-end framework, facilitates web app development.
- Bootstrap provides many HTML and CSS templates to simplify Django template development.
- If you want to use Bootstrap CSS style classes without downloading Bootstrap, add a link to the latest Bootstrap version into the head element of your HTML template.
- To add static files to your apps, you first create folders for different static files, such as HTML templates, images, or CSS files.
- Under each folder to hold static files, you create a subfolder using the same app name. This creates namespacing to uniquely refer to static files that use the same name across multiple apps in a Django project.
- Django provides a set of STACFILES_FINDERS for locating the static files in an app. It also provides a staticfiles app to collect all static files in a single directory when an app is deployed.
- To deploy reliable, scalable, and maintainable Django apps, you need to deploy them on web servers.
- Since most web servers are not written in Python, Django apps need extra interfaces to talk to web servers.
- The Web Server Gateway Interface, or WSGI, is the main Python standard for communicating between web servers and applications.
- The Asynchronous Server Gateway Interface is another web server interface the Django app supports.
- Infrastructure as a service and platform as a service offering allows you to focus on your app development and deploy apps without worrying about the underlying infrastructure and platform.

Containers, Docker, OpenShift, Kubernetes

1.1 Understanding the benefits of containers

- A container is a unit of software that encapsulates everything needed to build, ship, and run applications.
- Containers lower deployment time and costs, improve utilization, automate processes, and support next-gen applications (microservices). Major container vendors include Docker, Podman, LXC, and Vagrant.
- Containers are small, fast, lightweight, and portable. Unlike virtual machines, they leverage the features and resources of the host operating system.
- Docker is an open platform used for developing, shipping, and running applications as containers.
- Docker containers are not a good fit for applications based on monolithic architecture or applications that require high performance or security.

- Docker architecture consists of the Docker client, the Docker host, and the container registry.
- The Docker host contains objects such as Dockerfiles, images, containers, networks, storage volumes, and other objects, such as plugins and add-ons.
- Docker uses networks to isolate container communications.
- Docker uses volumes and binds mounts to persist data even after a container stops running.
- Plugins, such as storage plugins, provide the ability to connect to external storage platforms.
- Some of the common Docker CLI commands include:
    - The **build** command is used for creating container images.
    - The **tag** command is used to name images.
    - The **images** command lists all images, their repositories and tags, and their sizes.
    - The **run** command is used to run a container.
    - The **push** and **pull** commands are used for storing images in a remote location and then retrieving those images.

2. Kubernetes Basics

2.1 Understanding Kubernetes architecture

- Container orchestration automates the container lifecycle, resulting in faster deployments, reduced errors, higher availability, and more robust security.
- Container orchestration tools have a wide variety of features and perform the following functions:
    - Aids in the provisioning and deployment of containers to make this a more automated, unified, and smooth process
    - Ensures that containers are redundant and available so that applications experience minimal downtime
    - Scales containers up and down to meet the demand and load-balance requests across instances so that no one instance is overwhelmed
    - Handles the scheduling of containers to the underlying infrastructure
    - Can perform health checks to ensure that applications are running and take necessary actions when checks fail
- Kubernetes is a highly portable, horizontally scalable, open-source container orchestration system with automated deployment and simplified management capabilities.
- Kubernetes architecture consists of a control plane and one or more worker planes.
    - A control plane includes controllers, an API server, a scheduler, and an **etcd**.
    - A worker plane includes nodes, a kubelet, container runtime, and kube-proxy.
- Kubernetes objects include Namespaces, Pods, ReplicaSets, Deployments, and Services.
    - Namespaces help in isolating groups of resources within a single cluster.
    - Pods represent a process or an instance of an app running in the cluster.
    - ReplicaSets create and manage horizontally scaled running Pods.
    - Deployments provide updates for Pods and ReplicaSets.
    - A service in Kubernetes is a REST object that provides policies for accessing the pods and cluster.
- Kubernetes capabilities include automated rollouts and rollbacks, storage orchestration, horizontal scaling, automated bin packing, secret and configuration management, Ipv4/Ipv6 dual-stack support, batch execution, self-healing, service discovery, load balancing, and extensible design.
- Services in Kubernetes are REST objects that provide policies for accessing the Pods and clusters.

    - ClusterIP provides Inter-service communication within the cluster.
    - NodePort Service, an extension of the ClusterIP service, creates and routes the incoming requests automatically to the ClusterIP Service.
    - The external load balancer (ELB) creates NodePort and ClusterIP Services automatically.
    - External Name service represents external storage as well as enables Pods from different namespaces to talk to each other.
- Ingress is an API object that provides routing rules to manage external users' access to multiple services in a Kubernetes cluster, whereas using a DaemonSet ensures that there is at least one copy of the Pod on all nodes; a StatefulSet manages stateful applications, manages Pod deployment and scaling, maintains a sticky identity for each Pod request and provides persistent storage volumes for your workloads and lastly a Job creates pods and tracks the Pod completion process; Jobs are retried until completed.

3. Managing Applications with Kubernetes

- A ReplicaSet enables scaling by creating or deleting pods.
- A ReplicaSet always tries to match the actual state to the desired state.
- Autoscaling enables scaling as needed at the cluster or node level and the Pod level.
- Autoscaler types include horizontal Pod (HPA), vertical pod (VPA), and cluster (CA).
- Rolling updates roll out app changes in a controlled and automated way.
- Rolling updates and rollback can be performed using all-at-once and one-at-a-time strategies.
- ConfigMaps are used to provide variables for your application.
- Secrets are used to provide sensitive information to your application.
- Binding an external Service to your deployment automatically provides the credentials to use the Service inside the code.
- Binding manages configuration and credentials for back-end Services while protecting sensitive data.

4. The Kubernetes Ecosystem: OpenShift, Istio

4.1 The Kubernetes ecosystem

- OpenShift® is an enterprise-ready Kubernetes container platform built for open hybrid cloud.
- OpenShift is easier to use, integrates with Jenkins, and has more services and features.
- Custom resource definitions (CRDs) extend the Kubernetes API.
- CRDs paired with custom controllers create new, declarative APIs in Kubernetes.
- Operators use CRDs and custom controllers to automate cluster tasks.
- A build is a process that transforms inputs into an object.
- An ImageStream is an abstraction for referencing container images in OpenShift.
- A service mesh provides traffic management to control traffic flow between services, security to encrypt traffic between services, and observability of service behavior to troubleshoot and optimize applications.
- Istio is a service mesh that supports four concepts: connection, security, enforcement, and observability. It is commonly used with microservices applications.
- Istio provides service communication metrics for these basic service monitoring needs: latency, traffic, errors, and saturation.

Microservices and Serverless App development

1. Introduction to Microservices

- Modern software development often delivers centrally hosted, web-based software as a service (SaaS) applications.
- The twelve-factor app methodology maps to the code, deploy, and operate stages of the software delivery lifecycle, enabling developers to create more efficient and more easily maintained SaaS applications.
- Microservices make each application component its own service, and each service communicates via an API.
- Microservices allow application components to use different technology stacks.
- Microservices enable individual components to scale in response to demand.
- Microservices lessen risks associated with change because components can iterate independently.
- Failures in one service do not necessarily impact other services. There are some anti-patterns to avoid when building microservices.
- Microservice patterns provide a structure that reduces reinventing solutions for commonly encountered challenges. The Backend for the Frontend pattern uses microservices to facilitate different user experiences more easily, while the Strangler pattern can help break up monolithic apps into microservices.

2. Web API Essentials: REST APIs and GraphQL

- REST APIs provide flexible but uniform interfaces between components.
- REST APIs are stateless and scalable.
- REST APIs communicate using HTTP methods POST, GET, PUT, and DELETE.
- REST is an architectural style defining how applications communicate.
- API Gateway is the door to your backend services while also enabling you to plug additional services while providing unified access.
- API Gateway makes it easier to scale or replace your backend services.
- Flask is a micro web framework to host Python web applications.
- cURL is used for transferring data with URLs and can be used at a command line or in scripts.
- Postman is a simple and popular tool for building, testing, and using APIs.
- Swagger helps you document and test your API.
- OpenAPI specification is a standard way of representing your APIs.

3. Serverless Overview

- Serverless computing removes the need for infrastructure management by end users and enables developers to focus on their applications' business-specific needs.
- Serverless uses BaaS services and FaaS platforms.
- Cloud providers take care of infrastructure management and maintenance tasks. The arrival of serverless computing means development teams can focus on writing high-quality code.
- Serverless offers several benefits, such as built-in high availability and fault tolerance, faster function run times, and pay-per-request billing.
- Serverless computing has several constraints, including unacceptable latency for time-critical apps, complex monitoring and debugging, and no state persistence.
- FaaS is a type of cloud-computing service that allows you to execute code in response to events without a complex infrastructure.
- FaaS is a subset of serverless computing that creates applications in the form of multiple functions and can be deployed on cloud, hybrid, or on-premises.

- A serverless stack comprises FaaS, BaaS, and an API Gateway.
- The Serverless Framework is a free and open-source web framework written using Node.js.
- The use cases for serverless web applications are Event streaming, Post-processing, and Multi-language.
- AWS Lambda provides you with an event-driven and pay-as-you-go serverless platform.
- Google Cloud Functions provides you with a simplified developer experience, along with Firebase for real-time data sync.
- Microsoft Azure promotes cloud and edge computing.
- IBM Cloud Functions gives you high availability and cost-effective computing.
- Knative promotes platform-agnostic alternatives.

4. Create and Deploy Microservices using Serverless

- Self-hosted microservices can be very complex and challenging.
- IBM Cloud Code Engine is a fully managed platform that handles all the hard deployment work, allowing developers to focus on their code.
- IBM Cloud Code Engine has three main use cases: 1) deploy applications, 2) build and deploy applications, and 3) run jobs.
- A project in Code Engine represents a group that contains and manages its resources and entities. A grouping in Code Engine contains entities such as build, app, job, and certificate for transport layer service (or TLS) HTTPS connections, and so on.
- An application runs your code to serve HTTP requests or to create WebSocket sessions.
- A build is a process to create a container image from your source code.
- A job runs one or more instances of your executable code.
- A container is a standalone executable software unit packaged with all its dependencies.
- Docker is a popular container-building and running platform.
- You can compose a Dockerfile to instruct the Docker platform to build a container image. After the container image is built, you can push it to the container registry and then pull it by its image name.
- You can create a Cloud Engine application either from a pushed container image or a source code repository. As per your preference, you can choose to use the IBM Cloud Console or IBM Cloud CLI to perform the application deployment tasks.

5. OpenShift Essentials/Working with OpenShift and Istio

- Red Hat OpenShift is a platform for running containerized workloads like microservices.
- OpenShift is like a Kubernetes distribution in that OpenShift with additional capabilities. OpenShift services help manage workloads, build cloud-native apps, and increase developer productivity. For example, OpenShift creates a Jenkins job to build microservices into containers automatically. In addition, OpenShift pushes the built containers to a registry and deploys those containers to the OpenShift cluster.
- Microservices architectures need security among services as well as ways to manage and test services.
- A service mesh is a dedicated layer that provides security and more by coordinating communication. Istio is a service mesh that provides traffic shifting, mutual transport layer security, and telemetry when deployed with microservices.
- Certified software fills development gaps in your application, eliminating the need for your organization to spend the time and money to develop new microservices. Red Hat Marketplace provides a central location to try, buy, deploy, and manage software certified for OpenShift environments.

CheatSheets & Glossaries

Welcome! This alphabetized glossary contains many of the terms in this course. This comprehensive glossary also includes additional industry-recognized terms not used in course videos. These terms are essential for you to recognize when working in the industry, participating in user groups, and in other professional certificate programs.

Estimated reading time: 10 minutes

Term	Definition
Access group	A group of users and service IDs is created so that the same access can be assigned to all entities within the group with one or more access policies.
ACL	Access Control Lists
Administrative users	Create, update, and delete application and service instances, and need insight into their team members' activities.
AI	Artificial intelligence
API keys	Unique identifiers are passed into an API to identify calling application or user.
API	Application Programming Interface
Application modernization	Helps organizations accelerate their digital transformation, take advantage of new technologies and services, and become more responsive to changing market dynamics
Application Performance Monitoring (APM)	Measures application availability and performance, providing tools needed to troubleshoot issues in an application's environment.
Application users	Users of the cloud-hosted applications.
AppSec	Application Security
Audit and compliance	A critical service within identity and access framework used to validate implemented controls against policies.
Authentication	Also known as identity service, it enables applications deployed to the cloud to authenticate users at an application level.
AZ	Availability Zones are distinct Data Centers with their own power, cooling and networking resources. These Zones can have names like DAL-09 or us-east-1.
Bare-metal hypervisor	Installed directly on top of the physical server. They're more secure, have lower latency, and are usually the ones you see in the market the most.
Block storage	Is presented to compute nodes using high-speed fiber connections, which means that read and write speeds are typically much faster and reliable than with file storage.
Blockchain	An immutable network allowing members to view only those transactions that are relevant to them.

Term	Definition
BPM	Business Process Management
Broad Network Access	Cloud computing resources can be accessed through the network.
BYOK	Bring Your Own Keys
CDNs	Content Delivery Networks is a distributed server network that delivers temporarily stored, or cached, copies of website content to users based on the user's geographic location.
Client-side encryption	Occurs before data is sent to cloud storage.
Cloud computing	A model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.
Cloud directory services	Used to securely manage user profiles and associated credentials inside a cloud environment.
Cloud encryption	Also known as the last line of defense, it encrypts data and provides robust data access control, key management, and certificate management.
Cloud monitoring solutions	Assess data, application, and infrastructure behaviors for performance, resource allocation, network availability, compliance, and security risks and threats.
Cloud native application	An application developed from the outset to work only in the cloud environment, or an existing app that has been refactored and reconfigured with cloud native principles
Cloud Region	A geographic area or location where a Cloud provider's infrastructure is clustered, and may have names like NA South or US East.
Cloud security	Policies, technological procedures, services, and solutions designed to secure the enterprise applications and data on the cloud against insider threats, data breaches, compliance issues, and organized security threats.
Composite multicloud	A variant of hybrid multicloud, distributes single applications across multiple providers, allowing you to move application components across cloud services and vendors as needed.
Containers	Are an executable unit of software in which application code is packaged, along with its libraries and dependencies, in common ways so that it can be run anywhere, whether it be on desktop, traditional IT, or the cloud.
Continuous delivery	Delivering small, well-designed, high-quality increments of software to customers
Continuous deployment	Progressing each new packaged build through the deployment lifecycle as rapidly as possible
Continuous integration	Creating packaged builds of the code changes released as immutable images

Term	Definition
Continuous monitoring	Monitoring with tools that help developers understand the performance and availability of their applications, even before they're deployed to production
CRM	Customer Relationship Management
Data center	A huge room or a warehouse containing cloud infrastructure.
Database monitoring tools	Help track processes, queries, and availability of services to ensure the accuracy and reliability of database management systems.
Decryption key	Defines how the encrypted data will be transformed back to legible data.
Dedicated hosts	Offer single-tenant isolation.
Delivery pipeline	An automated sequence of steps that involves the stages of Ideation, Coding, Building, Deploying, Managing, and Continuous Improvement
Developer users	Authorized to read sensitive information and to create, update, and delete applications.
DevOps	Collaborative approach where business owners and the development, operations, and quality assurance teams collaborate to continuously deliver software
Direct Attached storage	Or Local storage is storage which is presented directly to a cloud-based server and is effectively either within the host server chassis or within the same rack.
Encryption algorithm	Defines the rules by which data will be transformed so that it becomes illegible.
Encryption at rest	Protecting data while it is stored.
Encryption in transit	Protecting data while it is transmitted from one location to another.
Encryption in use	Protecting data when it is in use in memory.
Encryption	Scrambling data to make it illegible.
File storage	Is typically presented to compute nodes as 'NFS Storage'. NFS stands for Network File System and means that the storage is connected to compute nodes over a standard ethernet network.
GCP	Google Cloud Platform
HCM	Human Capital Management
Hosted hypervisor	There's a layer of host OS between the physical server and the hypervisor. These hypervisors are less frequently used and mostly used for "end-user" virtualization.
HPC	High-performance computing
Hybrid cloud	A computing environment that connects an organization's on-premises private cloud and third-party public cloud into a single, flexible infrastructure for running the organization's applications and workloads.

Term	Definition
Hybrid monocloud	A hybrid cloud with one cloud provider.
Hybrid multicloud	An open standards-based stack that can be deployed on any public cloud infrastructure
Hypervisor	A piece of software that runs above the physical server or host.
IaaS	Infrastructure as a service is a form of cloud computing that delivers fundamental compute, network, and storage resources to consumers on-demand, over the network, on a pay-as-you-go basis.
IDC	International Data Corporation
Identity and access management	Also known as access control, it helps authenticate and authorize users and provides user-specific access to cloud resources, services, and applications.
Infrastructure monitoring tools	Identify minor and large-scale hardware failures and security gaps so that developers and administrators can take corrective action before problems affect user experience.
IOPS	Input/Output Operations Per Second and refers to the speed at which the disks can write and read data.
IoT	Internet of things
Key management services	Help perform life cycle management for encryption keys that are used in cloud services or customer-build ap.ps
KYOK	Keep Your Own Keys
MDM	Master Data Management
Measured Service	You only pay for what you use or reserve as you go.
Microservices architecture	Approach in which a single application is composed of many loosely coupled and independently deployable, smaller components or services
Microservices	Break down large applications into their core functions
Monolithic architecture	Approach in which a single application is built out of one piece of software
Multifactor authentication	Adds an additional layer or authentication for application users.
NFS	Network File Storage
NIST	National Institute for Standards and Technology
Object storage	Storage not attached to a compute node, rather it is accessed via an API.
PaaS	Platform as a service

Term	Definition
Pay-As-You-Go	Users can order cloud resources from a larger pool of available resources and pay for them on a per-use basis.
POP	Post office protocol
Private Cloud	Cloud infrastructure provisioned for exclusive use by a single organization comprising multiple consumers, such as the business units within the organization.
Public Cloud	Users get access to servers, storage, network, security, and applications as services delivered by cloud service providers over the internet.
Rapid elasticity	You can increase or decrease resources as per your demand because of the elastic property of the cloud.
Reporting	Provides a user-centric view of access to resources.
Reserved virtual server	Instances allow you to reserve capacity and guarantee resources for future deployments.
SaaS	Software as a service
SDN	Software Defined Networking
Server-side encryption	Occurs after cloud storage receives your data but before the data is written to disk and stored.
Serverless	Approach to computing that offloads responsibility for common infrastructure management tasks
Service discovery	Creates a roadmap for microservices to communicate
Shared or Public Cloud VMs	Are provider-managed, multi-tenant deployments that can be provisioned on-demand with predefined sizes.
SIP	SaaS integration platforms
SSL	Secure Sockets Layer
TCO	Total cost for ownership
TLS	Transport Layer Security
Transient or Spot VMs	Take advantage of unused capacity in a cloud data center.
User and service access management capability	Enables cloud application and service owners to provision and de-provision user profiles with minimal human interaction.
Utility model of billing	You are charged after the usage and at the end of the pre-defined period.
Virtualization	Process of creating a software-based or virtual version of something whether that be compute, storage, networking, servers, or applications.

Term	Definition
VLANs	Virtual Local Area Networks
VM	Virtual machines are software-based computers, based on virtualization technologies.
VPC	Virtual Private Cloud
VPN	Virtual Private Networks

Glossary: Introduction to Web Development with HTML, CSS, JavaScript

Welcome! This alphabetized glossary contains many of the terms in this course. This comprehensive glossary also includes additional industry-recognized terms not used in course videos. These terms are essential for you to recognize when working in the industry, participating in user groups, and in other professional certificate programs.

Estimated reading time: 10 minutes

Term	Definition
AJAX	"Asynchronous JavaScript and XML" that encompasses more than asynchronous server calls through JavaScript and XML. It is not programing language or technology but rather a programming concept. Ajax represents a series of techniques that provide richer, interactive web applications through HTML, JavaScript, Cascading style sheets, and modifying the web page through the Document Object Model. The name is misleading though because nowadays, JSON is commonly used instead of XML.
AngularJS	An open-source JavaScript framework for dynamic web applications.
Anonymous Functions	A type of function that has no name or we can say which is without any name. It is declared without any identifier and is often used as a parameter of another function. It is a common way to execute a function immediately after its declaration.
Application Programming Interface (API)	Code that allows two software programs to communicate with each other.
Array	A data structure that aids the programmer in the storage and retrieval of data by indexed keys. Arrays use a zero-based

Term	Definition
	indexing scheme, meaning that the first element of an array has an index of zero. Arrays grow or shrink dynamically by adding or removing elements.
Build Automation Servers	Execute build-automation utilities on a scheduled or triggered basis.
Build Automation Utilities	Generate executables by compiling and linking code.
Build Automation	Allows you to download dependencies, compile code, package binary code, run tests, deploy to production.
Classes	Classes act as a blueprint or template for building objects with similar characteristics and behaviours. A class encapsulates data (in the form of properties) and functions (in the form of methods) that work on that data.

Client-Side Script	A program that accompanies an HTML doc or embedded in HTML. Scripts run during load of a document or when an action is performed. They can be used to validate forms, process input, or dynamically create document elements. Embed a script in HTML, with the `<script>` tag in either of the following ways:
	`<script> // JS code </script>`
	`<script src="path name"></script>`
	Use `<noscript>` tag for browsers with JavaScript disabled or ones that don't support JavaScript.

Component Framework	Component frameworks provide pre-styled components and templates which are easy to add to any website.
Continuous Integration/Continuous Deployment (CI/CD)	A method for releasing code and integrating it into code that has already been developed in order to prevent the application from breaking throughout the app's lifecycle.
CSS	"Cascading Style Sheets" is a style sheet language that describes how HTML elements are displayed. It is the design that is layered over the top of an HTML web page.
Django	A framework for Python web development.

Term	Definition
Document Objects	Document representing the main web page that gives access to all HTML elements on the page. When page is loaded HTML doc becomes a document. It is referred to with "document".
DOM	"Document Object Model" is a programming interface (API) between HTML and JavaScript. It allows for dynamically accessing and updating content, structure, and style. JavaScript uses the DOM to access and modify web page elements in the browser.
Dynamic Content	Data that is created each time a request is sent to a server.
Element Nodes	All HTML tags.
Element Objects	The most general base class that all element objects in a Document inherit. It only has methods and properties common to all elements. Everything in a HTML page is an element. And one element can have other elements nested within itself.
Endpoint	The point at which an API connects with the software program.
Event Binding	Refers to telling the browser that a function should be called whenever some 'event' occurs.

Event Handlers	A function that declares what to do when an action is performed such as the click of a button. Example:
	1. 1
	1. <button type="button" onclick="showAnswers()">
	Copied!
	Show Solution
	1. 1
	1. <script>
	function showAnswers()
	{
	//code
	alert(""A"") }
	</script>
	</button>
	Copied!

Term	Definition
	Note that `showAnswers()` is an event handler.
Event	An event is something either a browser or a user does that the JavaScript can react to such as a button click or when a user submits input on a form.
Extend	This keyword is used in class declarations or class expressions to create a class that is a child of another class.
Fixed Layout	A fixed layout is a layout where you specify the height and width of elements, and those values remain the same regardless of which operating system or browser you use to access the website.
Fluid Layout	A fluid layout is a layout in which the height and width of elements is flexible and can expand or contract based on the browser window, the operating system, and other user preferences.
Frameworks	Provide a standard way to build an application. Frameworks dictate architecture and program flow.

Functions	Functions are modules of code that execute a particular task. They may take-in data, called arguments or parameters, and sometimes return data as well, called the return value. Functions are commonly defined with this syntax:
	1. 1
	1. function functionName() {
	// function code;
	// optional return statement;
	}
	Copied!

IDE	"Integrated Development Environment" helps create and manage code.
IIFE	"Immediately Invoked Function Expression" runs immediately after it is defined. After the function executes it cannot be called again elsewhere in the program. It is a type of self-executing function.
Inversion of Control	A predefined workflow where the developer is not in full control of how the application operates.

Term	Definition
JavaScript Framework	An application framework written in JavaScript to create responsive sites.
LESS	"Learner Style Sheets" add more style and functions to CSS.
less.js	A JavaScript tool that converts LESS styles to CSS.
Libraries	They are reusable collections of code.
Nodes	The basis of all elements in the Document Object Model (DOM) structure.
Objects	Objects are instances created from a class. They are real-world entities that represent the characteristics defined by the class. Objects have a special set of properties that store data and methods that specify behaviours. These methods and properties can be accessed and changed to carry out specific tasks and communicate with other programs.
Opinionated	Frameworks that have a lot of control are sometimes considered "opinionated".
Package Managers	Coordinate with file archivers to extract packages. Verify check sums and digital certificates. Locate, download, and install updates of existing software from a repository as well as manage dependencies. Common package managers include the following: Debian Package Management System (DPMS), Red Hat Package Manager for Linux, Chocolatery for Windows, Homebrew and MacPorts for MacOS.
Packages	Archive files that include app files, instructions for installation, and metadata.
Prototypes	A function prototype lets you easily define and add properties or methods to an object. Prototypes exist for all objects that can be created with the keyword "new". All object constructors create objects that inherit properties and methods that are defined by the prototype. At instantiation objects inherit the current state of the prototype. Note however, that scripts can override prototype properties and functions. Following is an example of using a prototype to add a method to the Car class: 1. 1

Term	Definition

	1. function Car(make, model, year) {
	this.make = make;
	this.model = model;
	this.year = year;
	}
	Car.prototype.getName = function() {
	return this.make + ' ' + this.model + ' ' + this.year;
	}
	Copied!

React.js	A JavaScript framework developed by Facebook that helps build and drop elements onto a page.
Responsive Design	Design technique that automatically resizes a display to adapt to a specific screen size.
Route	Allows front-end client to plug into correct socket on the backend. They are the paths that network traffic takes from a virtual machine (VM) instance to other destinations.
SASS	"Syntactically Awesome Stylesheets" are an extension of CSS.
Script	Offers developers means to modify and extend HTML documents in highly interactive ways. Scripts can be used to validate forms or to process input as it is typed. Scripts can be triggered by events that occur on a web page, such as the clicking of a button. Scripts can be used to dynamically create document elements on an HTML page.
Self-Executing Functions	Often used to initialize data or declare DOM elements. These functions can be anonymous.
Static Content	A display of data that has been previously stored on a server.
Text Nodes	The nodes that contain actual text that go between an element start tag and end tag.
this	Keyword "this" refers to current instance of the object. The value of "this" can vary depending on how the object is called.

Term	Definition
Utility Framework	The utility framework provides utility classes that are scoped to individual CSS properties, which helps in building custom designs in HTML files.
Version Control	Allows you to revert to earlier versions of code, resolves conflicts between the same files, and split and merge different code branches.
Vue.js	A community-based JavaScript framework focused on UI. Includes UI components such as buttons and other visual elements, and is both a library and a frame.
Web Storage APIs	APIs that allow data storage in a browser.
XHTML	An "eXtensible Hypertext Markup Language" similar to HTML but with stricter formatting rules.
XML	An "eXtensible Markup Language" Designed to store and transport data allowing users to define their own markup languages, primarily to display documents on the web.

Cheat Sheet: Introduction to Web Development with HTML, CSS, JavaScript

Estimated reading time: 20 minutes

HTML		
Element	**Description**	**Example**
<!-- -->	This tag denotes a comment in HTML, which is not displayed by a browser but can be useful to hide and document code.	1. 1 2. 2 1. <!-- This is a comment 2. --> Copied!
<!DOCTYPE html>	All HTML documents must start with this declaration. It tells the browser	1. 1 2. 2 3. 3

HTML		
	what document type to expect. Note that this element has no ending tag.	4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 1. <!DOCTYPE html> 2. <html> 3. <head> 4. <!-- Metadata Here --> 5. </head> 6. <body> 7. <!-- Document Body Here --> 8. </body> 9. </html> Copied!
link name	This tag, called an "anchor tag" creates hyperlinks using the href attribute. In place of path enter the URL or path name to the page you want to link to.	1. 1 2. 2 1. <a 2. href="https://www.ibm.com">IBM Copied!
<body>	Contains the contents of the HTML document. It should contain all other tags besides the <head> element to display the body of the document.	1. 1 2. 2 3. 3 4. 4 5. 5 6. 6

HTML		
		7. 7 8. 8 9. 9 1. <!DOCTYPE html> 2. <html> 3. <head> 4. <!-- Metadata Here --> 5. </head> 6. <body> 7. <!-- Document Body Here --> 8. </body> 9. </html> Copied!
	Often used to separate sections in the body of a document in order to style that content with CSS.	1. 1 2. 2 3. 3 1. 2. This element has no particular semantic meaning but is often used in conjunction with CSS for styling purposes. 3. Copied!
<h1>	Adds a level 1 heading to the HTML document.	1. 1 1. Thomas J. Watson Copied!
<head>	Contains metadata and should be placed after the <html> tag and before the <body> tag.	1. 1 2. 2 3. 3 4. 4

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td></td><td></td><td>5. 5
6. 6
7. 7
8. 8
9. 9
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <!-- Metadata Here -->
5. </head>
6. <body>
7. <!-- Document Body Here -->
8. </body>
9. </html>
Copied!</td></tr>
<tr><td><html></td><td>The root element of an HTML document. All other tags in the document should be contained in this tag.</td><td>1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <!-- Metadata Here -->
5. </head></td></tr>
</table>

HTML		
		6. <body> 7. <!-- Document Body Here --> 8. </body> 9. </html> Copied!
<img src="path" width="dim1" height="dim2">	This tag is used to place an img. In place of path insert a URL or a relative file path to the image location. Other optional attributes include width and height of the image in pixels.	1. 1 2. 2 1. Copied!
	Element that creates bulleted line items in an ordered or unordered list. Should be used in conjunction with the or tags.	1. 1 2. 2 3. 3 4. 4 1. 2. <li1>Bullet point 1<l1> 3. <l1>Bullet point 2<l1> 4. Copied!
<link>	Used to link an external document, such as a CSS file, to an HTML document.	1. 1 2. 2 3. 3 1. <head> 2. <link rel="stylesheet" href="styles.css"> 3. </head> Copied!

HTML		
<meta>	Used to provide metadata about the HTML document.	1. 1 2. 2 3. 3 1. <head> 2. <meta name="author" content="Christopher Moore"> 3. </head> Copied!
	Element that creates an ordered list using numbers. Should be used in conjunction with the tag.	1. 1 2. 2 3. 3 4. 4 1. 2. Numbered bullet point 1 3. Numbered bullet point 2 4. Copied!
	This tag is used to identify a paragraph. It places a line break after the text it is enclosed in.	1. 1 1. This is a paragraph of text. It can be as short or as long as needed. Copied!
<script>	Used to embed JavaScript in an HTML document.	1. 1 2. 2 3. 3 1. <script> 2. alert("Hello World"); 3. </script> Copied!

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td><table></td><td>This tag is used to denote a table. Should be used with <tr> (defines a table row) and <td> (defines a table cell within a row) tags. The <th> tag can also be used to define the table header row.</td><td>

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

```
<table>
<tr>
<th>Header cell 1</th>
<th>Header cell 2</th>
</tr>
<tr>
<td>First row first cell</td>
<td>First row second cell</td>
</tr>
<tr>
<td>Second row first cell</td>
<td>Second row second cell</td>
</tr>
```

</td></tr>
</table>

HTML		
		14. </table> 15. 16. </td> Copied!
<td>	Denotes a cell within a row, within a table.	1. 1 1. <td>Cell Content</td> Copied!
<th>	Denotes the header cells within a row within a table.	1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 10. 10 11. 11 12. 12 13. 13 14. 14 1. <table> 2. <tr> 3. <th>Header cell 1</th> 4. <th>Header cell 2</th> 5. </tr> 6. <tr> 7. <td>First row first cell</td> 8. <td>First row second cell</td>

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td></td><td></td><td>

```
</tr>
<tr>
<td>Second row first cell</td>
<td>Second row second cell</td>
</tr>
</table>
```

Copied!
</td></tr>
<tr><td><title></td><td>Defines the title of the HTML document displayed in the browser's title bar and tabs. It is required in all HTML documents. It should be contained in the <head> tag.</td><td>

```
1
2
3
4
5
6
7
8
9
```

```
<!DOCTYPE html>
<html>
<head>
<title>Document Title</title>
</head>
<body>
<!-- Document Body Here -->
</body>
</html>
```

Copied!
</td></tr>
<tr><td><tr></td><td>Denotes a row within a table.</td><td>

```
1
2
3
```

</td></tr>
</table>

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td></td><td></td><td>4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
<table>
<tr>
<th>Header cell 1</th>
<th>Header cell 2</th>
</tr>
<tr>
<td>First row first cell</td>
<td>First row second cell</td>
</tr>
<tr>
<td>Second row first cell</td>
<td>Second row second cell</td>
</tr>
</table>
Copied!</td></tr>
<tr><td></td><td>Element that creates an unordered list using bullets. Should be used</td><td>1. 1
2. 2
3. 3</td></tr>
</table>

HTML		
	in conjunction with thetag.	4. 4 1. 2. Bullet point 1 3. Bullet point 2 4. Copied!
CSS		
Element	**Description**	**Example**
OCTYPE html>	All HTML documents must start with this declaration. It tells the browser what document type to expect. Note that this element has no ending tag.	1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 1. <!DOCTYPE html> 2. <html> 3. <head> 4. <title>Document Title</title> 5. </head> 6. <body> 7. Document body here 8. </body> 9. </html> Copied!
	This tag, called an "anchor tag"	1. 1

HTML		
	creates hyperlinks using the href attribute. In place of path enter the URL or path name to the page you want to link to.	2. 2 1. <a 2. href="https://www.ibm.com">IBM Copied!
<article>	Identifies a self-contained piece of content that could be distributed to other websites and platforms as a stand- alone unit. Similar to tag in that it does not render as anything special in the browser unless it is styled with CSS.	1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 1. <article 2. class="CSS-Style-Reference"> 3. HTML 4. HTML stands for Hypertext Markup Language. 5. It is a standardized system for tagging text files to display specific 6. font, color, graphics, and hyperlinks on World Wide Web pages. 7. </article> Copied!
<aside>	Defines some content aside from the content it is placed in. Similar to tag in that it does not render as anything special in the	1. 1 2. 2 3. 3 4. 4 1. <aside>

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td></td><td>browser unless it is styled with CSS.</td><td>2. Use the aside tag to represent a section of a page that consists
3. of content tangentially related to the content.
4. </aside>
Copied!</td></tr>
<tr><td><body></td><td>Contains the contents of the HTML document. It should contain all other tags besides the <head> element to display the body of the document.</td><td>1 2 3 4 5 6 7 8 9
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title>Document Title</title>
5. </head>
6. <body>
7. Document body here
8. </body>
9. </html>
Copied!</td></tr>
<tr><td><div></td><td>Often used to separate sections in the body of a document in order to style that content with CSS.</td><td>1 2 3
1. <div></td></tr>
</table>

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td></td><td></td><td>

```
2. This element has no particular semantic meaning but is often used in conjunction with CSS for styling purposes.
3. </div>
```

Copied!
</td></tr>
<tr><td><fieldset></td><td>Groups related elements in a form and puts a box around them.</td><td>

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
1. <form action= "/script.php">
2. <fieldset>
3. <legend>User:</legend>
4. <label for="fname">First name:</label>
5. <input type="text" id="fname"
6. name="fname"><br><br>
7. <label for="lname">Last name:</label>
8. <input type="text" id="lname"
9. name="lname"><br><br>
10. <input type="submit" value="Submit">
11. </fieldset>
12. </form>
```

</td></tr>
</table>

HTML		
		Copied!
<figcaption>	Used in conjunction with the <figure> tag to mark up an image.	1 2 3 4 1. <figure> 2. <img src="durham.jpg"> 3. <figcaption>Fig.1 - Durham, NC</figcaption> 4. </figure> Copied!
<figure>	Used to mark up an image in conjunction with the <figcaption> tag.	1 2 3 4 1. <figure> 2. <img src="durham.jpg"> 3. <figcaption>Fig.1 - Durham, NC</figcaption> 4. </figure> Copied!
<footer>	Contains a footer of a document and often contains information such as authoring, copyright info, contact info, sitemap, and related documents.	1 2 3 4 1. <footer> 2. Author: Christopher Moore 3. </footer> 4. <form action="/script.php"> Copied!

HTML		
<form>	Creates an HTML form for user input.	1 2 3 4 5 6 7 8 9 10 11 12 `<form action="/script.php">` `<fieldset>` `<legend>User:</legend>` `<label for= "fname">First name:</label>` `<input type="text" id="fname"` `name="fname">  ` `<label for= "lname">Last name:</label>` `<input type="text" id="lname"` `name="lname">  ` `<input type="submit" value="Submit">` `</fieldset>` `</form>` Copied!

<h1>	Adds a level 1 heading to the HTML document.	1 2
		`<h1>Thomas J.`
		`Watson</h1>`

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td></td><td></td><td>Copied!</td></tr>
<tr><td><head></td><td>Contains metadata and should be placed after the <html> tag and before the <body> tag.</td><td>1 2 3 4 5 6 7 8 9
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title>Document Title</title>
5. </head>
6. <body>
7. Document body here
8. </body>
9. </html>
Copied!</td></tr>
<tr><td><header></td><td>A container for introductory content such as heading elements, logo, or authoring information.</td><td>1 2 3 4 5
1. <header>
2. <h1>Header tag example</h1></td></tr>
</table>

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td></td><td></td><td>

```
3. <p>It works as a container for introductory
   content such as heading elements, logo, or
   authoring information.</p>
4. <p>Author: Christopher Moore</p>
5. </header>
```

Copied!
</td></tr>
<tr><td><html></td><td>The root element of an HTML document. All other tags in the document should be contained in this tag.</td><td>

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
```

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title>Document Title</title>
5. </head>
6. <body>
7. Document body here
8. </body>
9. </html>
```

Copied!
</td></tr>
<tr><td><img src="path" width="dim1" height="dim2"></td><td>This tag is used to place an img. In place of path insert a URL or a relative file path to the image location. Other</td><td>

```
1. 1
2. 2
3. 3
4. 4
```
</td></tr>
</table>

HTML		
	optional attributes include width and height of the image in pixels.	 1. <img 2. src="https://upload.wikimedia.org/wikipedia/commons/7/7e/Thomas_J_Watson_Sr.jpg" 3. width="300" 4. height="300"> Copied!
<input type="?">	Specifies an input field on a form with the type attribute. Common input types include: "color," "date," "datetime-local," "email," "number," "range," "search," "url," "tel," "datalist," "select," "text," "option," and "placeholder."	 1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 10. 10 11. 11 12. 12 1. <form action="/script.php"> 2. <fieldset> 3. <legend>User:</legend> 4. <label for="fname">First name:</label> 5. <input type="text" id="fname" 6. name="fname"> 7. <label for="lname">Last name:</label> 8. <input type="text" id="lname" 9. name="lname"> 10. <input type="submit" value="Submit"> 11. </fieldset>

HTML		
		12. </form> Copied!
	Element that creates bulleted line items in an ordered or unordered list. Should be used in conjunction with the or tags.	1. 1 2. 2 3. 3 4. 4 1. 2. - Bullet point 1 3. - Bullet point 2 4. Copied!
<link>	Used to link a CSS document to an HTML document.	1. 1 2. 2 3. 3 1. <head> 2. <link rel="stylesheet" href="styles.css"> 3. </head> Copied!
<meta>	Used to provide metadata about the HTML document.	1. 1 2. 2 3. 3 1. <head> 2. <meta name="author" content="Christopher Moore"> 3. </head> Copied!
<nav>	Used to define a set of navigational elements.	1. 1 2. 2 3. 3

HTML		
		4. 4 5. 5 1. <nav> 2. Home 3. Page 1 4. Page 2 5. </nav> Copied!
	Element that creates an ordered list using numbers. Should be used in conjunction with the tag.	1. 1 2. 2 3. 3 4. 4 1. 2. - Numbered bullet point 1 3. - Numbered bullet point 2 4. Copied!
	This tag is used to identify a paragraph. It places a line break after the text it is enclosed in.	1. 1 2. 2 3. 3 4. 4 1. Thomas J. Watson, Sr. is the American 2. industrialist, who built the International Business 3. Machines Corporation (IBM) into the largest 4. manufacturer of electric typewriters and data processing equipment in the world. Copied!
<script>	Used to embed Javascript in an HTML document.	1. 1 2. 2

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td></td><td></td><td>3. 3

1. <script>
2. alert("Hello World");
3. </script>
Copied!</td></tr>
<tr><td><section></td><td>Defines an important section of a document. Can be used within headers and footers as well.</td><td>1. 1
2. 2
3. 3
4. 4
5. 5

1. <section>
2. <h2>Introduction</h2>
3. <p>This document provides some examples of how to
4. use a variety of HTML elements</p>
5. </section>
Copied!</td></tr>
<tr><td><style></td><td>Used to apply simple CSS to an HTML document.</td><td>1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9

1. <head>
2. <style>
3. p {color:red}</td></tr>
</table>

HTML		
		4. </style> 5. </head> 6. <body> 7. This paragraph will be red because I've styled 8. the paragraph tag with CSS. 9. </body> Copied!
<table>	This tag is used to denote a table. Should be used with <tr> (defines a table row) and <td> (defines a table cell within a row) tags. The <th> tag can also be used to define the table header row.	1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 10. 10 11. 11 12. 12 13. 13 1. <table> 2. <tr> 3. <th>Header cell 1</th> 4. <th>Header cell 2</th></tr> 5. <tr> 6. <td>First row first cell</td> 7. <td>First row second cell</td> 8. </tr>

HTML		
		9. <tr> 10. <td>Second row first cell</td> 11. <td>Second row second cell</td> 12. </tr> 13. </table> Copied!
<td>	Denotes a cell within a row, within a table.	1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 10. 10 11. 11 12. 12 13. 13 14. 14 1. <table> 2. <tr> 3. <th>Header cell 1</th> 4. <th>Header cell 2</th> 5. </tr> 6. <tr> 7. <td>First row first cell</td> 8. <td>First row second cell</td> 9. </tr>

HTML		
		10. <tr> 11. <td>Second row first cell</td> 12. <td>Second row second cell</td> 13. </tr> 14. </table> Copied!
<th>	Denotes the header cells within a row within a table.	1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 10. 10 11. 11 12. 12 13. 13 14. 14 1. <table> 2. <tr> 3. <th>Header cell 1</th> 4. <th>Header cell 2</th> 5. </tr> 6. <tr> 7. <td>First row first cell</td> 8. <td>First row second cell</td> 9. </tr>

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td></td><td></td><td>

```
<tr>
<td>Second row first cell</td>
<td>Second row second cell</td>
</tr>
</table>
```

Copied!
</td></tr>
<tr><td><title></td><td>Defines the title of the HTML document displayed in the browser's title bar and tabs. It is required in all HTML documents. It should be contained in the <head> tag.</td><td>

```
1
2
3
4
5
6
7
8
9
<!DOCTYPE html>
<html>
<head>
<title>Document Title</title>
</head>
<body>
Document body here
</body>
</html>
```

Copied!
</td></tr>
<tr><td><tr></td><td>Denotes a row within a table.</td><td>

```
1
2
3
4
```

</td></tr>
</table>

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td></td><td></td><td>5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
<table>
<tr>
<th>Header cell 1</th>
<th>Header cell 2</th>
</tr>
<tr>
<td>First row first cell</td>
<td>First row second cell</td>
</tr>
<tr>
<td>Second row first cell</td>
<td>Second row second cell</td>
</tr>
</table>
Copied!</td></tr>
<tr><td></td><td>Element that creates an unordered list using bullets. Should be used in conjunction</td><td>1. 1
2. 2
3. 3
4. 4</td></tr>
</table>

HTML		
	with the tag.	1. 2. - Bullet point 1 3. - Bullet point 2 4. Copied!
JAVASCRIPT		
Element	**Description**	**Example**
Child()	An HTML DOM method that after creating an element, you can use this function to place the element in the appropriate location within the document. The element to append is the only parameter.	1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 10. 10 11. 11 12. 12 13. 13 1. //Creates the element <p> and text "Hello World". Appends Hello World <p> to the HTML document. 2. <head> 3. <script> 4. function addPara() { 5. var newPara = document.createElement("p");

HTML		
		6. var newText = document.createTextNode("Hello World!"); 7. newPara.appendChild(newText); 8. document.body.appendChild(newPara); 9. } 10. </script> 11. </head> 12. <body onload="addPara()"> 13. </body> Copied!
Arrays	Created by declaring the array elements in [ ]. An array can be assigned to a variable, usually using the keyword const or var. Arrays use zero based indexing to access their elements.	1. 1 2. 2 1. const Beatles = ["Ringo", "Paul", "George", "John"]; 2. //Here Beatles[0] is "Ringo". Copied!
Date()	Constructor is new Date([optional parameters]). If the constructor is declared with no parameters, it returns current local date and time. New dates can be created by passing parameters to new Date function.	1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 1. //create a new date from a string 2. var newDate = new Date("2021-1-17 13:15:30"); 3. 4. //create a new date instance representing 17 Jan 2021 00:00:00

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td></td><td></td><td>

```
//note that the month number is zero-based
var newDate = new Date(2021, 0, 17);
```

Copied!
</td></tr>
<tr><td>document.createElement()</td><td>Takes one tag name parameter and creates an element with that name. Can place the element elsewhere on the page using functions like insertBefore(), appendChild(), replaceChild().</td><td>

```
1
2
3
4
5
6
7
8
9
10
11
12
13
//Creates the element <p> and text "Hello World". Appends Hello World <p> to the HTML document.
<head>
<script>
function addPara() {
var newPara = document.createElement("p");
var newText = document.createTextNode("Hello World!");
newPara.appendChild(newText);
document.body.appendChild(newPara);
}
</script>
```
</td></tr>
</table>

HTML		
		11. </head> 12. <body onload="addPara()"> 13. </body> Copied!
document.createTextNode()	Takes a string as input text and returns a text node with the input text.	1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 10. 10 11. 11 12. 12 13. 13 1. //Creates the element <p> and text "Hello World". Appends Hello World <p> to the HTML document. 2. <head> 3. <script> 4. function addPara() { 5. var newPara = document.createElement("p"); 6. var newText = document.createTextNode("Hello World!"); 7. newPara.appendChild(newText); 8. document.body.appendChild(newPara); 9. }

HTML		
		10. </script> 11. </head> 12. <body onload="addPara()"> 13. </body> Copied!
document.getElementByID()	A method of the DOM that takes an ID value parameter and returns an element that matches the id.	1. 1 2. 2 3. 3 4. 4 5. 5 6. 6 7. 7 8. 8 9. 9 10. 10 1. //Changes the content of the div to "Hello World!" 2. 3. Hello 4. Hello 5. 6. 7. <script> 8. document.getElementById("div1").innerHTML = " Hello 9. World! "; 10. </script> Copied!
document.getElementsByTagName()	A method of the DOM that takes a tag name parameter and	1. 1 2. 2

HTML		
	returns an array called "NodeList" that contains elements with the specified tag name.	1. //Gets an array of all elements in a document with the <p> tag. 2. var tagNameArray = document.getElementsByTagName("p"); Copied!
document.write()	Writes HTML or JavaScript to a document. Note that it overwrites any other text in the document so is mostly used for testing purposes only.	1. 1 2. 2 1. //Writes "Hello World" to the output stream. 2. document.write("Hello World"); Copied!
element.getAttribute()	Returns the value of the specified attribute. Takes one parameter: the attribute name whose value is to be returned.	1. 1 2. 2 3. 3 4. 4 5. 5 1. //Removes the CSS style color blue 2. 3. <script> 4. var div1 = document.getelementById("div1").getAttribute("style"); 5. </script> Copied!
element.innerHTML()	A property of the Element class that returns or alters contents of an HTML element as a text string.	1. 1 2. 2 3. 3 4. 4 5. 5

HTML		
		6. 6 7. 7 8. 8 9. 9 10. 10 11. 11 1. //Changes the content of the div to "Hello 2. World!" 3. 4. Hello 5. Hello 6. 7. 8. <script> 9. document.getElementById("div1").innerHTML = " Hello 10. World! "; 11. </script> Copied!
element.removeAttribute()	A property of the Element class that removes all previously set inline CSS styles for a particular element. Takes one parameter: the attribute name that is being removed.	1. 1 2. 2 3. 3 4. 4 5. 5 1. //Removes the CSS style color blue 2. 3. <script>

HTML		
		var div1 = document.getelementById("div1").getAttribute("style"); </script> Copied!

element.setAttribute()	A property of the Element class that overwrites all previously set inline CSS styles for a particular element. Takes two parameters: the attribute name that is being set and the attribute value the attribute is set to.	1 2
		//In all elements named "theImage" sets the name of all src attributes to "another.gif"
		document.getElementById("theImage").setAttribute("src", "another.gif");
		Copied!

element.style()	A property of the Element class that returns or alters inline CSS. Syntax is element.style.propertyName = value	1 2 3 4 5 6
		//Changes the CSS style color from blue to red
		
		<script>
		var div1 = document.getelementById("div1");
		div1.style.color = "red";
		</script>
		Copied!

Error Objects	Instance creates two properties about the error:	1 2

HTML		
	message that contains description of the error and the name property identifies the type of error. Generic error plus 6 other core errors: TypeError, RangeError, URIError, EvalError, ReferenceError, SyntaxError. Error object can be extended to create custom error messages using the throw keyword.	3. 3 4. 4 5. 5 6. 6 1. //Catch statement defines a block of code to be executed if an error occurs in the try block. 2. catch (err) { 3. document.getElementById("myfile").innerHTML = err.name; 4. } 5. //Creates custom error message 6. throw new Error("Only values 1-10 are permitted"); Copied!
History Objects	The history object is part of the window object and contains the URLs visited by the user within a browser window. It exposes useful methods and properties that let you navigate back and forth through the user's history and manipulate the contents of the history stack.	1. 1 2. 2 1. //Go back two pages if the history exists in the history list. 2. history.go(-2); Copied!
insertBefore()	An HTML DOM method that, after creating an element, places a child element in the	1. 1 2. 2 3. 3 4. 4

HTML		
	appropriate location before an existing child. The method takes two parameters, the node object to be inserted and the existing node to insert before.	5. 5 1. //Creates a new element and places it in the elementList before the first child of 2. let newLI = document.createElement("li"); 3. newLI.innerText = "new Element"; 4. let elementList = document.getElementById("thisList"); 5. elementList.insertBefore(newLI, elementList.childNodes[0]); Copied!
Location Objects	The location object is part of the window object and contains information about the current URL.	1. 1 2. 2 3. 3 1. //Returns the hostname property 2. let myhost = location.hostname; 3. newLI.innerText = "new Element"; Copied!
Navigator Objects	The navigator object is part of the window object class in the DOM that represents the client Internet browser, also called the user agent. There is no standard for this object so what it returns differs from browser to browser.	1. 1 2. 2 1. //Retrieves the name of the browser 2. var browsername = navigator.appName; Copied!
onload()	A DOM event that starts a	1. 1 2. 2

HTML		
	method when a page is loaded.	1. //Executes myFunction after MyHTMLPage has been loaded 2. document.getElementById("MyHTMLPage").onload = function () {myFunction}; Copied!
replaceChild()	After creating an element, this function replaces a child node with a new node.	1. 1 2. 2 3. 3 4. 4 1. //Creates a new node and replaces the second element in "thisList" with the word "blue" 2. let secondBullet = document.createTextNode("blue"); 3. var myList = document.getElementById("thisList").childNodes[1]; 4. myList.replaceChild(secondBullet, myList.childNodes[1]); Copied!
Screen Objects	The screen object is part of the window object class in the DOM that can be used to return properties about the user's screen.	1. 1 2. 2 3. 3 1. //Returns the height and width of the user's screen 2. var height=screen.height; 3. var width=screen.width; Copied!
Window Objects	The DOM window object is at the top of the DOM hierarchy and serves as the global object.	1. 1 2. 2 1. //Opens a new browser window with the specified URL

HTML		
	Everything in the DOM takes place in a window. The window object controls the environment that contains the document.	2. window.open("http://www.w3schools.com"); Copied!
window.open()	Opens a new window. The first parameter is a path, a URL, or an empty string, and optional parameters include the window name, features such as the placement of the window or the dimensions, and a Boolean replace value. The feature parameter is a comma separated string of name-value pairs and the replace parameter is an optional Boolean. This parameter has been deprecated so modern browsers may not support it. This method returns a reference to the new window object.	1. 1 2. 2 3. 3 1. //Opens a new window that opens the IBM home page and has a width of 600 and a height of 800) 2. let thisWindow = window.open("http://www.ibm.com", "myWindow", 3. "width"=600, "height"=800); Copied!
window.scrollTo()	Scrolls to a particular place	1. 1

<table>
<tr><th colspan="3">HTML</th></tr>
<tr><td></td><td>in a window. Parameters include the x-coordinate which is the left-most pixel and the y-coordinate which is the upper-most pixel.</td><td>

```
//Scrolls the window to the pixel located at
the coordinate (20, 200)
window.scrollTo(20, 200);
```

Copied!</td></tr>
<tr><td>Wrapper Objects</td><td>Primitive types can be converted to objects using wrapper objects. They are the same name as the primitive except they start with uppercase letter. The typeof keyword returns a string indicating the data type of the operand.</td><td>

```
//Enables the use of properties and methods of
the String class such as the property n.length
let n = new String ("abc");

//Returns string
typeof "abc";

//Returns object
typeof new String("abc");
```

</td></tr>
</table>