Начало формы

Конец формы

1. **Engineering for Net**

**1. GIT**

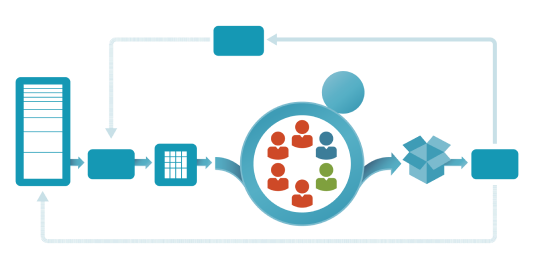
1. What is Version Control System?
2. What is Git?
3. Fundamentals of how Git works.
4. What is a Git Repository?
5. What are the Git objects?
6. What is file status lifecycle, file states within it?
7. How to install and configure Git?
8. What are the best committing practices?
9. What is the branching strategy? Types, difference.
10. Basic commands: get/create repository, basic snapshotting, sharing & updating projects, branching & merging.
11. When use merge and when rebase?
12. How to stash changes?
13. [Pro Git book](https://git-scm.com/book/en/v2)
14. [Git - the simple guide](https://rogerdudler.github.io/git-guide/)
15. [Git Glossary at Bitbucket](https://www.atlassian.com/git/glossary)
16. [Git cheat sheet by GitLab](https://about.gitlab.com/images/press/git-cheat-sheet.pdf)
17. [Version Control with Git](https://learn.epam.com/detailsPage?id=16d7f2e7-cc80-4870-928e-400723f732bb)
18. [Git Essential Training: The Basics](https://www.linkedin.com/learning/git-essential-training-the-basics/use-git-version-control-software-to-manage-project-code?autoplay=true&u=2113185)
19. [Git: Branches Merges and Remotes](https://www.linkedin.com/learning/git-branches-merges-and-remotes/unlock-powerful-code-management-and-collaboration-tools-in-git?autoplay=true&u=2113185)

**2. SCRUM**

1. What is SDM?
2. What is Waterfall?
3. When do we use Waterfall?
4. What is SCRUM?
5. What is Agile?
6. What is Kanban?
7. When do we use Scrum? Kanban?
8. What are the differences between Agile and SCRUM?
9. What are the differences between Kanban and SCRUM?
10. What is Agile Manifesto?
11. Should you write documentation according to Agile approach?
12. Should you sign any documents with the Customer according to Agile approach?
13. What SCRUM roles do you know?
14. What SCRUM meetings do you know?
15. What SCRUM artifacts do you know?
16. Can a developer be a Scrum Master?
17. Can a Business Analyst be a Scrum Master?
18. Can SCRUM Master be a Product Owner?
19. Is UX Designer a part of Developers according to SCRUM Guide?
20. [Software Development Methodologies training](https://learn.epam.com/detailsPage?id=094df392-3fa4-4332-8691-af73a6582abf)
21. [Agile Manifesto](https://agilemanifesto.org/)
22. [The Scrum Guide](https://www.scrum.org/resources/scrum-guide)
23. [Mural training for self-check](https://app.mural.co/template/84674b10-94b3-476c-bb16-ffae490420e2/d0854771-6e0f-4312-adf3-923745216d08)

**Tasks**

By the [link](https://static.scrum.org/web/images/Scrumorg-Scrum-Framework-scrubbed.png) you can find the Scrum helicopter view. To prepare for the interview please create your own Scrum helicopter view, where specify all the roles, meetings and artifacts you know in Scrum framework.



**3. CI/CD**

What is CI/CD?

CI/CD is a method to frequently deliver apps to customers by introducing automation into the stages of app development. The main concepts attributed to CI/CD are continuous integration, continuous delivery, and continuous deployment. CI/CD is a solution to the problems integrating new code can cause for development and operations teams (AKA "integration hell").

Specifically, CI/CD introduces ongoing automation and continuous monitoring throughout the lifecycle of apps, from integration and testing phases to delivery and deployment. Taken together, these connected practices are often referred to as a "CI/CD pipeline" and are supported by development and operations teams working together in an agile way with either a DevOps or site reliability engineering (SRE) approach.

1. What is CI/CD Pipeline? Why to use it? When to use it?
2. What are the advantages of CI/CD Pipeline?
3. What do you understand by fully automated CI/CD (Continuous Deployment) Pipeline?
4. What is difference between Continuous Deployment and Continuous Delivery?
5. How to choose right CI/CD Tool?
6. What are the best practices for managing CI/CD secrets?
7. What different CI/CD tools do you know?
8. What is Continuous Testing (CT)?
9. [CI/CD Pipeline: Learn with Exampl](https://www.guru99.com/ci-cd-pipeline.html)
10. [GitLab's guide to CI/CD for beginner](https://about.gitlab.com/blog/2020/07/06/beginner-guide-ci-cd/)
11. [TeamCity CI/CD Guid](https://www.jetbrains.com/teamcity/ci-cd-guide/)
12. [Getting started with Jenkin](https://www.jenkins.io/doc/tutorials/)
13. [Get started with GitLab CI/C](https://docs.gitlab.com/ee/ci/quick_start/)
14. [CI/CD Best Practice](https://www.jetbrains.com/teamcity/ci-cd-guide/ci-cd-best-practices/)
15. [Top 10 Benefits of Continuous Integration & Continuous Deliver](https://katalon.com/resources-center/blog/benefits-continuous-integration-delivery)

## Tasks

Using free-tier GitLab CI/CD, create console app and pipeline for it that includes three stages:

* Build
* Test
* Publish

Build step should contain build of your .net console app in release mode  
Test step should contain dotnet test step  
Publish step should contain some dummy 'echo' command that represent publish tier of your pipeline

Use three different stages, consider to use pipeline variables

**4. Containerization. Docker.**

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

SContainer images become containers at runtime and in the case of Docker containers – images become containers when they run on Docker Engine. Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

1. What is Docker?
2. What are the advantages of using Docker container?
3. What are the important features of Docker?
4. What is Docker image?
5. What is Docker Engine?
6. Explain Registries
7. List the most commonly used instructions in Dockerfile?
8. Whati is Dockerfile?
9. [Learn to build and deploy your distributed applications easily to the cloud with Docke](https://docker-curriculum.com/)
10. [Docker Tutorial for Beginner](https://www.youtube.com/watch?v=pTFZFxd4hOI)
11. [Getting started with Docke](https://docs.docker.com/get-started/)
12. [Get started with Docker Compos](https://docs.docker.com/compose/gettingstarted/)
13. [Networking in Compos](https://docs.docker.com/compose/networking/)
14. [Docker Compose Tutorial: advanced Docker made simpl](https://www.educative.io/blog/docker-compose-tutorial)
15. [Dockerize an ASP.NET Core applicatio](https://docs.docker.com/samples/dotnetcore/)
16. [Tutorial: Containerize a .NET ap](https://docs.microsoft.com/en-us/dotnet/core/docker/build-container?tabs=windows)

## Tasks

Basic

* Create a .NET Core console app
* Create Dockerfile and create image of your app
* Run container with your application

Advanced

* Create two .NET Core Web API's (also you can create containers for them or use docker-compose to build it)
* Create in each of them endpoint to call each service (from App1 call App2 and vice versa)
* Create docker-compose file and describe configurations of each app
* Run your docker-compose and call any of applications to invoke other

**2.NET BASICS**

**1.Memory management**

1. Which instance and type methods available for any type?
2. What are implicit and explicit type casts?
3. How operators 'is' and 'as' used? What is the result of their use?
4. Imagine, you have 2+ types with same type names. You need to use those in the same file. Adding their's full names in 'using' statements at the beginning of the file leads to compile-time error. Name 2 options to resolve the issue.
5. What are the differences between value and reference types?
6. What are boxing and unboxing? When these happen? What are the implications of their happening?
7. What are peculiarities of dynamic type?
8. What are managed and unmanaged resources?
9. What are the benefits of using GC?
10. What are the implications on which GC's work algorithm built?
11. What are the triggers for GC?
12. What phases GC algorithm has? Describe the purpose of each one.
13. What are GC roots? Name them.
14. How to free unmanaged resources?
15. What are safe handles?
16. Why to use full Disposing pattern?
17. What is freacheable Queue? What is finalization queue?
18. When to implement IDisposable interface?
19. CLR via C#, Jeffrey Richter, Chapter 4: Type Fundamentals
20. CLR via C#, Jeffrey Richter, Chapter 5: Primitive, Reference, and Value Types
21. [Garbage collection fundamentals](https://docs.microsoft.com/en-us/dotnet/standard/garbage-collection/fundamentals)
22. [Releasing unmanaged resources](https://docs.microsoft.com/en-us/dotnet/standard/garbage-collection/implementing-dispose)
23. Alternative way to study GC: Under the Hood of .NET Memory Management, Chris Farrell and Nick Harrison, Section 1: Introduction to .NET Memory Management

**Tasks**

1. Create a type that will be collected by GC only on end of program without using static.
2. Create a type that will be resurrecting itself from finalization queue.
3. What will be displayed when this piece of code is run? Run and check. class Program

{

static void Main(string[] args)

{

var points = new List(Enumerable.Repeat(0, 10).Select(p => new Point()));

foreach (var p in points)

{

p.IncX();

}

foreach (var p in points)

{

Console.WriteLine(p.X);

}

Console.ReadKey();

}

}

public struct Point

{

public int X;

public void IncX()

{

X++;

}

}

1. Fix the code above to display the expected result (ask mentor what is expected, if you hesitate).

**2.Multithreading**

1. What is a thread? What is a process?
2. What problem is resolved by using multithreading?
3. What is ManualResetEvent? What is AutoResetEvent? What is Semaphore? What is Mutex? What is Monitor?
4. What is lock statement? How it looks under the hood?
5. When to use thread-safe collections? Which ones do you know?
6. What is the difference between foreground and background thread?
7. What is thread pool?
8. What is a timer?
9. [Threads and threading](https://docs.microsoft.com/en-us/dotnet/standard/threading/threads-and-threading)
10. [Overview of synchronization primitives](https://docs.microsoft.com/en-us/dotnet/standard/threading/overview-of-synchronization-primitives)
11. [Lock statement](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/statements/lock)
12. [Thread-safe collections](https://docs.microsoft.com/en-us/dotnet/standard/collections/thread-safe/)
13. [When to use thread-safe collections](https://docs.microsoft.com/en-us/dotnet/standard/collections/thread-safe/when-to-use-a-thread-safe-collection)
14. [Foreground and background threads](https://docs.microsoft.com/en-us/dotnet/standard/threading/foreground-and-background-threads)
15. [Managing threading best practices](https://docs.microsoft.com/en-us/dotnet/standard/threading/managed-threading-best-practices)
16. [Thread pool](https://docs.microsoft.com/en-us/dotnet/standard/threading/the-managed-thread-pool)
17. [Timers](https://docs.microsoft.com/en-us/dotnet/standard/threading/timers)

**Tasks**

1. Work out examples in[this page](https://docs.microsoft.com/en-us/dotnet/standard/threading/creating-threads-and-passing-data-at-start-time)
2. Create a simulation, in which there is:
   * a library, that may let in N readers at a time
   * M readers come to the library and spent there random amount of time from T1 to T2

Every entry to and exit from library should be out to console. You should also log amount of time each user spends in the library. N, M, T1, T2 are set from console.

**3.TPL & Asynchronous Programming**

 Success Criteria

1. Why to use asynchrony?
2. What is difference between different asynchrony models?
3. What are Tasks? How to use them without async/await?
4. What will happen under the hood, when you mark method as async?
5. What will be the difference between calls "DoSomethingAsync()" and "await DoSomethingAsync()" in async method?
6. What is the difference between WaitAll and WhenAll?
7. How to cancel a Task?
8. Why and when to use PLINQ? When and why to use Parallel.For/Parallel.ForEach?

 References

1. [Asynchrony](https://docs.microsoft.com/en-us/dotnet/standard/async)
2. [Asynchronous models in .NET](https://docs.microsoft.com/en-us/dotnet/standard/asynchronous-programming-patterns/)
3. [Tasks](https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-based-asynchronous-programming)
4. [Asynchronous programming with async and await](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/)
5. [Aynchronous programming with TPL in depth](https://docs.microsoft.com/en-us/dotnet/standard/asynchronous-programming-patterns/consuming-the-task-based-asynchronous-pattern)
6. [Task cancellation](https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-cancellation)
7. [Introduction to PLINQ](https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/introduction-to-plinq)

## Tasks

1. Create a chain of tasks, each one will start after the previous is finished. Do it without using async/await, only Task type methods.
2. Create a console program and make async Main.
3. Do task 1 with async/await.
4. Imagine you're doing some time-consuming work (Thread.Sleep & return a\*a) and you want to parallelize the work. Do this with PLINQ and Parallel.ForEach.

**4.EF Code First features**

Entity Framework (EF) is an object-relational mapper that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write. The Entity Framework provides three approaches to create an entity model: Code First, Database First, Model First. Code First modeling workflow targets a database that does not exist, and Code First will create it based on your entity classes and configuration. The development workflow in the Code First approach would be: Create or modify domain classes -> Configure these domain classes using Fluent-API or data annotation attributes -> Create or update the database schema using automated migration or code-based migration.

 Success Criteria

1. What is Entity Framework?
2. What are different types of Entity Framework approaches? Pros and cons of each.
3. What is Code First approach in Entity Framework and its stages?
4. What are Data Annotations? What are best practices of using?
5. What is Fluent Api? Basic classes and methods of Fluent Api.
6. What is Entity Framework's Database Seed Method? How is it used?
7. What is the Migration in Entity Framework? What does it provide?
8. What are types of Migrations and what is the difference?
9. Main commands for implementation of Migration process.

 References

1. [Entity Framework overview](https://www.entityframeworktutorial.net/what-is-entityframework.aspx)
2. [Entity Framework approaches](https://www.entityframeworktutorial.net/choosing-development-approach-with-entity-framework.aspx)
3. [Entity Framework Code First approach](https://www.ecanarys.com/Blogs/ArticleID/228/Entity-Framework-CodeFirst-Approach)
4. [Entity Framework - Data Annotations](https://www.tutorialspoint.com/entity_framework/entity_framework_data_annotations.htm)
5. [Fluent API – Relationships](https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/fluent/relationships)
6. [Fluent API - Configuring and Mapping Properties and Types:](https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/fluent/types-and-properties)
7. [Entity Framework - Seed Database](https://www.tutorialspoint.com/entity_framework/entity_framework_seed_database.htm)
8. [Migration overview](https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/migrations/)
9. [Types of Migrations](https://www.entityframeworktutorial.net/code-first/migration-in-code-first.aspx)
10. [Automated Migration in Entity Framework](https://www.entityframeworktutorial.net/code-first/automated-migration-in-code-first.aspx)
11. [Code-Based Migration in Entity Framework](https://www.entityframeworktutorial.net/code-first/code-based-migration-in-code-first.aspx)
12. [Code First Migrations in Team Environments](https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/migrations/teams)

## Tasks

1. Work with example of [EF Code First approach](https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/workflows/new-database)
2. Using Entity Framework Code First approach create database “Library” with next tables:
   * “Readers” – every record contains basic information about person who takes book: id, name, phone number, email, reader's card id.
   * “Cards” – contains id, reader id, book id that's currently taken by reader (only one allowed).
   * “Books” – id, name of book, author id.
   * “Authors” – id, name, list of books.
   * Define relationships between tables.

It is basic requirements. You can add your features to the database.

**5.Repository & Unit of Work**

The Repository pattern is used to decouple the business logic and the data access layers in application. In using the Repository design pattern, the business logic layer of application need not have any knowledge on how data persistence happens beneath. Essentially, a repository mediates between the domain and the data mapping layers of your application. It's supposed to provide you an encapsulation on the way that data is actually persisted in the data storage layer. Unit of Work is the concept related to the effective implementation of the Repository pattern. Unit of Work is referred to as a single transaction that involves multiple operations of insert/update/delete and so on.

 Success Criteria

1. What are Design patterns?
2. What is Repository pattern?
3. What is the goal of using Repository pattern?
4. What is Unit of Work pattern and its function?
5. What is the role of Unit of Work pattern in Repository pattern implementation?
6. What SOLID principles are implemented in using Repository pattern?
7. What are best practices of Repository and Unit of Work patterns implementations?
8. What kinds of applications are Repository pattern/Unit of Work used for?
9. What are pros and cons of using Repository pattern/Unit of Work?

 References

1. [Design Patterns In C# .NET](https://www.c-sharpcorner.com/UploadFile/bd5be5/design-patterns-in-net/)
2. [Repository Design Pattern](https://medium.com/@pererikbergman/repository-design-pattern-e28c0f3e4a30)
3. [Repository Pattern features](https://deviq.com/design-patterns/repository-pattern)
4. [Unit of Work](https://martinfowler.com/eaaCatalog/unitOfWork.html)
5. [Unit Of Work in Repository Pattern](https://dotnettutorials.net/lesson/unit-of-work-csharp-mvc/)
6. [SOLID Architecture and the Repository Pattern](https://topherlandry.wordpress.com/2015/01/26/solid-architecture-and-the-repository-pattern/)
7. [Design the infrastructure persistence layer](https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design)
8. [Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application)
9. [Is the repository pattern useful with Entity Framework Core?](https://www.thereformedprogrammer.net/is-the-repository-pattern-useful-with-entity-framework-core/)

## Tasks

1. Work with [example](https://www.c-sharpcorner.com/UploadFile/b1df45/unit-of-work-in-repository-pattern/" \t "_blank)
2. Using Entity Framework Code First approach create database “Library” with next tables:
   * “Readers” – every record contains basic information about person who takes book: id, name, phone number, email, reader’s card id.
   * “Cards” – contains id, reader id, book id that’s currently taken by reader (only one allowed).
   * “Books” – id, name of book, author id.
   * “Authors” – id, name, list of books.
   * Define relationships between tables.

It is basic requirements. You can add your features to the database. Implement CRUD operations for database using Repository/Unit of Work patterns.

**6. DI Overview**

Dependency Injection (DI) is a design pattern used to implement IoC. It allows the creation of dependent objects outside of a class and provides those objects to a class through different ways. Using DI, we move the creation and binding of the dependent objects outside of the class that depends on them.

.NET supports the dependency injection (DI) software design pattern, which is a technique for achieving Inversion of Control (IoC) between classes and their dependencies. Dependency injection in .NET is a built-in part of the framework, along with configuration, logging, and the options pattern.

 Success Criteria

1. Define What Is Dependency Injection?
2. What Are The Different Ways To Implement Dependency Injection?
3. Provide An Overview Of How Dependency Injection Works?
4. What Are The Advantages Of Dependency Injection?
5. What Are The Disadvantages Of Dependency Injection?
6. What Dependency Injection containers do you know?
7. Does pure .NET Core supports Dependency Injection?

 References

1. [Dependency injection in .NET](https://elearn.epam.com/courses/course-v1:RD_CEE+NBfNL+0622/courseware/5791982843d14ab6a8dcf8305155ea06/249b64848c2a43ea866be81272b3b39b/(https:/docs.microsoft.com/en-us/dotnet/core/extensions/dependency-injection))
2. [Implementation of Dependency Injection Pattern in C#](https://elearn.epam.com/courses/course-v1:RD_CEE+NBfNL+0622/courseware/5791982843d14ab6a8dcf8305155ea06/249b64848c2a43ea866be81272b3b39b/(https:/www.dotnettricks.com/learn/dependencyinjection/implementation-of-dependency-injection-pattern-in-csharp))
3. [Dependency injection guidelines](https://docs.microsoft.com/en-us/dotnet/core/extensions/dependency-injection-guidelines)
4. [AddTransient AddScoped and AddSingleton Services Differences](https://stackoverflow.com/questions/38138100/addtransient-addscoped-and-addsingleton-services-differences)
5. [Integration tests in ASP.NET Core](https://docs.microsoft.com/en-us/aspnet/core/test/integration-tests)
6. [Differences Between Scoped Transient And Singleton Service](https://www.c-sharpcorner.com/article/differences-between-scoped-transient-and-singleton-service/)

## Tasks

1. Create console application and try build-in Dependency Injection container
2. Inject services in different ways
3. Use different types of registration (Scoped, Transient, And Singleton)

**7.Unit testing and TDD**

#### Types of Testing. Unit test frameworks and extensions (MSTest, NUnit, XUnit etc.). Triple A. TDD (overview). Mock, Fake

Check that your code is working as expected by creating and running unit tests. It's called unit testing because you break down the functionality of your program into discrete testable behaviors that you can test as individual units. Unit testing has the greatest effect on the quality of your code when it's an integral part of your software development workflow. As soon as you write a function or other block of application code, create unit tests that verify the behavior of the code in response to standard, boundary, and incorrect cases of input data, and that check any explicit or implicit assumptions made by the code.

**Success Criteria**

1. What are types of testing?
2. What is unit testing?
3. Name conventions for unit tests
4. What is AAA?
5. What is test driven development (TDD)?
6. What testing frameworks are there?
7. Common guidelines for writing unit tests
8. What is code coverage
9. Unit tests’ role in CI/CD
10. Mock, Fake

 References

1. Roy Osherove, The Art of Unit Testing: with examples in C# 2nd Edition
2. [Unit testing basics](https://docs.microsoft.com/en-us/visualstudio/test/unit-test-basics?view=vs-2022)
3. [Unit testing with MSTest](https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-mstest)
4. [Unit testing with NUnit](https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-nunit)
5. [NUnit documentation](https://docs.nunit.org/)
6. [Unit testing with XUnit](https://docs.microsoft.om/en-us/dotnet/core/testing/unit-testing-with-dotnet-test)
7. [XUnit documentation](https://xunit.net/#documentation)

**8.Multithreading and asynchronous programming**

With .NET, you can write applications that perform multiple operations at the same time. Operations with the potential of holding up other operations can execute on separate threads, a process known as multithreading or free threading. Applications that use multithreading are more responsive to user input because the user interface stays active as processor-intensive tasks execute on separate threads.

With asynchronous programming model you write code as a sequence of statements, just like always. You can read that code as though each statement completes before the next begins. The compiler performs many transformations because some of those statements may start work and return a Task that represents the ongoing work.

 Success Criteria

1. What is thread? Background and foreground threads.
2. How to start a new thread? How to stop, pause a thread?
3. What are signaling structures?
4. Shared context and data synchronization
5. Thread exception handling
6. What is Threadpool?
7. What is asynchronous programming model?
8. What is Task Parallel Library (TPL)?
9. What is Task?
10. What is Task Continuation?
11. How does async-await work?

 References

1. [Overview of synchronization primitives](https://docs.microsoft.com/en-us/dotnet/standard/threading/overview-of-synchronization-primitives)
2. [Using threads and threading](https://docs.microsoft.com/en-us/dotnet/standard/threading/using-threads-and-threading)
3. [Task Parallel Library](https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-parallel-library-tpl)
4. [Task](https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task?view=net-6.0)
5. [Thread Pool](https://docs.microsoft.com/en-us/dotnet/standard/threading/the-managed-thread-pool)
6. [Task-based asynchronous programming](https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-based-asynchronous-programming)
7. [Continuations](https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/chaining-tasks-by-using-continuation-tasks)
8. [Task cancellation](https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-cancellation)
9. [Exception handling](https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/exception-handling-task-parallel-library)
10. [Asynchronous programming with async and await](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/)

**3.Web**

**1.ASP.NET fundamentals**

 Success Criteria

1. Which OS could use Web Forms applications?
2. Which programming languages could be used for ASP.NET Web Pages development?
3. Describe MVC lifecycle.
4. What is difference between web forms and web pages?
5. Could you use MVC in .NET Core?
6. Could you use Web Forms in .NET Core?
7. What is blazor?
8. How to store DB connection in MVC application?
9. Describe OSI model. Which layers are responsible for data transfer (PDU)?
10. What is session in terms of http?
11. How do you undestand term token?

 References

1. [Web pages](https://docs.microsoft.com/en-us/aspnet/web-pages/)
2. [.NET MVC (Framework)](https://docs.microsoft.com/en-us/aspnet/mvc/)
3. [.NET Core MVC](https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0)
4. [Web forms](https://docs.microsoft.com/en-us/aspnet/web-forms/)
5. [Asp.Net Core configuration](https://asp.mvc-tutorial.com/core-concepts/configuration/)
6. [MVC Lifecycle](https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_life_cycle.htm)
7. [Blazor for Asp.NET Web Forms](https://devblogs.microsoft.com/dotnet/blazor-aspnet-webforms-ebook/)
8. [State management](https://www.c-sharpcorner.com/article/state-management-in-asp-net-mvc/)
9. [OSI model](https://en.wikipedia.org/wiki/OSI_model)
10. [Http sessions overview](https://docs.progress.com/bundle/pas-for-openedge-administration-117/page/Overview-of-HTTP-sessions.html#:~:text=HTTP%20sessions%20is%20an%20industry,application%20and%20a%20Web%20application.)
11. [Session example](https://developer.mozilla.org/en-US/docs/Web/HTTP/Session)
12. [Http tokens and authentication](https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication)

**2.MVC. Controllers. Routing**

For communication frontend and backend parts of your application you should define communication endpoints and their types. Setup of this part on backend is known as routing. Also here you can find useful information about how to store client data and how to get data about current request.

 Success Criteria

1. What is Controller in terms of MVC?
2. Which type of request you will use for data update?
3. Please provide example of route which you will use to delete user record based on its id.
4. How to get current request data?
5. Where you can store required user information from request?

 References

1. [Request types](https://www.c-sharpcorner.com/article/crud-with-asp-net-web-api-part-one/)
2. [MVC Controllers](https://www.tutorialsteacher.com/mvc/mvc-controller)
3. [MVC Routing](https://www.c-sharpcorner.com/UploadFile/3d39b4/routing-in-mvc/)
4. [MVC CRUD](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/nerddinner/provide-crud-create-read-update-delete-data-form-entry-support)
5. [Action result](https://www.c-sharpcorner.com/article/action-result-in-asp-net-mvc/#:~:text=Action%20Result%20is%20actually%20a,of%20action%20when%20it%20executes.)
6. [Request context](https://docs.microsoft.com/en-us/dotnet/api/system.web.routing.requestcontext?view=netframework-4.8)
7. [HttpContextBase](https://docs.microsoft.com/en-us/dotnet/api/system.web.httpcontextbase?view=netframework-4.8)
8. [Cookies MVC](https://asp.mvc-tutorial.com/httpcontext/cookies/)

**3.MVC. Views. Controller-View data transfer. Razor.**

To implement front end part in MVC you will use views. It's component which is suitable for providing UI for interact with your data. In this section you will find information about what and how you can do UI in MVC.

 Success Criteria

1. What is view and what it should store?
2. What is Layout View in ASP.NET MVC?
3. How to pass data from controller to view and from view to controller.
4. What is html helper?
5. What is bootstrap and when you should use it?
6. What is razor?

 References

1. [Views in MVC Core](https://docs.microsoft.com/en-us/aspnet/core/mvc/views/overview?view=aspnetcore-6.0)
2. [Views in MVC Framework](https://www.tutorialsteacher.com/mvc/mvc-view)
3. [Views paths](https://stackoverflow.com/questions/799838/asp-net-mvc-how-to-specify-which-folder-the-view-pages-reside-in)
4. [Controller-View data transfer](https://www.c-sharpcorner.com/UploadFile/df5827/data-transfer-between-view-and-controller-Asp-Net-mvc-5/)
5. [Razor](https://en.wikipedia.org/wiki/ASP.NET_Razor#:~:text=Razor%20is%20an%20ASP.NET,and%20the%20WebMatrix%20tool%20set)
6. [Layout view MVC](https://www.tutorialsteacher.com/mvc/layout-view-in-asp.net-mvc)
7. [Layout and sections](https://www.c-sharpcorner.com/UploadFile/3d39b4/Asp-Net-mvc-4-layout-and-section-in-razor/)
8. [Partial views](https://docs.microsoft.com/en-us/aspnet/core/mvc/views/partial?view=aspnetcore-6.0)
9. [HTML helpers](https://www.tutorialsteacher.com/mvc/html-helpers)
10. [Bootstrap for beginners](https://www.c-sharpcorner.com/article/bootstrap-for-beginners-part-one-introduction-and-impleme/)

**4.MVC. Validation. Exception handling. Filters**

This chapter will be useful to get information about validation and data filtering what will allow to decrease amount of calls to your backend part of application and move some part of validation responsibilities to frontend.

Also here you can find basic information about authentication and authorization.

 Success Criteria

1. How you can validate model data in MVC?
2. Where you will add validation for password
3. complexity\length\existance?
4. What is metadata and what is metadata validation?
5. What is client-side-validation?
6. What is the difference between client and server-side validation?
7. What is MVC filters? Which type do you know?
8. What is custom filters? Provide example of usage custom filter.
9. What is the difference between Authentication and Authorization?
10. What is asp.net identity?

 References

1. [Model validation](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-aspnet-mvc3/cs/adding-validation-to-the-model)
2. [Model validation (+Validation messages)](https://www.programmingwithwolfgang.com/model-validation-in-asp-net-mvc/)
3. [Model validation MVC Core](https://docs.microsoft.com/en-us/aspnet/core/mvc/models/validation?view=aspnetcore-6.0)
4. [Metadata validation](https://www.codeproject.com/Questions/830585/How-to-add-Metadata-Validation-to-MVC-model-using)
5. [Metadata validation](https://stackoverflow.com/questions/21707323/validate-an-object-using-its-metadata-from-outside-of-the-controller-or-the-view)
6. [Client-side validation](https://www.c-sharpcorner.com/UploadFile/3d39b4/Asp-Net-mvc-client-side-validation/)
7. [Client-side validation vs server-side validation](http://net-informations.com/faq/asp/validation.htm)
8. [Filters (+types) MVC](https://www.tutorialsteacher.com/mvc/filters-in-asp.net-mvc)
9. [Custom filters MVC](https://www.c-sharpcorner.com/UploadFile/8a67c0/custom-action-filters-in-Asp-Net-mvc/)
10. [Authentication vs. Authorization](https://www.sailpoint.com/identity-library/)
11. [Custom authorization MVC](https://www.c-sharpcorner.com/UploadFile/56fb14/custom-authorization-in-mvc/)
12. [Custom authentication MVC](https://www.c-sharpcorner.com/article/custom-authentication-with-asp-net-mvc/)
13. [Asp.NET identity](https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity)

**5.MVC. Authentication & authorization. ASP.NET Identity**

#### Data and view models. Views with non-typed and typed models. Scaffolding. Strongly typed helpers. Data transfer to controller. Model binding basics. Value providers. Primitive and complex type binding. Collection binding.

This chapter will provide information about data bindings in MVC models and views and useful tools which will help you to resolve frequent problems.

 Success Criteria

1. How to bind model to view?
2. What is scaffolding?
3. How to bind collection to view?
4. What is custom value provider?
5. What is model in MVC?
6. What is the difference between:
   * view and view model?
   * dynamic and strongly typed views?
   * Value provider and Model Binder?

 References

1. [Data models MVC example](https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_data_model.htm)
2. [Model in MVC](https://www.tutorialsteacher.com/mvc/mvc-model)
3. [View and view models](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/mvc-music-store/mvc-music-store-part-3)
4. [Managing data with viewmodel in MVC](https://www.c-sharpcorner.com/article/managing-data-with-viewmodel-in-asp-net-mvc/)
5. [Dynamic vs strongly typed views](https://docs.microsoft.com/en-us/aspnet/mvc/overview/views/dynamic-v-strongly-typed-views)
6. [Scaffolding](https://www.dotnettricks.com/learn/mvc/understanding-aspnet-mvc-scaffolding#:~:text=Scaffolding%20is%20a%20technique%20used,code%20according%20to%20your%20need.)
7. [Strongly type html helpers mvc](https://www.c-sharpcorner.com/article/overview-of-strongly-type-html-helpers-in-mvc/)
8. [Model bindings basics](https://www.tutorialspoint.com/asp.net_mvc/asp.net_mvc_model_binding.htm#:~:text=Model%20binding%20is%20a%20well,complete%20this%20behind%20the%20scene.)
9. [Custom value providers](https://www.c-sharpcorner.com/UploadFile/97fc7a/smart-working-with-custom-value-providers-in-Asp-Net-mvc/#:~:text=CustomValueProvider.rar,This%20article%20describes%20Custom%20Value%20Providers%20in%20MVC%20and%20their,use%20at%20the%20action%20level.)
10. [Value provider vs Model Binder](https://stackoverflow.com/questions/35798726/asp-net-mvc-what-s-the-difference-between-a-value-provider-and-model-binder)
11. [Bind Query string to an Action Method Parameters in MVC](https://www.tutorialsteacher.com/mvc/model-binding-in-asp.net-mvc)
12. [Model binding with dynamic collection](https://stackoverflow.com/questions/40681707/mvc-model-binding-with-dynamic-collection)

**6.REST & ASP.NET Web API overview**

#### ASP.NET Web API overview. REST, RESTful. RPC vs SOAP vs HTTP. HTTP request-response. HTTP processing pipeline. Fiddler and Postman.

Internet is a common thing nowadays and HTTP is a big deal of it. How does it work? What do request and response look like? How modern applications, APIs in part, are created and tested? In this module you can find the answers ot these question.

 Success Criteria

1. What does service-oriented architecture mean?
2. What does microservice architecture mean?
3. What is RPC? SOAP? HTTP?
4. What is REST?
5. Describe the levels of Richards' maturity model.
6. What is the structure of HTTP message?
7. Describe the pipelining feature of HTTP?
8. What is Postman/Fiddler?

 References

1. [SOA](https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/service-oriented-architecture)
2. [Microservices architecture](https://docs.microsoft.com/en-us/dotnet/architecture/microservices/architect-microservice-container-applications/microservices-architecture)
3. [Rest & RestFul web API design](https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design)
4. [Richardson maturity model](https://martinfowler.com/articles/richardsonMaturityModel.html)
5. [RPC](https://www.enterpriseintegrationpatterns.com/EncapsulatedSynchronousIntegration.html)
6. [SOAP](https://www.w3.org/TR/2000/NOTE-SOAP-20000508/)
7. [HTTP](https://datatracker.ietf.org/doc/html/rfc2616)
8. [HTTP message structure](https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages)
9. [HTTP Request path](https://www.halldorstefans.com/how-an-http-request-travels-the-internet/)
10. [HTTP pipelining feature](https://en.wikipedia.org/wiki/HTTP_pipelining)
11. [Create first ASP.NET Core App](https://docs.microsoft.com/en-us/aspnet/core/tutorials/min-web-api?view=aspnetcore-6.0&tabs=visual-studio)
12. [Web Api naming conventions](https://docs.microsoft.com/en-us/aspnet/core/web-api/advanced/conventions?view=aspnetcore-6.0)
13. [What is Postman](https://www.postman.com/product/what-is-postman/)
14. [What is Fiddler](https://www.telerik.com/fiddler/fiddler-classic)

**7.Web API. Controllers. Request handling.**

#### Request handling. Action methods. Action Results in WebApi. Data models. Model binding and validation. JSON and XML serialization in WebApi. Build RESTful APIs with ASP.NET WebApi.

ASP.NET is a great multifunctional engine for APIs creation based on .NET. This topic will acknowledge you about the basics of ASP.NET Web.Api.

 Success Criteria

1. What are the differences between DTO and POCO?
2. What is parameter binding? Which options are there to bind a request into a model?
3. What is needed to apply validation on Api level?
4. What is routing?
5. How can you map requests (2 options)?
6. What kind of constraints can you apply in routing templates?
7. What are middlewares?
8. How to enable JSON/XML serialization in WebApi?

 References

1. [DTO vs POCO](https://social.msdn.microsoft.com/Forums/en-US/f969dbce-1361-47f3-8e10-561516c7741b/what-is-difference-between-dto-and-poco-classes?forum=asparchitecture)
2. [Parameter binding](https://docs.microsoft.com/en-us/aspnet/web-api/overview/formats-and-model-binding/parameter-binding-in-aspnet-web-api)
3. [Data model validation](https://docs.microsoft.com/en-us/aspnet/web-api/overview/formats-and-model-binding/model-validation-in-aspnet-web-api)
4. [Routing](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-6.0)
5. [Middlewares](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-6.0)
6. [Request/Response serializaton](https://docs.microsoft.com/en-us/aspnet/web-api/overview/formats-and-model-binding/json-and-xml-serialization)

**8.Web API. Error handling. Unit testing Web API. Routing.**

1. How to enable global error handler in ASP.NET WebApi?
2. What types of filters are there in ASP.NET? What order are they executed in?
3. What purpose does each type of filter should serve to?

 References

1. [Exception handling](https://docs.microsoft.com/en-us/aspnet/core/web-api/handle-errors?view=aspnetcore-6.0)
2. [Filters](https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/filters?view=aspnetcore-6.0)
3. [Unit testing controllers](https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/testing?view=aspnetcore-6.0)

**9.Web API. Authentication and Authorization in Web API**

#### Authentication and Authorization. Identity. Authentication and Authorization in Web API. Authentication Filters in Web API. SSL

To protect your application you should know about authentication, authorization and SSL/TLS. This module also gives some predefined ways of securing you applications.

 Success Criteria

1. What is the difference between authentication and authorization?
2. What is JWT? What does its structure look like?
3. What is Identity?
4. What is CORS?
5. What is the difference between HTTP and HTTPS?
6. How to enforce https in ASP.NET Web.Api?

 References

1. [Authorization vs authentication](https://auth0.com/docs/get-started/identity-fundamentals/authentication-and-authorization)
2. [What is JWT](https://jwt.io/introduction)
3. [JWT Validation and Authorization in ASP.NET Core](https://devblogs.microsoft.com/dotnet/jwt-validation-and-authorization-in-asp-net-core/)
4. [Introduction to Identity](https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0&tabs=visual-studio)
5. [CORS](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS)
6. [Enable CORS in ASP.Net Core](https://docs.microsoft.com/en-us/aspnet/core/security/cors?view=aspnetcore-6.0)
7. [HTTP vs HTTPS](https://www.guru99.com/difference-http-vs-https.html)
8. [Enforce HTTPS](https://docs.microsoft.com/en-us/aspnet/core/security/enforcing-ssl?view=aspnetcore-6.0&tabs=visual-studio)
9. [A bit about SSL and TLS](https://www.cloudflare.com/en-gb/learning/ssl/what-happens-in-a-tls-handshake/)

**4.Linq essential training**

<https://www.linkedin.com/learning/linq-with-c-sharp-essential-training?dApp=53239054&leis=LAA&u=2113185>

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Flinq-with-c-sharp-essential-training%3Ftrk%3Dshare_ent_url%26shareId%3DzO3ibrdoRMGneH9hrArSMg%253D%253D>

**5.C# best practices**

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fc-sharp-best-practices-for-developers%3Ftrk%3Dshare_ent_url%26shareId%3D8EZdZZ3dQ7uvHa1fS51pCA%253D%253D>

**6.Relational Databases Essential Training**

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Frelational-databases-essential-training%3Ftrk%3Dshare_ent_url%26shareId%3DkNkWayVjSyCBmoj7ORzYSA%253D%253D>

7.NoSQL Essential Training

Description

As the shiny new object in the data world, you might have heard a lot of people talk excitedly about NoSQL and all the things it can do. It’s great in terms of flexibility, speed, and is easy to work with. It’s super scalable, so it can accommodate increased numbers of users as websites and applications grow. But will it replace SQL? Will it make relational databases obsolete? In this course, Mel McGee explains just exactly what NoSQL is, the pros and cons, and tradeoffs you’ll make when using NoSQL. Mel takes a high-level approach without delving into the details of any one NoSQL query language or solution, so if you’re a developer looking for a bigger picture of NoSQL, or an entrepreneur wanting to explore options for your product, or just plain curious about non-relational databases, this course is for you.

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fnosql-essential-training%3Ftrk%3Dshare_ent_url%26shareId%3D3DvPuPEnS6izALjSfjz5lg%253D%253D>

8.Introduction to MongoDB

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fintroduction-to-mongodb%3Ftrk%3Dshare_ent_url%26shareId%3Dd8ll4Eg7Q6Gzt%252BWF7QCFJg%253D%253D>

9.Building and Securing RESTful APIs in ASP.NET Core

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fbuilding-and-securing-restful-apis-in-asp-dot-net-core-2018%3Ftrk%3Dshare_ent_url%26shareId%3DWg5EqJGfR7ymlcmLHf5Sug%253D%253D>

10.Learning ASP.NET Core MVC

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Flearning-asp-dot-net-core-mvc%3Ftrk%3Dshare_ent_url%26shareId%3DWjHGOGKCQROULUfJ11nr9Q%253D%253D>

**11.ASP.NET Core: Token-Based Authentication**

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fasp-dot-net-core-token-based-authentication%3Ftrk%3Dshare_ent_url%26shareId%3DsPQUEKiUS3qBmK9T8Ig%252B3g%253D%253D>

**12.Web Security: User Authentication and Access Control**

Начало формы

Начало формы

Конец формы

**13.Unit Testing for .Net**

**1.Test Implementation**

**2.Test Execution**  
**3.Test Reporting**

**Homework**

* 1. Write unit tests for the calculator dll library ([**Calculator1.dll**](https://elearn.epam.com/assets/courseware/v1/2e4c1efc326f49b202302bcc32ec9e92/asset-v1:EPAM+UTN+1120+type@asset+block/Calculator1.dll.zip)).
  2. Setup parallel test execution.
  3. Configure html report for your test run.

### Requirements

* + Add Calculator1.dll to your project references.
  + Write tests using Nunit or MsTest.
  + Each class should test only one method in library (1 method – 1 class).
  + Should test ability to enter numbers as well as string numbers.
  + Add any additional logger to your project for good logs.
  + Run tests in Visual Studio.
  + Run tests in command line (MSTest , nunit-console) with specified output directory for results.
  + Convert your test logs into html using any tool.

14.C# Test-Driven Development (Linkedin learning)

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fc-sharp-test-driven-development-14275015%3Ftrk%3Dshare_ent_url%26shareId%3DoEDEKfKnRw69YXISRYH8lA%253D%253D>

15.ASP.NET Core: Test-Driven Development

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fasp-dot-net-core-test-driven-development%3Ftrk%3Dshare_ent_url%26shareId%3D%252FQi29PMOSQClU4HubCKmfg%253D%253D>

**16.Frontend basics for Net**

**HTTP, JSON, REST**

REST (Respresentational State Transfer) - architectural style for developing web applications to guide the design and development of the architecture for the World Wide Web. REST is not strictly based on using HTTP and JSON, meaning it can be used with other transfering protocols and data formats. However those are the most common components for REST today. JSON (JavaScript Object Notation) is a text-based data format and HTTP (Hypertext Transfer Protocol) is a transfer protocol which are used to transfer state of informational resources in RESTful applications. There are a set of constrains that application should follow to be considered RESTful and thus benefit from this arcitectural style.

 Success Criteria

Know answers to questions:

1. What is REST, JSON, HTTP?
2. REST vs RESTful?
3. What are the alternitives of REST?
4. What are the benefits and downsides of REST?
5. What are the alternitives of JOSN?
6. WHat is the structure of HTTP request, response?
7. What is HTTP request-response cycle?
8. What is idempotent request?
9. What is the difference between POST/GET? PATCH/PUT?
10. URI vs URL vs URN?
11. What is the correct way to form URL in RESTful applications?

 References

1. [SQL](https://en.wikipedia.org/wiki/SQL)
2. [Database](https://www.oracle.com/database/what-is-database/)
3. [SQL as programming language](https://learnsql.com/blog/sql-programming-language/)
4. [Difference between SQL and PL/SQL](https://www.geeksforgeeks.org/difference-between-sql-and-plsql/)
5. [Difference between In and Between Operator in SQL](https://www.c-sharpcorner.com/blogs/difference-between-in-and-between-operator-in-sql1)
6. [SQL Data Types](https://www.w3schools.com/sql/sql_datatypes.asp)
7. [Data Definition Language (DDL)](https://www.microfocus.com/documentation/enterprise-developer/ed70/ED-VS2019/HRQRRHSQLX02.html)
8. [Data Manipulation Language (DML)](https://www.microfocus.com/documentation/enterprise-developer/ed70/ED-VS2019/HRQRRHSQLX03.html)
9. [Primary key Vs Unique key](https://www.tutorialspoint.com/primary-key-vs-unique-key)
10. [SQL Views](https://www.w3schools.com/sql/sql_view.asp)
11. [SQL FOREIGN KEY Constraint](https://www.w3schools.com/sql/sql_foreignkey.asp)
12. [SQL Join (Inner Left Right and Full Joins)](https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/)
13. [SQL Query](https://www.techopedia.com/definition/5736/query)
14. [SQL - Sub Queries](https://www.tutorialspoint.com/sql/sql-sub-queries.htm)
15. [SQL – Operators](https://www.tutorialspoint.com/sql/sql-operators.htm)
16. [Difference between SQL Truncate and SQL Delete statements in SQL Server](https://www.sqlshack.com/difference-between-sql-truncate-and-sql-delete-statements-in-sql-server/)
17. [Variable Types in SQL: Local Global](https://www.guru99.com/sql-server-variable.html#variable-types-in-sql)
18. [Introducing SQL Set Operators: Union Union All Minus and Intersect](https://learnsql.com/blog/introducing-sql-set-operators-union-union-minus-intersect/)
19. [T-SQL Tutorial](https://www.tutorialspoint.com/t_sql/index.htm)

## Tasks

1. [SQL Exercises, Practice, Solution - Retrieve data from tables](http://www.w3resource.com/sql-exercises/sql-retrieve-from-table.php)
2. [Query on Multiple Tables](http://www.w3resource.com/sql-exercises/sql-exercises-quering-on-multiple-table.php)
3. [SQL Exercises, Practice, Solution – JOINS](http://www.w3resource.com/sql-exercises/sql-joins-exercises.php)

**JS Essential**

An Introduction to JavaScript. Basic rule syntax and JS Style Guide. defer/async. Function declaration vs expression. Loops. Rest & Spred. Lexical Environment. Callback functions. Closure. Array. JS and OOP.

Technically speaking, ECMAScript is a standard that was introduced in 1997. It tells companies that make browsers how exactly they should make JavaScript work in their respective browsers. “But,” you might ask, “JavaScript is JavaScript… Why would it work differently across browsers?”

 Success Criteria

1. What are ECMAScript and Javascript?
2. How to add JS in HTML code?
3. How to declare a variable in JS? What is the difference?
4. What is hoisting in JS?
5. What difference between primitive and non-primitive data types?
6. When to use a function declaration vs a function expression?
7. What is the context in JS? What does it point to?
8. Which methods can be used for setting context manually?
9. How to pass default parameter values for JS functions?
10. What is the “arguments” object?
11. What is the difference between Rest and Spread operator?
12. What benefits bring us the callback functions?
13. How do you understand closure in JS? How many scopes have each closure? Name and diff them.
14. What is the Array in JS? In which cases it is better to use an Array and in which Object?
15. Name the popular methods in Array?
16. How OOP is implemented in JS?

 References

1. [ECMAScript vs JS](https://www.freecodecamp.org/news/whats-the-difference-between-javascript-and-ecmascript-cba48c73a2b5)
2. [History of ECMAScript](https://www.benmvp.com/blog/learning-es6-history-of-ecmascript)
3. [Importing JS](https://www.digitalocean.com/community/tutorials/how-to-add-javascript-to-html)
4. [JS variables](https://www.programiz.com/javascript/variables-constants)
5. [Variable scope](https://www.programiz.com/javascript/variable-scope)
6. [Hoising](https://www.programiz.com/javascript/hoisting)
7. [Data types](https://www.programiz.com/javascript/data-types)
8. [Function declaration vs expression](https://javascript.info/function-expressions)
9. [Context](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/this?retiredLocale=uk)
10. [Call vs apply vs bin](https://betterprogramming.pub/when-to-use-bind-call-and-apply-in-javascript-1ae9d7fa66d5)
11. [Function parameters](https://www.smashingmagazine.com/2016/07/how-to-use-arguments-and-parameters-in-ecmascript-6)
12. [Callbacks](https://www.programiz.com/javascript/callback)
13. [Closures](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures?retiredLocale=uk)
14. [Arrays](https://www.w3schools.com/js/js_arrays.asp)
15. [OOP](https://dev.to/nehal_mahida/oops-in-javascript-with-easy-to-understand-examples-2ppn)

**Working with DOM, BOM**

Browser environment: DOM, BOM. Attributes and properties. Select, modifying and creating an element. NodeLists and HTMLCollections. Styles and classes. Objects: Window, Location, Navigator, History, Screen.

In the previous modules, we were considering how browser renders website. In this module, you will acknowledge how browser rendering works — behind the scenes.

 Success Criteria

1. What is DOM? How does it look?
2. What is BOM?
3. How does JS interact with DOM using DOM API?
4. What is the difference between an HTMLCollection and a NodeList?
5. How to manipulate CSS from JS?

 References

* 1. [DOM API](https://web-fundamentals.dev/dom-api)
  2. [DOM BOM](https://medium.com/@amritgangwar/javascript-dom-and-bom-5f563fa17b64)
  3. [DOM prop erties and attributes](https://javascript.info/dom-attributes-and-properties)
  4. [HTM Collection vs NodeList](https://www.w3schools.com/js/js_htmldom_nodelist.asp)
  5. [CSSOM](https://css-tricks.com/an-introduction-and-guide-to-the-css-object-model-cssom)

**ES6**

ES6 is Ecmascript 2015,6th new version of Javascript Programming Language. ES6 is considered to be the major upgrade to javascript since ES5 was introduced in 2009. With the introduction of ES6, whole new features are added to the language like modules, iterators, class, arrow functions, for...of loop, promises, and many more.

 Success Criteria

1. What is a module?
2. What is export default es6?
3. How to import module dynamically?
4. What is a class in es6?
5. Which keywords can be used to implement inheritance in es6?
6. What are static properties and methods?
7. What new primitives have been added in ES6?
8. What is the arrow function, and how to create it?
   * Give an example of an Arrow function in ES6? List down its advantages.
9. Discuss spread operator in ES6 with an example. Discuss the Rest parameter in ES6 with an example.
10. Discuss Destructuring Assignment in ES6.
11. What are the template literals in ES6?
12. What are default parameters?
13. What is block scope in ES6?
    * Define let and const keywords.
14. What is Iterable object.
    * What is the difference between for...in and for...of loops?
15. What do you understand by the Generator function?
16. What are Promises in ES6?
    * What are the states of promises in ES6?
    * How to handle errors with promises?
17. What is async/await?
18. What are Map and Set in ES6?
19. What are WeakMap and WeakSet in ES6?
20. What are Proxy and Reflect in ES6?

 References

1. [Modules introduction](https://javascript.info/modules-intro)
2. [Export and Import](https://javascript.info/import-export)
3. [Dynamic imports](https://javascript.info/modules-dynamic-imports)
4. [Class basic syntax](https://javascript.info/class)
5. [Class inheritance](https://javascript.info/class-inheritance)
6. [Static properties and methods](https://javascript.info/static-properties-methods)
7. [Primitive](https://developer.mozilla.org/en-US/docs/Glossary/Primitive)
   * [Symbol](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Symbol)
   * [BigInt](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/BigInt)
8. [Arrow functions the basics](https://javascript.info/arrow-functions-basics)
   * [Advanced working with functions](https://javascript.info/arrow-functions)
9. [Rest parameters and spread syntax](https://javascript.info/rest-parameters-spread)
10. [Destructuring assignment](https://javascript.info/destructuring-assignment)
11. [JavaScript Template Literals](https://www.w3schools.com/js/js_string_templates.asp)
12. [Default parameters](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Default_parameters)
13. [Variable scope closure](https://javascript.info/closure)
    * [Keyword let](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let)
    * [Keyword const](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const)
14. [Iterables](https://javascript.info/iterable)
15. [Generators](https://javascript.info/generators)
16. [Promise](https://javascript.info/promise-basics)
    * [Promises chaining](https://javascript.info/promise-chaining)
    * [Promise API](https://javascript.info/promise-api)
    * [Error: handling with promises](https://javascript.info/promise-error-handling)
17. [Async/await](https://javascript.info/async-await)
18. [Map and Set](https://javascript.info/map-set)
19. [WeakMap and WeakSet](https://javascript.info/weakmap-weakset)
20. [Proxy and Reflect](https://javascript.info/proxy)

## Tasks

1. We have a “regular” function called f. How can you call the async function wait() and use its result inside of f?
2. async function wait() {
3. await new Promise(resolve => setTimeout(resolve, 1000));
4. return 10;
5. }
6. function f() {
7. // ...what should you write here?
8. // we need to call async wait() and wait to get 10
9. // remember, we can't use "await"
10. }

1. We'd like to get an array of map.keys() in a variable and then apply array-specific methods to it, e.g. .push.  
     
   But that doesn't work:
3. let map = new Map();
4. map.set("name", "John");
5. let keys = map.keys();
7. // Error: keys.push is not a function
8. keys.push("more");

Why? How can we fix the code to make keys.push work?

1. Create a function makeObservable(target) that “makes the object observable” by returning a proxy.  
     
   Here's how it should work:
2. function makeObservable(target) {
3. /\* your code \*/
4. }
6. let user = {};
7. user = makeObservable(user);
9. user.observe((key, value) => {
10. alert(`SET ${key}=${value}`);
11. });

user.name = "John"; // alerts: SET name=John

In other words, an object returned by makeObservable is just like the original one, but also has the method observe(handler) that sets handler function to be called on any property change. Whenever a property changes, handler(key, value) is called with the name and value of the property.

**Typescript**

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript. It is pure object-oriented with classes, interfaces and statically typed programming languages like C# or Java. You will need a compiler to compile and generate the code in the JavaScript file. Basically, TypeScript is the ES6 version of JavaScript with some additional features.

 Success Criteria

1. What are the Benefits of using TypeScript? What are the Disadvantages of TypeScript?
2. What is tsconfig.json file?
3. What is a TypeScript Map file?
4. What are modules in TypeScript?
5. Explain Relative and Non-relative module imports.
6. What is namespace in Typescript and how to declare it?
7. What is TypeScript Declare Keyword?
8. List out the built-in data types in TypeScript.
9. What is the difference between 'unknown' and 'any' type in TypeScript?
   * Explain the purpose of the never type in TypeScript.
10. What are Classes in TypeScript?
11. What are abstract classes? When should you use one?
12. What are Interfaces in TypeScript?
13. What are the 'implements' clauses in TypeScript?
14. What's the difference between 'extends' and 'implements' in TypeScript
15. What is Generic Type in TypeScript?
16. Explain how enums work in TypeScript?
17. What are union types? What are intersection types?
18. Explain the tuple types in TypeScript.
19. What are the access modifiers supported by TypeScript?
20. How to make object properties immutable in TypeScript?
21. How to extend Types? How to specify optional properties in TypeScript?
22. Explain different ways to declare variable
    * What is spread?
    * What is parameter destructuring?
    * What are the rest parameters and arguments in TypeScript?
23. What are type assertions in TypeScript?
24. How to enforce strict null checks in TypeScript?
25. Does TypeScript support function overloading?
26. Explain Decorators in TypeScript.

 References

1. [What is a tsconfig.json](https://www.typescriptlang.org/docs/handbook/tsconfig-json.html)
2. [TypeScript for the New Programmer](https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html)
3. [Emitting TypeScript Source Maps](https://www.carlrippon.com/emitting-typescript-source-maps/)
4. [Modules](https://www.typescriptlang.org/docs/handbook/modules.html)
5. [Module Resolution](https://www.typescriptlang.org/docs/handbook/module-resolution.html)
6. [Namespaces](https://www.typescriptlang.org/docs/handbook/namespaces.html)
7. [Type Declaration files](https://medium.com/jspoint/typescript-type-declaration-files-4b29077c43)
8. [Basic Types](https://www.typescriptlang.org/docs/handbook/basic-types.html)
9. [Classes](https://www.typescriptlang.org/docs/handbook/classes.html)
10. [Abstract Class](https://www.tutorialsteacher.com/typescript/abstract-class)
11. [Interfaces](https://www.typescriptlang.org/docs/handbook/interfaces.html)
12. [Generics](https://www.typescriptlang.org/docs/handbook/2/generics.html)
13. [Enums](https://www.typescriptlang.org/docs/handbook/enums.html)
14. [Unions and Intersection Types](https://www.typescriptlang.org/docs/handbook/unions-and-intersections.html)
15. [Tuples](https://www.tutorialsteacher.com/typescript/typescript-tuple)
16. [Data Modifiers](https://www.tutorialsteacher.com/typescript/data-modifiers)
17. [ReadOnly](https://www.tutorialsteacher.com/typescript/typescript-readonly)
18. [Object Types](https://www.typescriptlang.org/docs/handbook/2/objects.html)
19. [Variable Declaration](https://www.typescriptlang.org/docs/handbook/variable-declarations.html)
20. [Type Assertion in TypeScript](https://www.tutorialsteacher.com/typescript/type-assertion)
21. [Strict Null Checks](https://www.typescriptlang.org/tsconfig#strictNullChecks)
22. [More on Functions](https://www.typescriptlang.org/docs/handbook/2/functions.html)
23. [Decorators](https://www.typescriptlang.org/docs/handbook/decorators.html)

## Tasks

1. Create a method decorator @delay(ms) that delays each call of method by milliseconds specified.
2. class Greeter {
3. greeting: string;
4. constructor(message: string) {
5. his.greeting = message;
6. }
8. @delay(1000)
9. greet(): void {
10. console.log("Hello, " + this.greeting);
11. }
12. }
14. const obj = new Greeter();
15. obj.greet(); // should print to console with delay 1000 ms.

1. Using type declaration please extend Number interface by adding method "isEven(): boolean". Expose this method for intellisense so we could write the code as follow without compilation error.
2. const n = 5;

const isEven = n.isEven();

**Angular**

Angular was introduced to create Single Page applications. This framework brings structure and consistency to web applications and provides excellent scalability and maintainability. Angular is an open-source, JavaScript framework wholly written in TypeScript. It uses HTML's syntax to express your application's components clearly.

 Success Criteria

1. What is Angular? Why was it introduced?
2. Why do we need compilation process?
   * What is JIT?
   * What is AOT?
   * What are the advantages with AOT?
   * What are the three phases of AOT?
3. What is a module?
4. What is a bootstrapping module?
5. What are the types of feature modules?
6. What is the difference between a normal feature module and a lazy loaded module?
7. What is Angular Router?
   * What is router state?
   * What is activated route?
   * What is router outlet?
   * What are router links?
   * Do I need a Routing Module always?
   * How do you detect route change in Angular?
8. Explain string interpolation and property binding in Angular.
9. How do you categorize data binding types? How to implement two-way data binding?
10. What are Components in Angular?
11. What is dynamic component in Angular?
12. How does components interact with each other?
13. What is Content projection?
14. What are attribute directives in Angular?
15. What are structural directives in Angular?
16. What are the built-in directives in Angular?
17. What are Pipes in Angular? What is the difference between pure and impure pipe?
18. What is the purpose of async pipe?
19. What are Built-in Pipes in Angular?
20. What are lifecycle hooks available?
21. What is OnPush Change Detection strategy?
22. What are the ways to trigger change detection in Angular?
23. What does Angular use under the hood for change detection?
24. What are template driven forms?
25. What are reactive forms?
26. What are dynamic forms?
27. What is dependency injection in Angular?
    * What are providers in Angular?
    * How do you provide a singleton service?
    * How do you restrict provider scope to a module?
    * How do you restrict provider scope to a component?
    * What are resolution modifiers in Angular?
    * What is a DI token?

 References

1. [What is Angular?: https://angular.io/guide/what-is-angula](https://elearn.epam.com/courses/course-v1:RD_CEE+FEBfNL+0622/courseware/59b05564df764a5ebcb67c6cc70647fa/d8a207ecd7a24ec3afea7a32ceebacef/1?activate_block_id=block-v1%3ARD_CEE%2BFEBfNL%2B0622%2Btype%40vertical%2Bblock%40bba5fb666c884a1cbbdb9015cfde5eac)
2. [JIT and AOT Compilation in Angular](https://levelup.gitconnected.com/just-in-time-jit-and-ahead-of-time-aot-compilation-in-angular-8529f1d6fa9d)
3. [Introduction to modules](https://angular.io/guide/architecture-modules)
4. [Angular Modules and NgModule](https://blog.angular-university.io/angular2-ngmodule/)
5. [Summary of NgModule categories](https://angular.io/guide/module-types)
6. [Lazy-loading feature modules](https://angular.io/guide/lazy-loading-ngmodules)
7. [Common Routing Tasks](https://angular.io/guide/router)
   * [Router Reference](https://angular.io/guide/router-reference)
8. [Text interpolation](https://angular.io/guide/interpolation)
9. [Property binding](https://angular.io/guide/property-binding)
10. [Angular Components Overview & Examples](https://www.tektutorialshub.com/angular/angular-component/)
11. [Dynamic component loader](https://angular.io/guide/dynamic-component-loader)
12. [Component interaction](https://angular.io/guide/component-interaction)
13. [Content projection](https://angular.io/guide/content-projection)
14. [Attribute directives](https://angular.io/guide/attribute-directives)
15. [Structural directives](https://angular.io/guide/structural-directives)
16. [Built-in directives](https://angular.io/guide/built-in-directives)
17. [Transforming Data Using Pipes](https://angular.io/guide/pipes)
18. [AsyncPipe](https://angular.io/api/common/AsyncPipe)
19. [Built-in Pipes](https://angular-book.dev/ch09-03-built-in-pipes.html)
20. [Lifecycle: hooks](https://angular.io/guide/lifecycle-hooks)
21. [Angular OnPush Change Detection and Component Design](https://blog.angular-university.io/onpush-change-detection-how-it-works/)
22. [Angular lifecycle: hooks](https://indepth.dev/posts/1494/complete-guide-angular-lifecycle-hooks)
23. [NgZone](https://angular.io/guide/zone)
24. [Introduction to forms in Angular](https://angular.io/guide/forms-overview)
25. [Reactive forms](https://angular.io/guide/reactive-forms)
26. [Building dynamic forms](https://angular.io/guide/dynamic-form)
27. [Dependency injection in Angular](https://angular.io/guide/dependency-injection)
    * [Dependency injection in action](https://angular.io/guide/dependency-injection-in-action)
    * [Hierarchical injectors](https://angular.io/guide/hierarchical-dependency-injection)

## Tasks

Work out examples in these pages:

* <http://angular.io/start>
* <http://angular.io/start/start-routing>
* <http://angular.io/start/start-data>
* <http://angular.io/start/start-forms>
* <http://angular.io/start/start-deployment>

HTML Essential Training

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fhtml-essential-training-4%3Ftrk%3Dshare_ent_url%26shareId%3DOANYN17PSRyfc2V5E08ovw%253D%253D>

CSS Essential Training

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fcss-essential-training-3%3Ftrk%3Dshare_ent_url%26shareId%3D1ZgW6MrjTAujPpILoipGgw%253D%253D>

JavaScript Essential Training

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fjavascript-essential-training%3Ftrk%3Dshare_ent_url%26shareId%3DeoFeUjPvQfikaRgdfZ1IBw%253D%253D>

TypeScript Essential Training [Optional](Learning)

Angular Essential Training [Optional] (Learning)

Net Advanced

Advanced C#: Functional Programming Patterns

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fadvanced-c-sharp-functional-programming-patterns%3Ftrk%3Dshare_ent_url%26shareId%3Do0eXrYYWR8ic6%252FSbE6xs%252BQ%253D%253D>

Software Architecture: Patterns for Developers

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fsoftware-architecture-patterns-for-developers%3Ftrk%3Dshare_ent_url%26shareId%3DmgxzlBNUSaKVjQ4ntNolXw%253D%253D>

C# Algorithms

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fc-sharp-algorithms%3Ftrk%3Dshare_ent_url%26shareId%3DDTDfeCqwRm2jZFjEmFA65A%253D%253D>

Using Docker and .NET Core

<https://www.linkedin.com/learning-login/share?account=2113185&forceAccount=false&redirect=https%3A%2F%2Fwww.linkedin.com%2Flearning%2Fusing-docker-and-dot-net-core%3Ftrk%3Dshare_ent_url%26shareId%3D9RwlZmFlQtS4O1i4mzFjqA%253D%253D>

**Clouds**

**Public Cloud Platforms**