


.NET

# Brand guidelines



The purpose of these guidelines is to provide a framework for communicating with the .NET developer community and establishing a consistent brand identity. This document serves as a guide and reference to designers, writers, and developers to create consistent, on-brand content for .NET.

# Contents

## 04 Story

- 06 Tone of voice
- 07 Audience
- 08 Terminology
- 10 Referring to .NET

## 11 Core elements

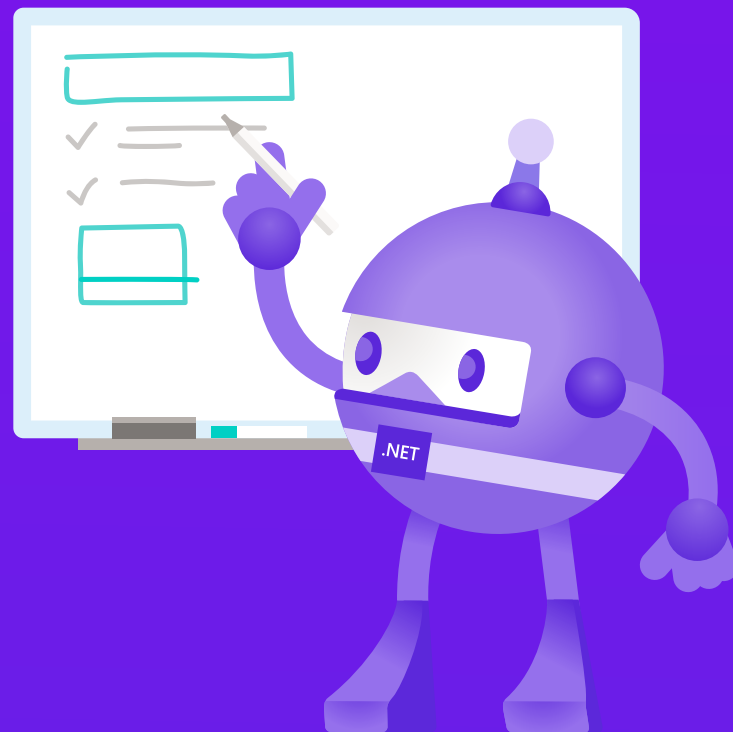
- 13 Logo
- 14 App models
- 15 Logo use
- 16 Colors
- 18 Type
- 20 Photography
- 22 Illustration
- 23 dotnet-bot

## 30 Examples


- 32 Twitter
- 33 Twitch
- 34 Conference poster and badge



# 01



# Story



.NET is a free, cross-platform, open source developer platform for building many different types of applications. With .NET, you can use multiple languages, editors, and libraries to build for web, mobile, desktop, gaming, and IoT.

.NET is designed to be general-purpose so that once a developer learns how to build one type of application, they can quickly build other types of applications for completely different workloads – developer skills can easily transfer to other domains using the same languages and libraries. .NET enables developers to be more productive building feature-rich applications of any type, for any device with great performance.

# Tone of voice

The .NET community is a large, global, diverse group of people from many backgrounds and cultures. When we speak with one another and the broader community we adhere to the following principles.

## .NET developers are:

Approachable

Helpful

Authentic

Clear-spoken

Good-natured

Inclusive

## .NET developers are not:

Arrogant

Dismissive

Fake

Ambiguous

Mean

Exclusive



# Audience

## New and emerging developers

New .NET developers should understand, in the simplest terms, that there are many types of applications they can build across operating systems. There are many application frameworks to choose from and we want new users to learn the latest, modern, open source technologies.

For those learning .NET, choose C# as the programming language because it is most widely used across all applications they can build. If appropriate, teach in the context of editors and IDEs that provide developer productivity like Visual Studio, Visual Studio Code and Visual Studio for Mac.

For this audience, .NET

- is easy to learn and has a consistent API
- is open source and cross platform
- is a sought-after job skill
- has an active community and large ecosystem of libraries, components and tools
- enables developers to quickly build modern, feature rich applications for any operating system or device

## Existing .NET developers

When positioning .NET to existing developers it's important to convey the changes to .NET that were made to make it modern, cross platform and open source. No longer is .NET a Windows-only technology.

.NET has made significant improvements for cloud native development with hyper scale and high performance as well as innovations in mobile and IoT. It has expanded into modern technology areas so that we can bring existing developers forward into the future, while still supporting applications that were built decades ago.

For this audience, .NET

- is innovative and modern
- is open source and cross platform
- is fast and scalable
- has a constantly growing ecosystem and developer base
- has expanded into modern technology areas

## Technical decision makers

Technical decision makers are CTOs, CIOs, architects and technical leads within an organization. These leaders have to invest in a technology stack that will bring the best ROI for the business and the many applications they must support.

For this audience, .NET

- is trusted and secure and supported by Microsoft
- has a large developer base and easy to hire for
- is used by millions of businesses in many different industries worldwide
- enables developers to quickly build modern, feature rich applications for any operating system or device

# Terminology

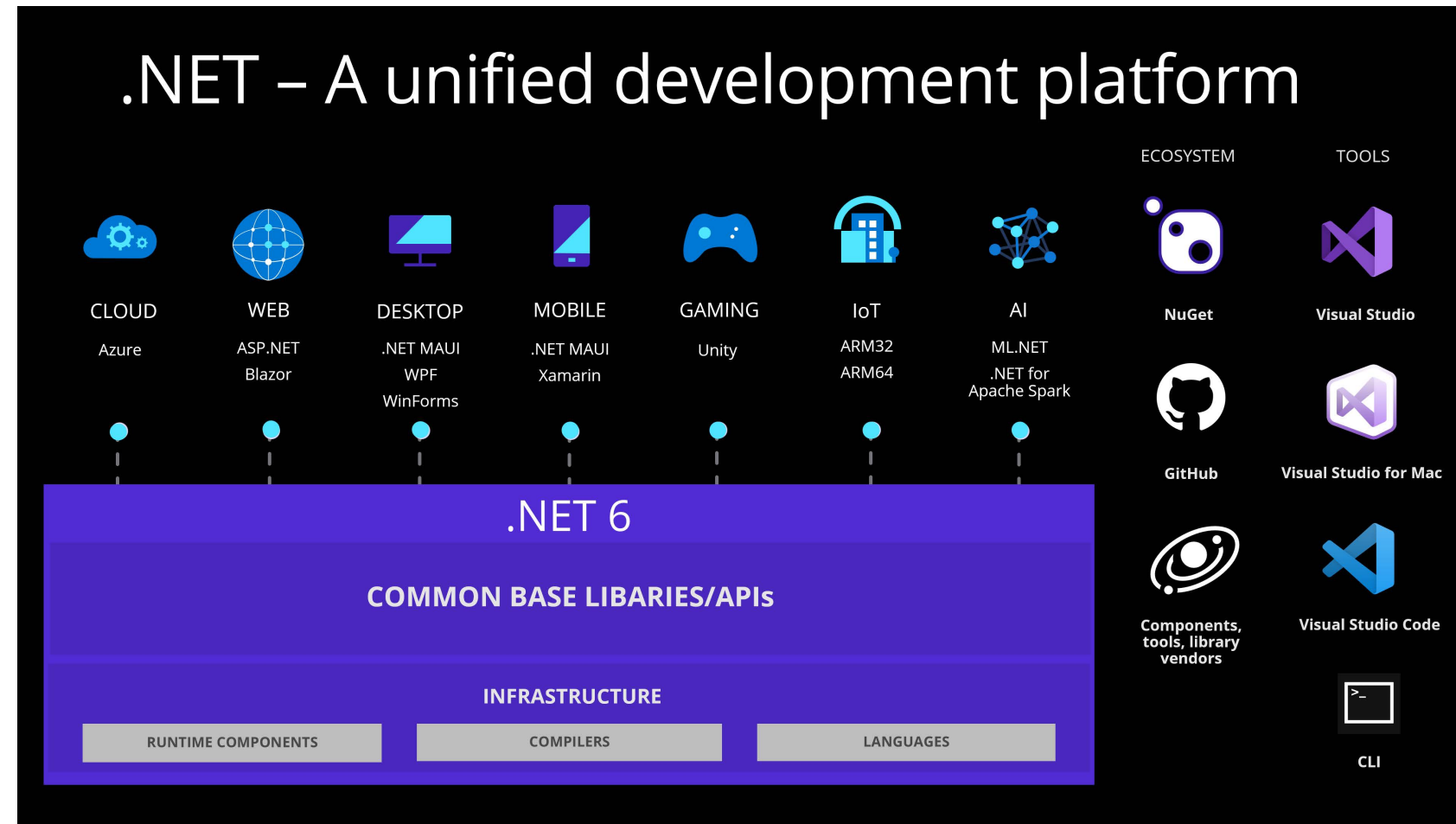
.NET is cross-platform and open source and enables developers to be more productive building feature-rich applications of any type, for any device with great performance. You can use the same tools, libraries, languages and runtime to build any application across web, mobile, desktop, AI/ML, data, gaming, and IoT workloads. .NET is a general-purpose programming platform that enables all kinds of application scenarios. Once you learn one, you can easily pick up another. To enable specific app scenarios, app models (or application frameworks) are built on top of the base libraries and runtime to extend what .NET can do. With .NET 6, there is a unified SDK, common base libraries, and runtime on which all of these app models sit.

.NET has a vast ecosystem of thousands of library packages via the NuGet package manager and open-source projects in development on GitHub. There are also hundreds of vendors providing additional components, tools, libraries and services. You can use a broad set of IDEs and editors with .NET including best-in-class tools with Visual Studio, Visual Studio for Mac and the cross-platform editor Visual Studio Code.

.NET was first released as .NET Framework in 2002 as a Windows-only developer platform and carries a long history of versions and libraries and runtimes (.NET Framework, .NET Core, Mono/Xamarin, etc.). When introducing .NET, do not focus on these implementations. Instead, talk about its languages, the workloads, and platforms it supports. As a developer gets deeper into workloads they want to build, then introduce the appropriate app models.

.NET is a world-wide, recognized brand name to developers and carries a long history with a certain perception. The brand strategy since 2014, when the platform was open sourced under the .NET Foundation, has been to change the perception of the .NET brand, not change the brand name itself.

Break the old perception that .NET is Windows-only, proprietary, and expensive. .NET is relevant and compelling to new & emerging developers, innovative and modern to existing .NET developers, and reliable, supported, and inexpensive to technical decision makers.





# Terminology

## Name usage

**.NET** – This is our brand name (ALWAYS CAPITALIZED) – this is the name to use anywhere across any implementation or workload.  
“.NET - Free. Cross-platform. Open source. A developer platform for building all your apps.”

**.NET version #** - Starting with .NET 5 released in November 2020, and onward, the platform name is just ".NET" and then the version number. New major versions release every November. This is the open source, cross-platform, high performance implementation of .NET and where investments in the platform are made.

**.NET Framework** – Windows-only implementation of .NET. The last major version was .NET Framework 4.8 released in April 2019. There should never be another word between these two words. Never say “.NET full framework” as this implies that other .NET implementations are subsets or incomplete. Never say “.NET classic framework” as this implies that current .NET investments are old and irrelevant when in actuality Microsoft will support .NET Framework as long as Windows exists. Unfortunately, this name describes an implementation but has the word “framework” in there. Because of its long history we must keep this name, but it can be clarified as “.NET Framework for Windows” for context.

**.NET Core** – This was the first cross-platform, high performance, open source implementation. The last version of .NET Core was 3.1. It is not a sub-set of .NET Framework. It is the basis of the current .NET developer platform.

**.NET Standard** is a specification that all implementations prior to .NET 5 adhered to in order to provide a standardized set of base class libraries that are supported across any older implementation be that .NET Framework, .NET Core, Mono. It is not needed for libraries targeting .NET 5 or higher.

See Referring to .NET for more example usage.

## Terminology applying across brand

These terms are typically used in documentation or technical content. Note these are purposely lower case.

**platform** for operating systems and chipsets (Windows, macOS, Linux) with distros for Linux flavors. Unless prefixed by “developer” which means a programming platform.

**implementation** for .NET “flavors” (.NET Framework, .NET Core, Mono). This describes the runtime, tools, libraries/frameworks for building different groups of applications. There are shared components across implementations (i.e. compilers, services, .NET Standard).

**framework** for APIs (.NET Framework base class library, .NET Core framework, Mono base class library). Libraries is OK here too depending on context.

**app model** for workload specific APIs (Windows Forms, ASP.NET, Blazor, etc.).

**workload** for a type of app someone is building. More generic than app model (Mobile, Desktop, IoT, Web, Games).

# Referring to .NET

Since moving the product development to the open under the .NET Foundation, .NET is no longer mentioned as “Microsoft .NET”, it’s just “.NET” or “The .NET Platform”.

## Correct

- .NET
- ASP.NET
- Microservices with .NET
- Game development with .NET
- ML .NET  
Note: There is a space between “ML” and “.NET” to avoid creating a URL
- .NET for Apache Spark
- .NET Core
- .NET Framework
- .NET Standard
- C#
- F#
- Visual Basic .NET on first reference, VB or Visual Basic after first reference
- .NET Multi-platform App UI on first reference, .NET MAUI after first reference
- Entity Framework on first reference, EF after first reference
- Windows Communication Foundation on first reference, WCF after first reference
- Windows Workf ow Foundation on first reference, WF after first reference
- Windows Presentation Foundation on first reference, WPF after first reference
- Windows Forms on first reference, WinForms after first reference

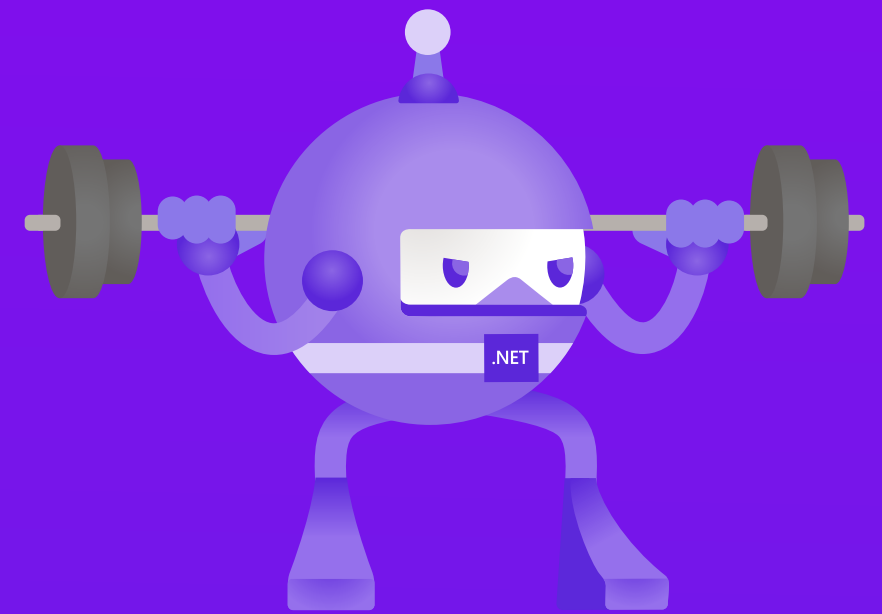
## Incorrect

Don’t spell out the .NET acronym within the names of technologies such as ASP.NET, in order to avoid redundancy.


- .Net
- .net
- Microsoft .NET
- .NET ASP.NET
- Microsoft ML.NET
- Microsoft .NET Xamarin
- Microsoft .NET Core
- .NET Desktop



# 02



## Core elements



Core elements in a design system allow for **flexible** and **dynamic** expression while still forming common ground through all brand collateral.

# Logo

The .NET logo is one of the most simple and direct visual representations of the .NET brand. This logo should be used in instances of official .NET communications.



## Use of logo

To help ensure the visibility of the logo, place it within a generous area of clear space so that it does not feel crowded by other graphic elements, copy, or page trim.

Minimum size: The symbol should never be smaller than 36 pixels tall on-screen and .5" (12.7 mm) tall in print.

The .NET logo and these brand guidelines can be found in the .NET Brand repo on GitHub ([github.com/dotnet/brand](https://github.com/dotnet/brand))



# App models

## App model branding guidelines

There are four strategies for branding app models:

1. App models with generic technologies in the name, like “Microservices with .NET,” use style one for sub-branding.
2. App models with specific third-party products, like “.NET for Apache Spark,” use style two.
3. App models that include “.NET” as a suffix, like “ASP.NET,” use style three. The first part of the app model is in light Open Sans, and “.NET” is in semi-bold.
4. App models that don’t include .NET in the name use style four in Open Sans light.

Examples of app models include:

- ASP.NET
- ASP.NET Web API
- Entity Framework or EF
- Windows Presentation Foundation or WPF
- Windows Communication Foundation or WCF
- Windows Workflow Foundation or WF
- Windows Forms or WinForms

## Usage

App models are branded using Open Sans Light and Semibold. .NET is always set in Open Sans Semibold. All other services, products, and languages are set in Open Sans Light.

Style 1

Microservices with  
**.NET**

Style 2

**.NET**  
for Apache Spark

Style 3

ASP.**NET**  
  
ML.**NET**

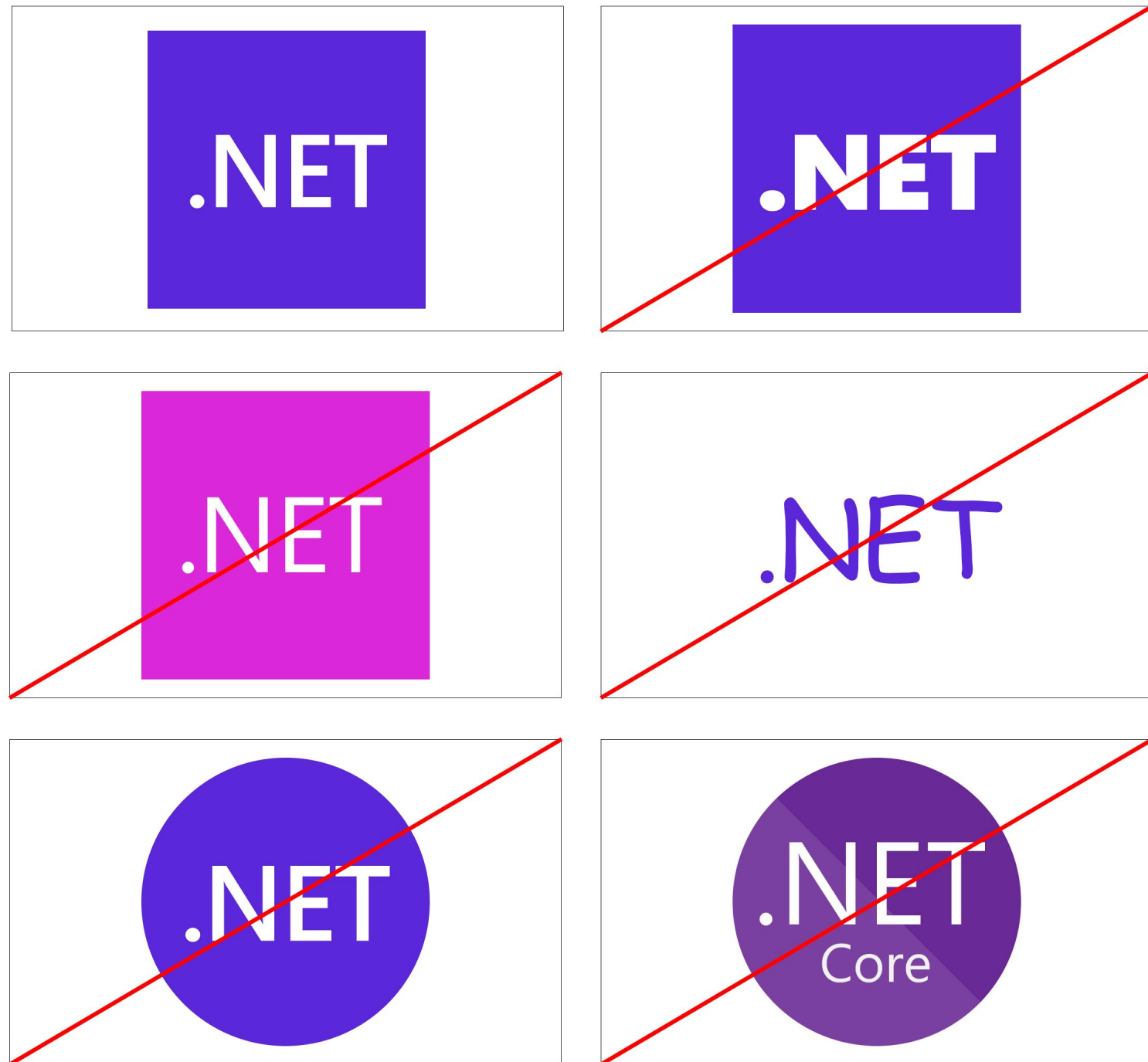
Style 4

WPF

# Logo use

The .NET logo is a core brand asset to foster immediate recognition. To ensure that the icon can provide a consistent and clear presence, it is critical not to modify the icon in any way. We use Open Sans Semibold for the logotype.

1. Do not color .NET type other than gray, white, or our purple (HEX #512BD4).
2. Do not change the font of the logotype.
3. Do not change the size relationship of any part of the logo.
4. Do not place in a shape other than what is specified in this document unless it is a constraint of the platform displaying the logo (i.e. Twitter places avatars in a circle.)
5. Do not use the old version of the icon that includes Microsoft above .NET.
6. Do not put another word or the sub-framework in the purple box.



Primary purple  
#512BD4

Primary gradient  
#7014E8 @ 9.04%  
#4931DE @ 100%

# Color

Our unique color palette helps people recognize .NET. It reveals our personality and reflects our diversity and wide range of offerings.

## Primary colors

The core color is the main purple. In addition to this color, there are corresponding lighter and darker colors. These three purples represent the distinguishing color set for .NET. Neutral colors are also used in certain settings.

Light purple  
#DFD8F7

White  
#FFFFFF

Black  
#000000

Dark purple  
#2B0B98

Light gray  
#E5E5E1

Mid gray  
#969696

Dark gray  
#505050



# Color

## Accent colors

To provide more flexibility, dynamic energy, and vibrancy to this palette, you can use these additional colors as accents to the purple hues. These accent colors should never be the dominant color.

|  |                                  |                                |                                |
|--|----------------------------------|--------------------------------|--------------------------------|
|  | <div>Yellow100<br/>#F7B548</div> | <div>Cyan100<br/>#28C2D1</div> | <div>Blue100<br/>#3E8EED</div> |
|  | <div>Yellow200<br/>#FFD590</div> | <div>Cyan200<br/>#7BDDEF</div> | <div>Blue200<br/>#72ACF1</div> |
|  | <div>Yellow300<br/>#FFE5B9</div> | <div>Cyan300<br/>#C3F2F4</div> | <div>Blue300<br/>#A7CBF6</div> |

# Type

The Open Sans typeface helps express the humanity in our voice and personality across all our communications. It's an essential, long-standing core brand element.

## Size & scale

Open Sans comes in five weights. However, for simplicity, focus on just two font weights:

Open Sans Semibold

Open Sans Regular

Occasionally, other font weights can be used when extra hierarchy or emphasis is needed.

The size relationship between body copy, sub-heads, and headlines is important. So when it comes to choosing type sizes for a design, try to use just three. This makes the contrast between them simple and establishes hierarchy.

## Open Sans

AaBbCcDdEeFfGgHhIijjKkLlMmNn  
OoPpQqRrSsTtUuVvWwXxYyZz

## H1 Open Sans Semibold

## H2 Open Sans Semibold

## B1 Open Sans Regular

## H3 Open Sans Semibold

## B2 Open Sans Regular

# Type

**After** the first word, all other words are lowercase, except proper nouns.

## Use sentence case

Default to sentence-style capitalization—capitalize only the first word of a heading or phrase. Also, skip end punctuation on titles, headings, subheads, UI titles, and items in a list that are three or fewer words.

Flush-left text is aligned along the left margin, also known as left-aligned or ragged right.

## Make it flush left

Generally, visual elements and type should be left-aligned. Text should never be full-justified and should very rarely be centered.

The quick brown fox jumps over the lazy dog. Few black taxis drive up major roads on quiet hazy nights. Jackie will budget for the most expensive zoology equipment.

All questions asked by five watched experts amaze the judge. Back in June we delivered oxygen equipment of the same size. My girl wove six dozen plaid jackets before she quit. Lovak won the squad prize cup for sixty big jumps. Who packed five dozen old quart jugs in my box?

## Enhance readability

Line spacing, or leading, refers to the space between lines of type. In general, headings are set between 100-120% and body copy is set at 150%. Tracking, or letter spacing, is the space between letters. For type smaller than 20pt, this should be set at “0”. All larger type should be set at -15. Avoid widows, orphans, and lines that end with hyphens.

# Photography

Photography is a visual element that greatly impacts our customers' understanding and impression of .NET. Our photography is human and relatable. Use images featuring real people in real situations as much as possible.







## Using photos

There are three main use categories for photography, each with a purpose:

- Excite to attract, entice, entertain, or motivate
- Inform to instruct, educate, or give context
- Connect to build rapport with audiences in an authentic, relatable way.



## Art direction

Feature diverse people. Highlight true personalities of people, their environments, and details. Think about how the images can work together, rather than trying to say everything in a single moment. Embrace authentic environments.



## Journalistic approach

Our method is to document people, their lives, and their experiences, rather than curating a typically staged, or device focused scenario. Focus on authentic moments, rich contexts, emotive color, interesting compositions, and dramatic scale, angles, and framing.

# Illustration

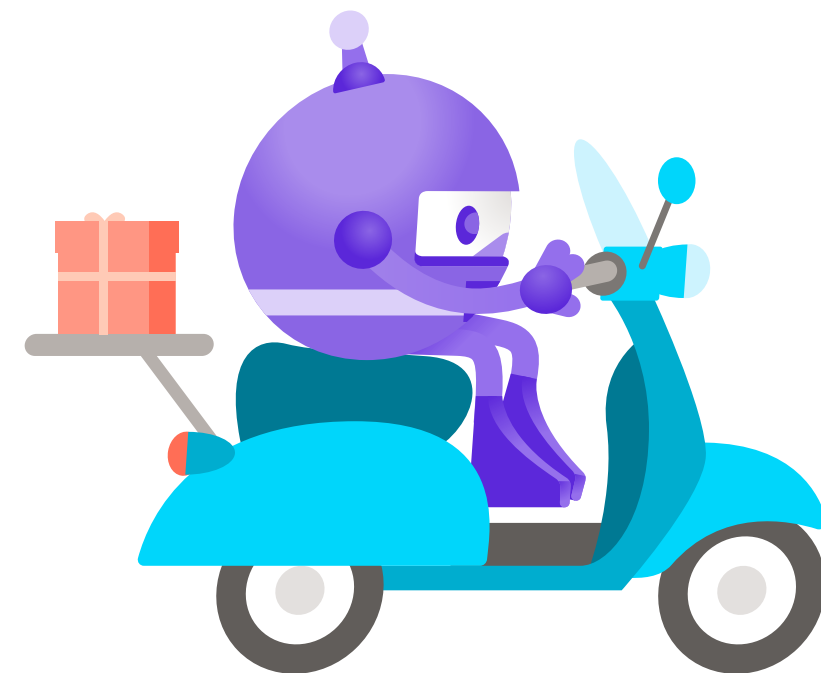
Our illustration is always purposeful, not decorative. To communicate complex and abstract ideas, we need visual tools that tell stories and provide metaphors. These illustrations are more than sketches, they're stories full of character, delightful surprises—and layers of meaning.

## Simplify the complex

The keys to our Illustration style is to make complex and abstract ideas clear with curated colors and geometric shapes, our illustration style feels simple and focuses on telling a story.

## Expression in metaphors

You can use the subject matter of an illustration as a way to get more or less expression. Example: You can use a more playful illustration as a metaphor for something like machine learning with ML .NET.

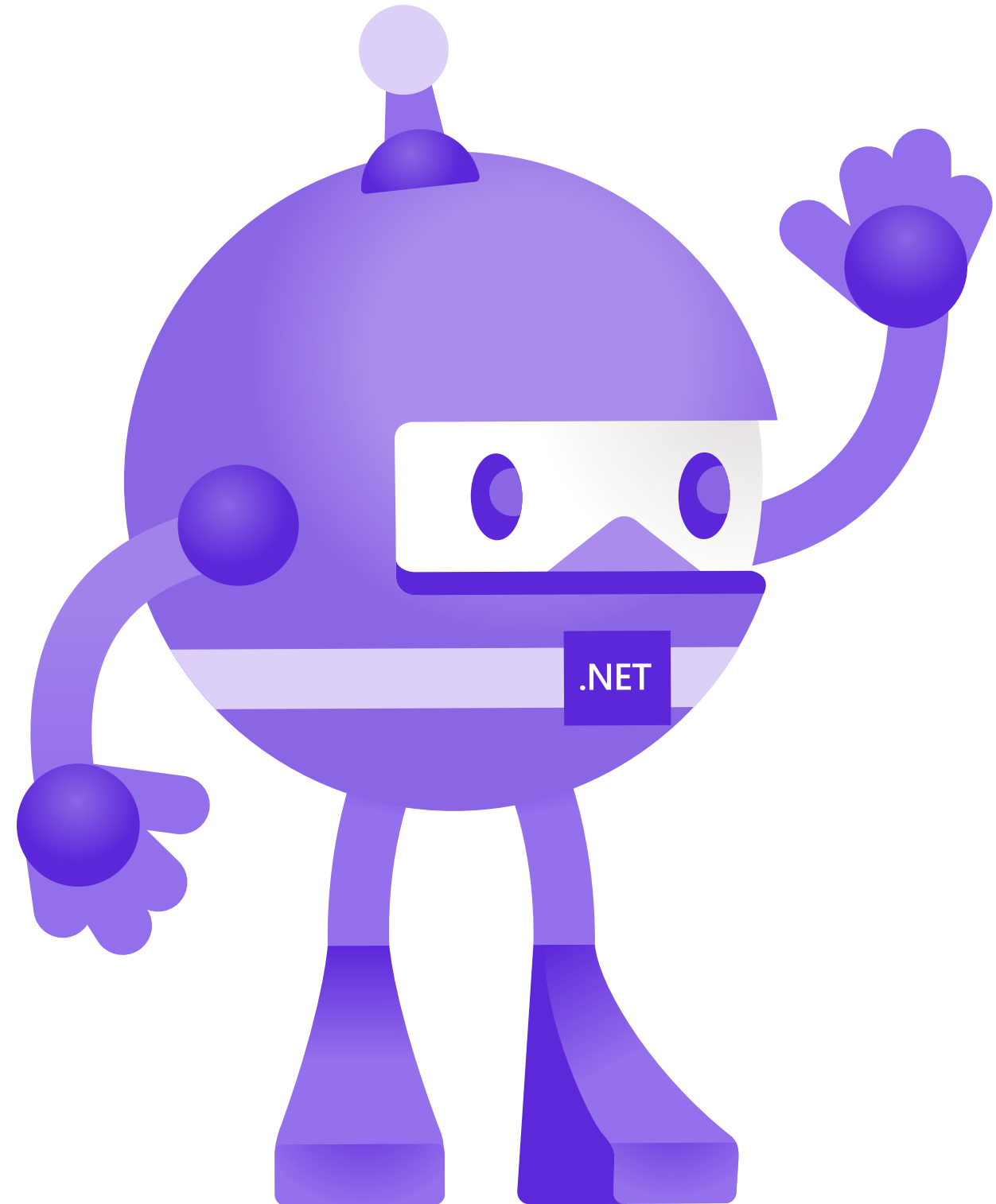


# dotnet-bot

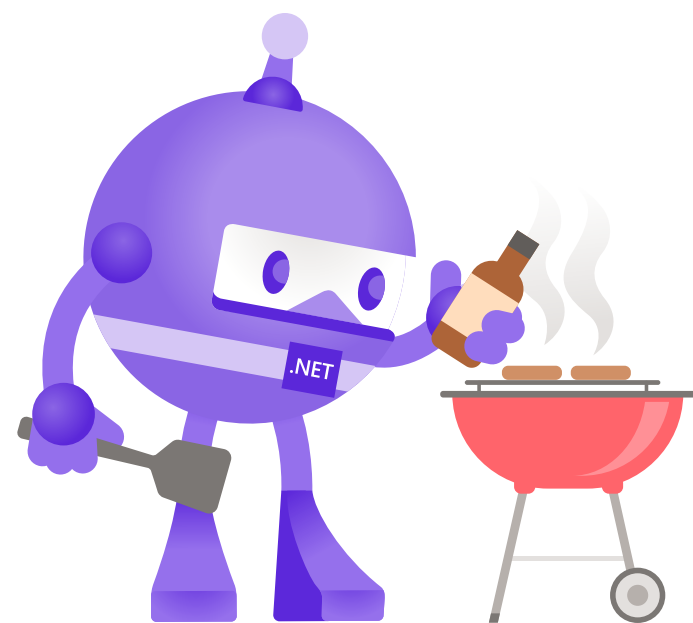
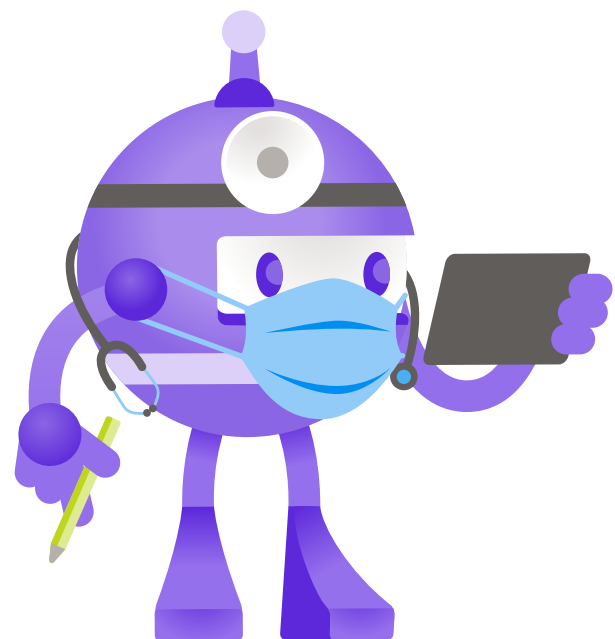
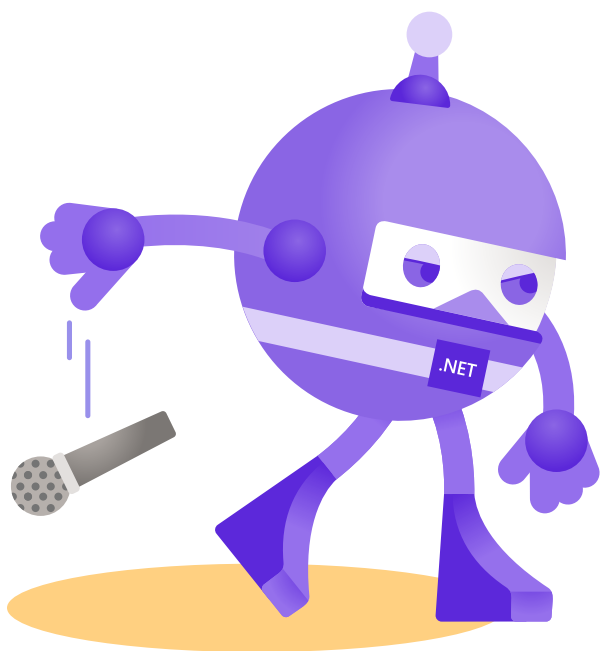
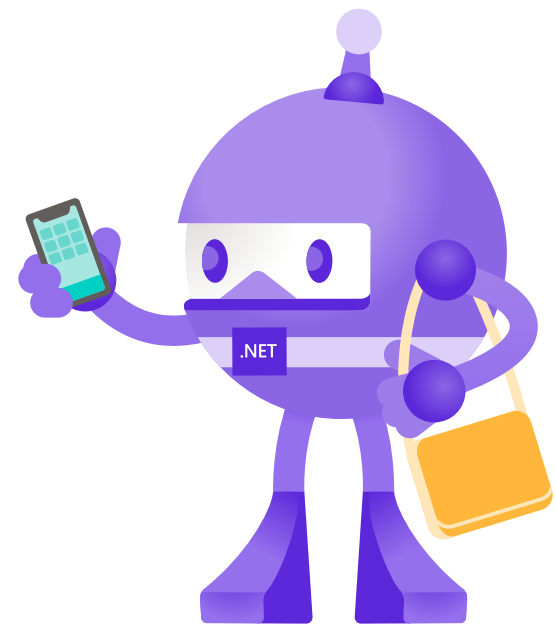
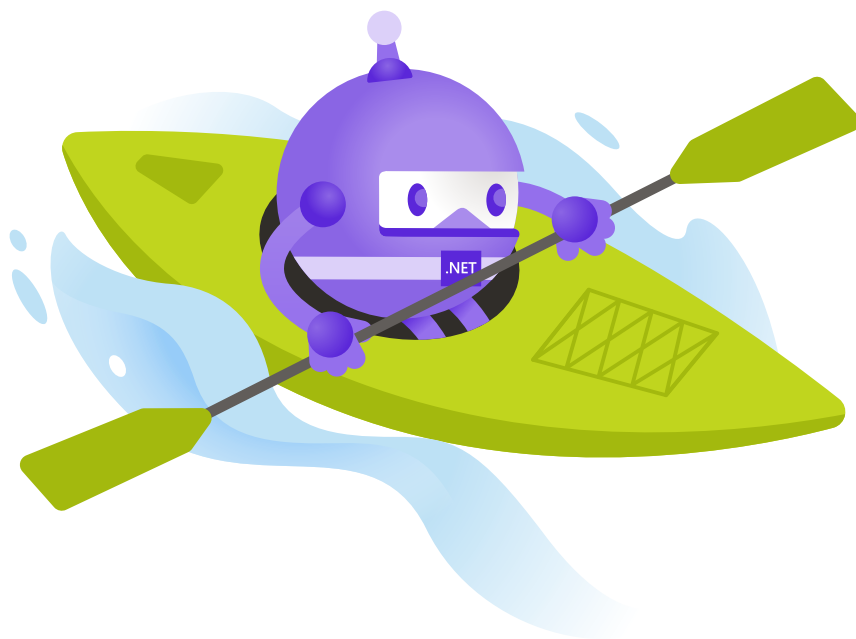
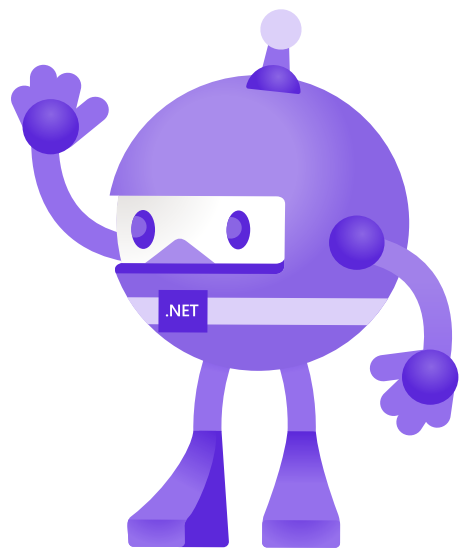
The dotnet-bot represents the community that comes with the .NET brand and platform. It also helps with checking pull-requests on .NET repos on GitHub.

When appropriate, it can be used on .NET communications. However, the bot is not the logo and should not be used instead of the logo.

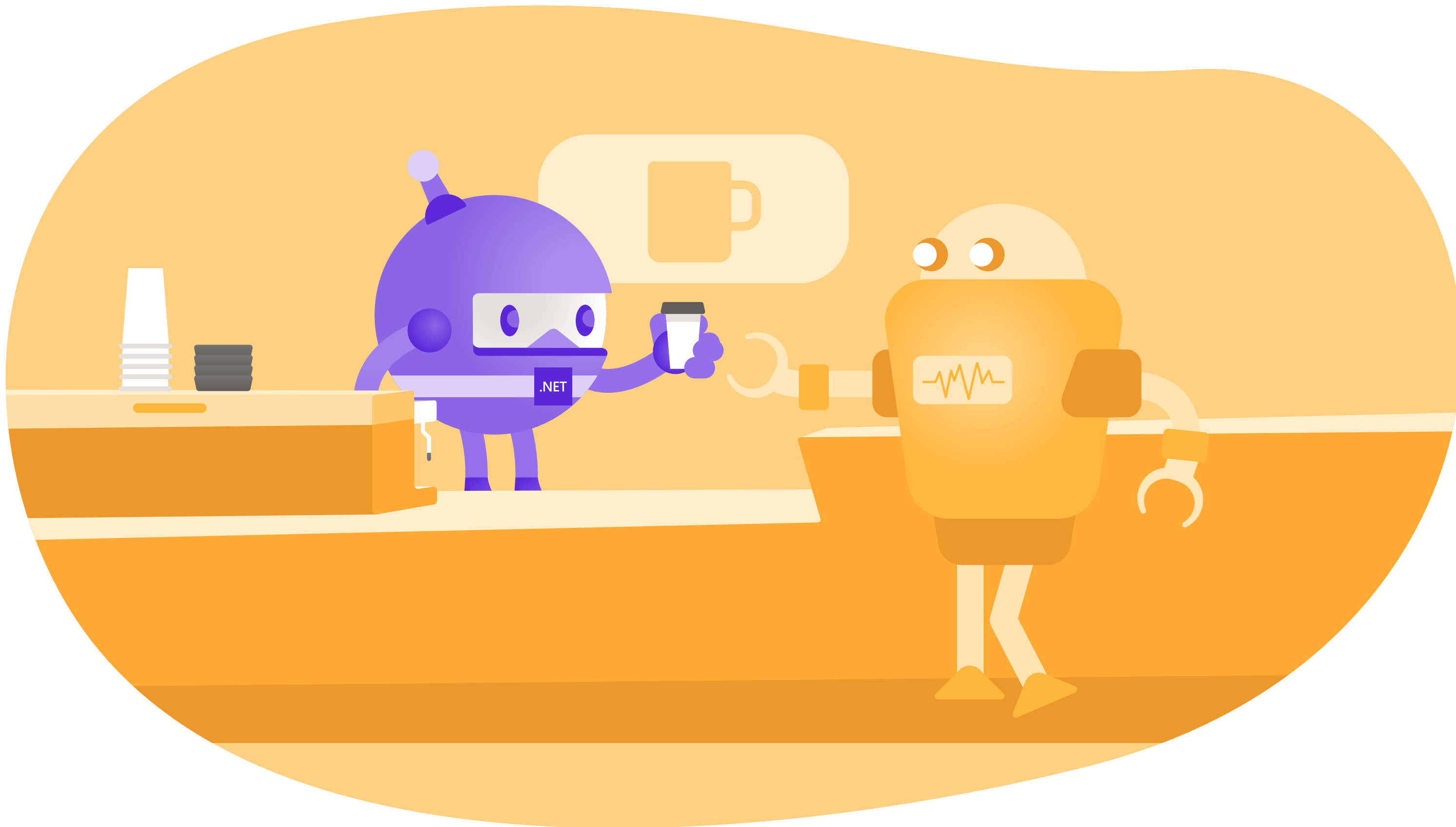
When using the dotnet-bot at larger sizes, use the dotnet-bot illustrations with the .NET logo on the belt. To avoid the .NET logo becoming illegible at smaller sizes, use one of the illustrations without the logo.



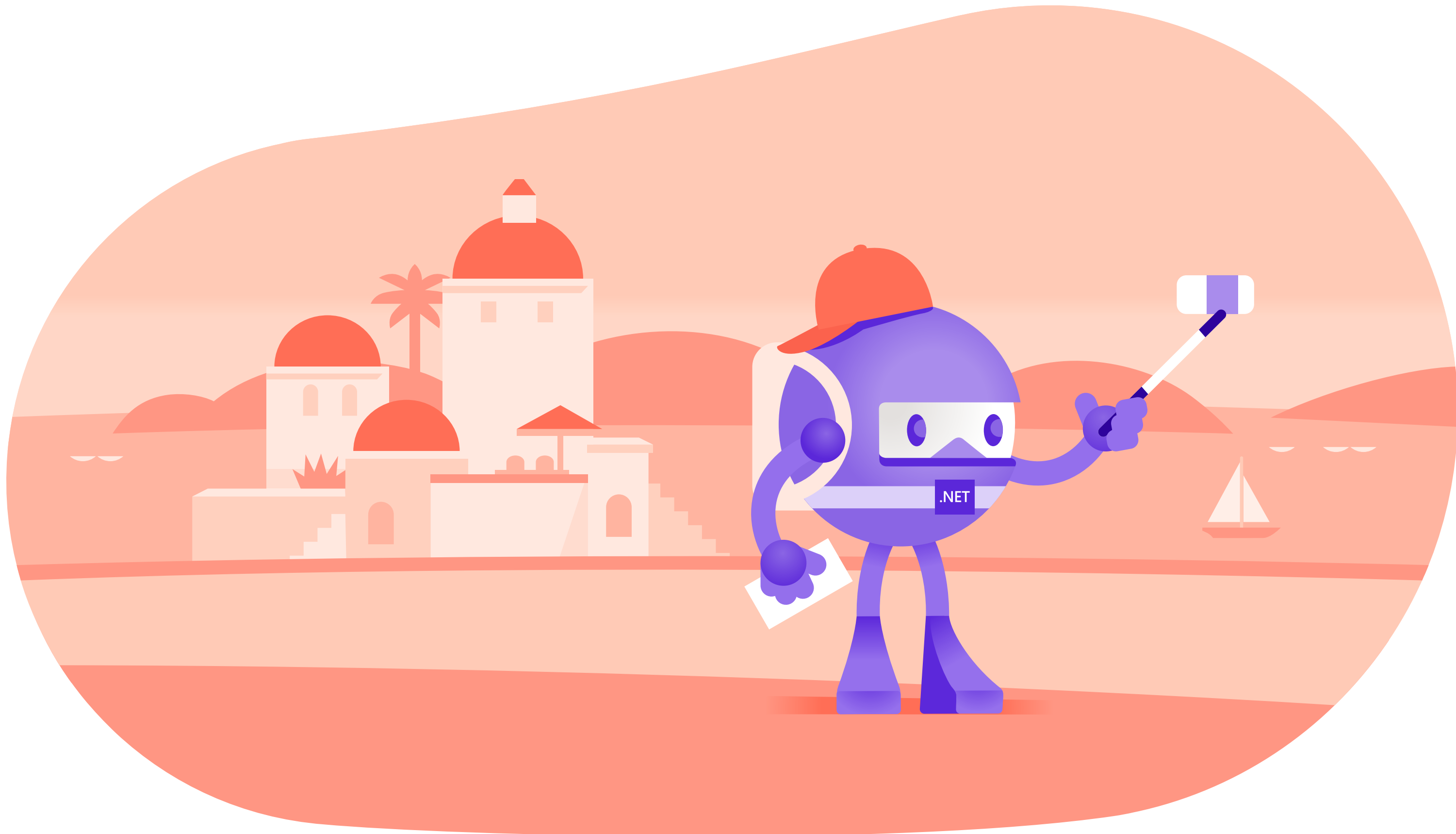












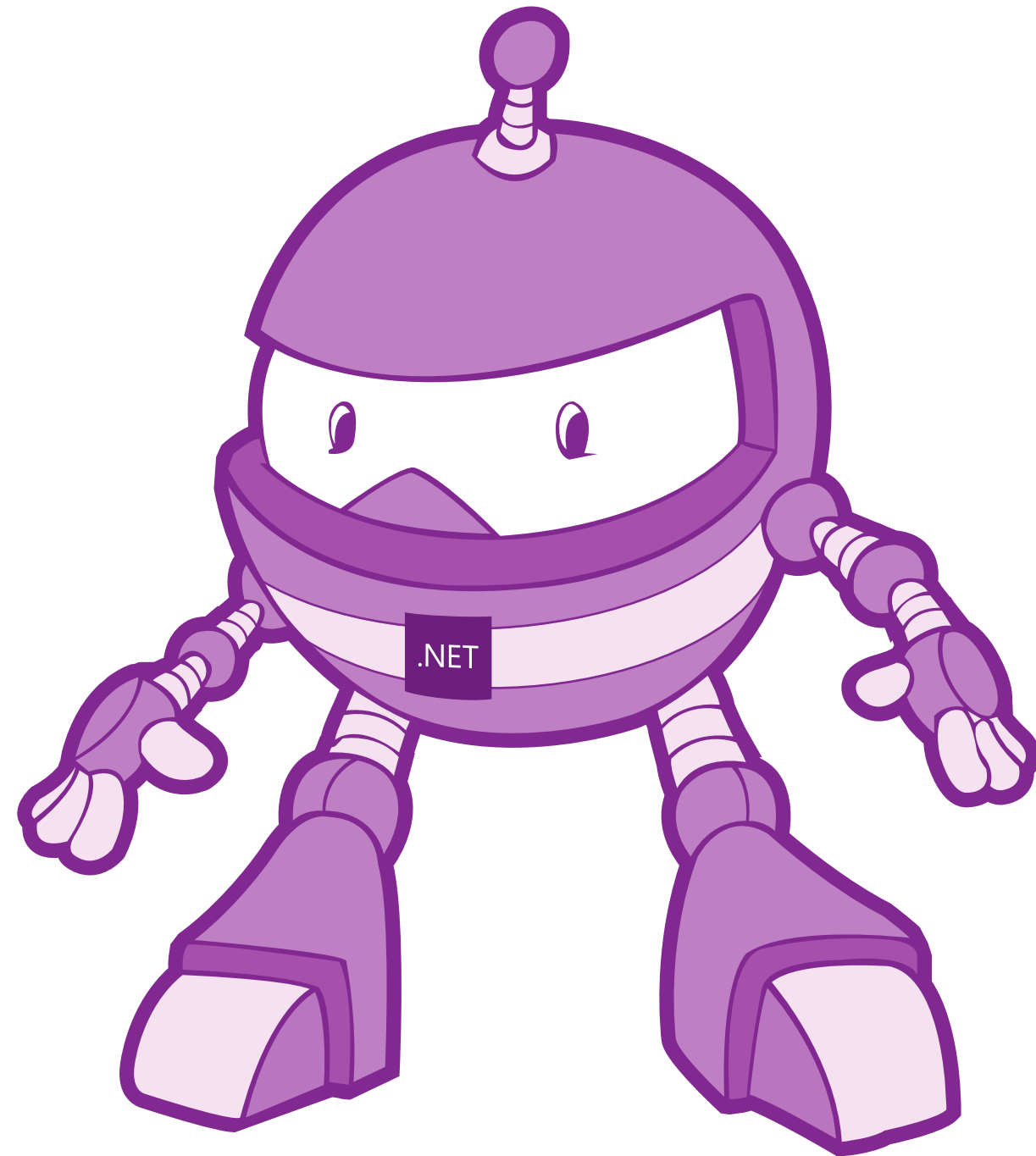


# Previous versions of dotnet-bot

Although we encourage everyone to use the new dotnet-bot design, the previous versions of the dotnet-bot can still be used by the community as long as the illustrations follow these principles:

- Include the .NET belt on dotnet-bot. For smaller sizes, if “.NET” on the belt becomes illegible, the text can be removed.
- You may use illustrations submitted by the community on [github.com/dotnet-foundation/swag](https://github.com/dotnet-foundation/swag)
- Do not position the dotnet-bot in any inappropriate places or forms.


Submit any new artwork you create for feedback and approval at [github.com/dotnet-foundation/swag](https://github.com/dotnet-foundation/swag)



# 03



# Examples



The following examples illustrate how to apply the .NET brand guidelines.





