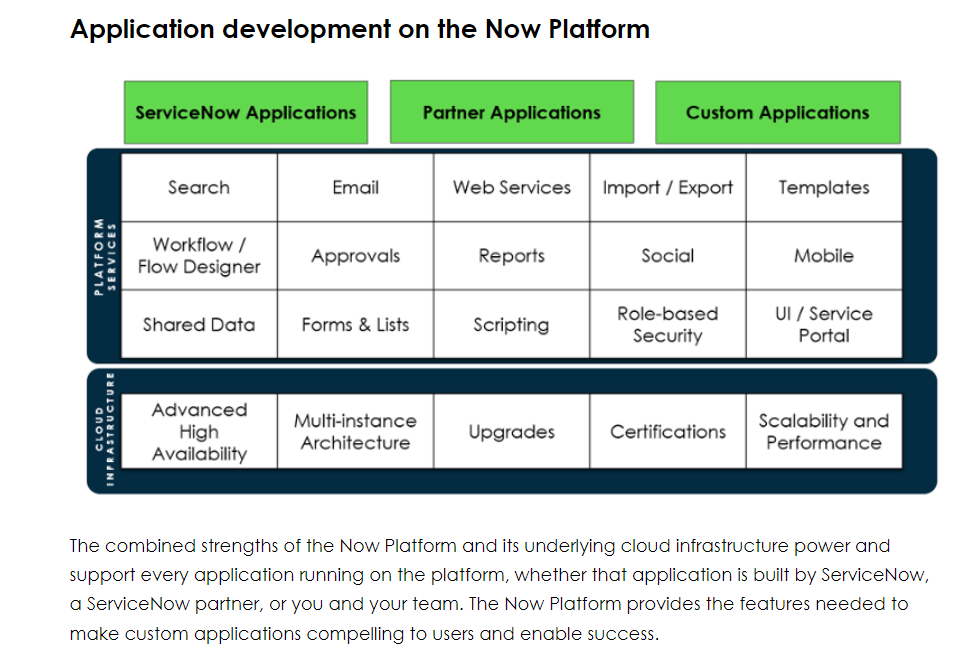
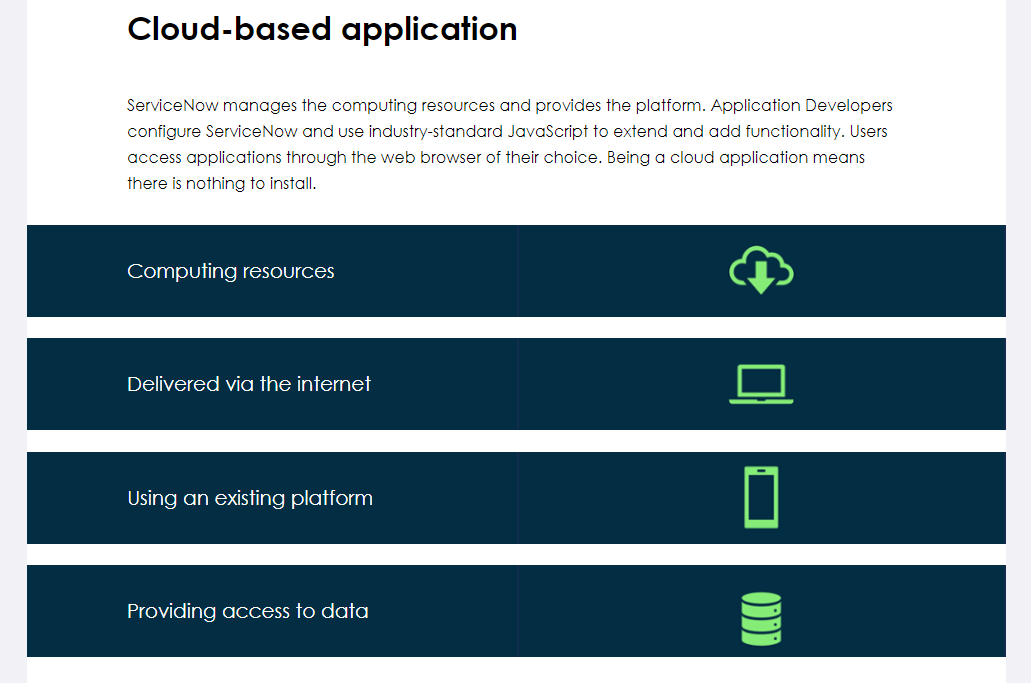
CAD NOTES





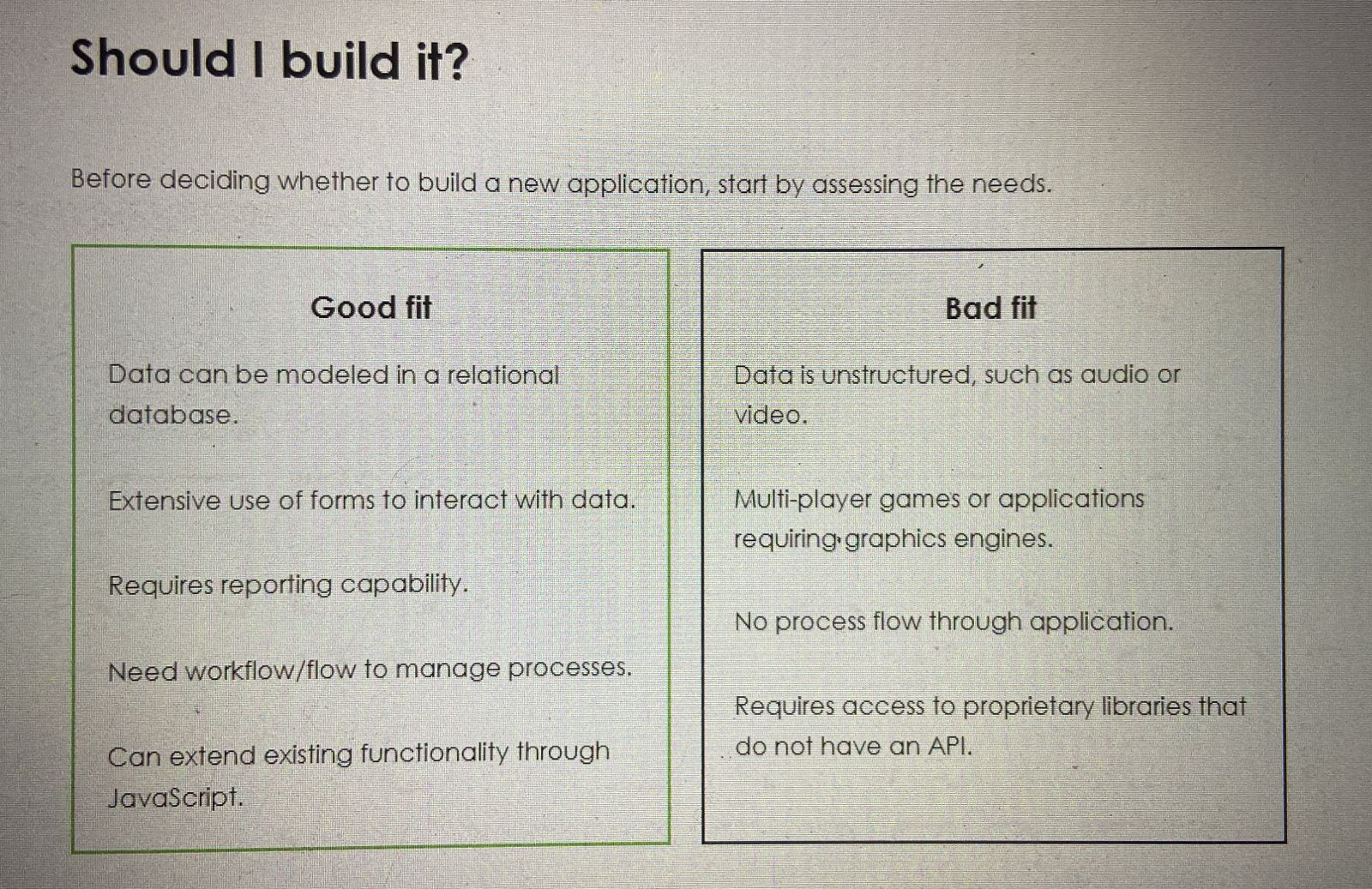
**Why develop custom applications?**

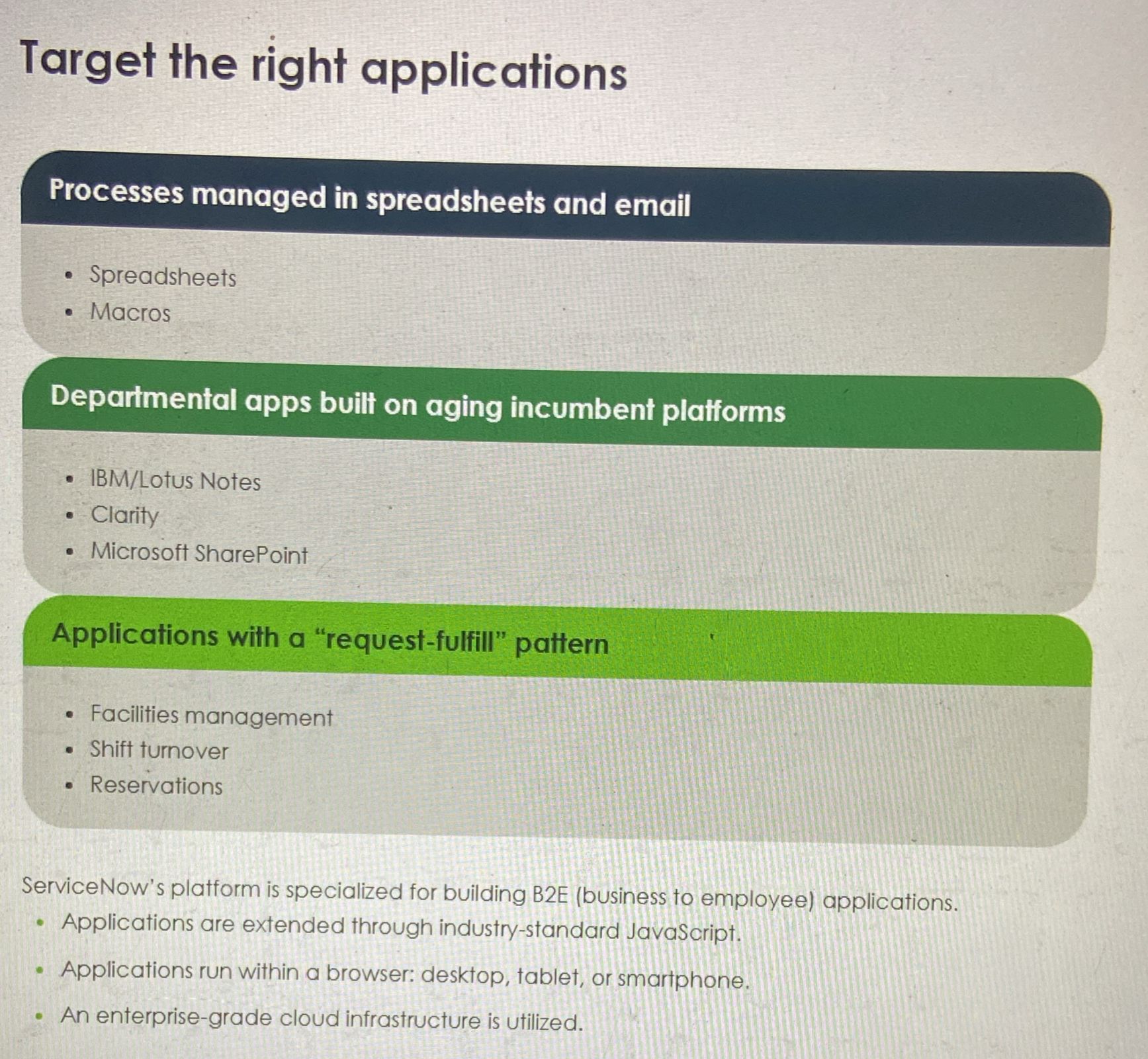
Custom applications are developed to:

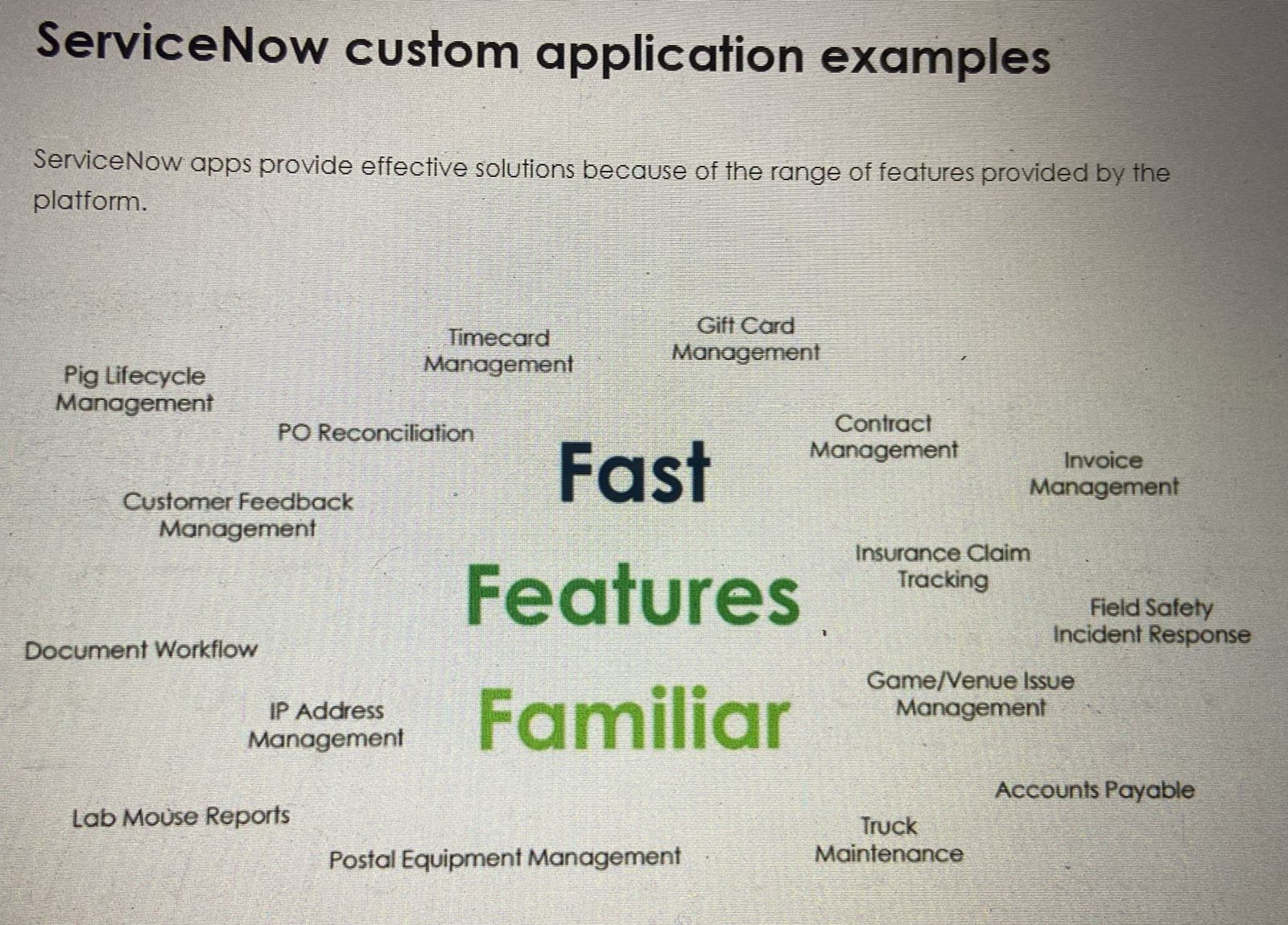
* Replace outdated, inadequate, custom business applications and processes.
* Extend the value of ServiceNow.
* Extend service delivery and management to all enterprise departments.
* Bring greater levels of automation and consolidation to enterprise services and their management.

Most enterprise decision makers do not realize how much of day-to-day business relies upon custom applications. Many of those applications only partially support business needs or user expectations.

Build effective ServiceNow applications that increase responsiveness, productivity, and user satisfaction.







**BackStory:**

Up to and including ServiceNow's Dublin release, all applications in ServiceNow were built in 'global" and were deployed using Update Sets. As the evolution progressed through the Eureka release, the concept of integrated application building (with Modules and Access Controls being built with the application) began to develop. Applications built at that time were built in the global scope, which meant problematic code in an application could impact other processes and applications.

Application scopes were introduced around the Fuji release (2014). Scoped applications allowed for discrete, modular development, with less impact on the overall aspects of the platform. With this in mind, some applications, such as HR and CSM were written (or re-written) as scoped applications. All new development should be done as a scoped application.

With the Geneva release, the integrated development environment, Studio, was incorporated to manage both scopes and general application development.

Studio and scoped application development/deployment have continued to be refined and improved with each subsequent release.

**Out-of-scope records**

All applications have scope.

**Global scope**

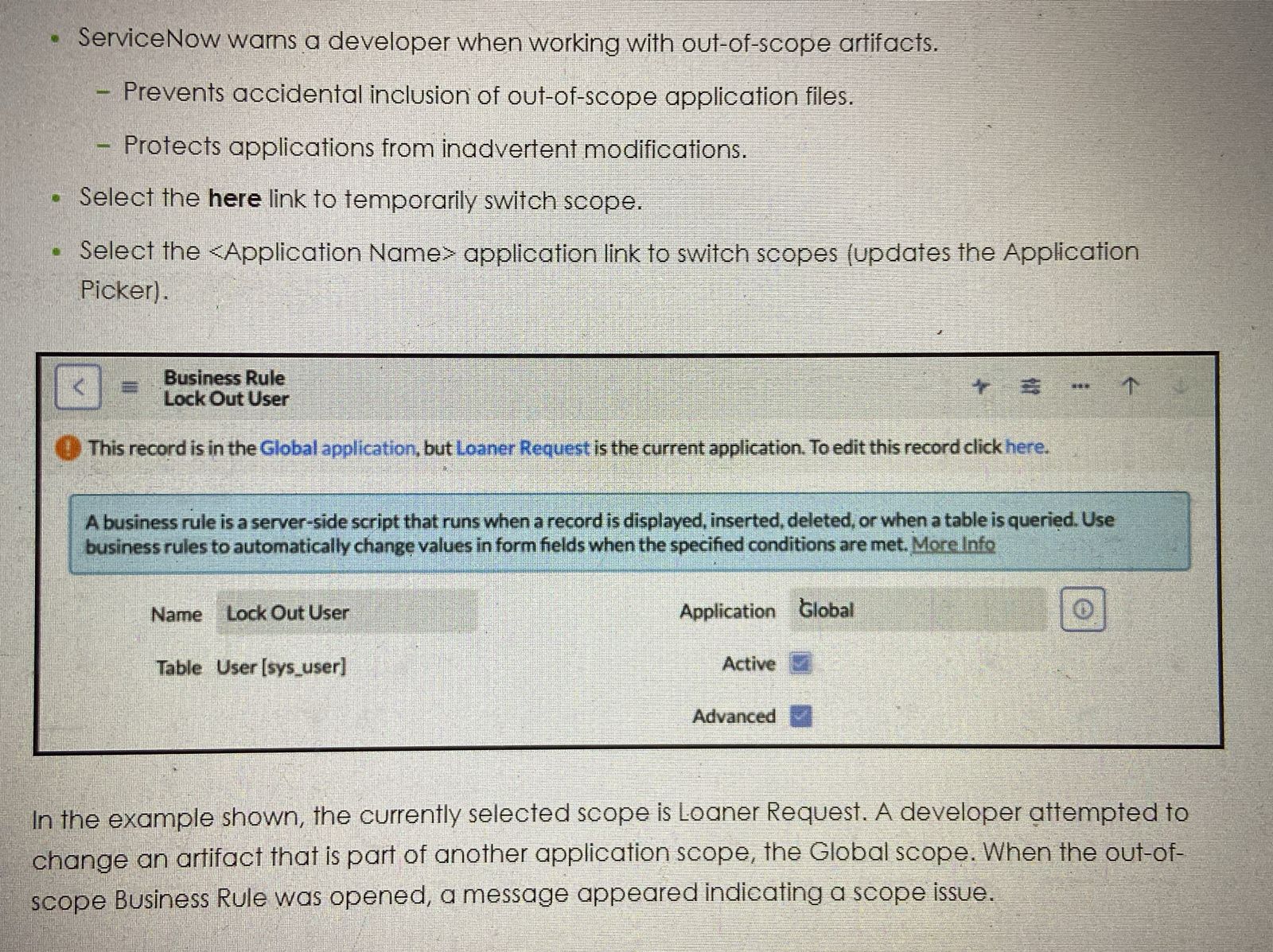
* Baseline applications.
* Custom applications built on ServiceNow versions prior to Scoping.

**Privately scope**

* Custom applications.

**Legacy custom applications will always be part of the global scope; there is no migration path to a custom or different scope.**

**It is not possible to change an application's scope once the record is saved to the database.**

****

**Scoped vs. Global**

**Scoped apps**

* Separate namespace
* Delegated Development
* Source Control integration
* Publish to app repository
* Easy file management

**Global apps**

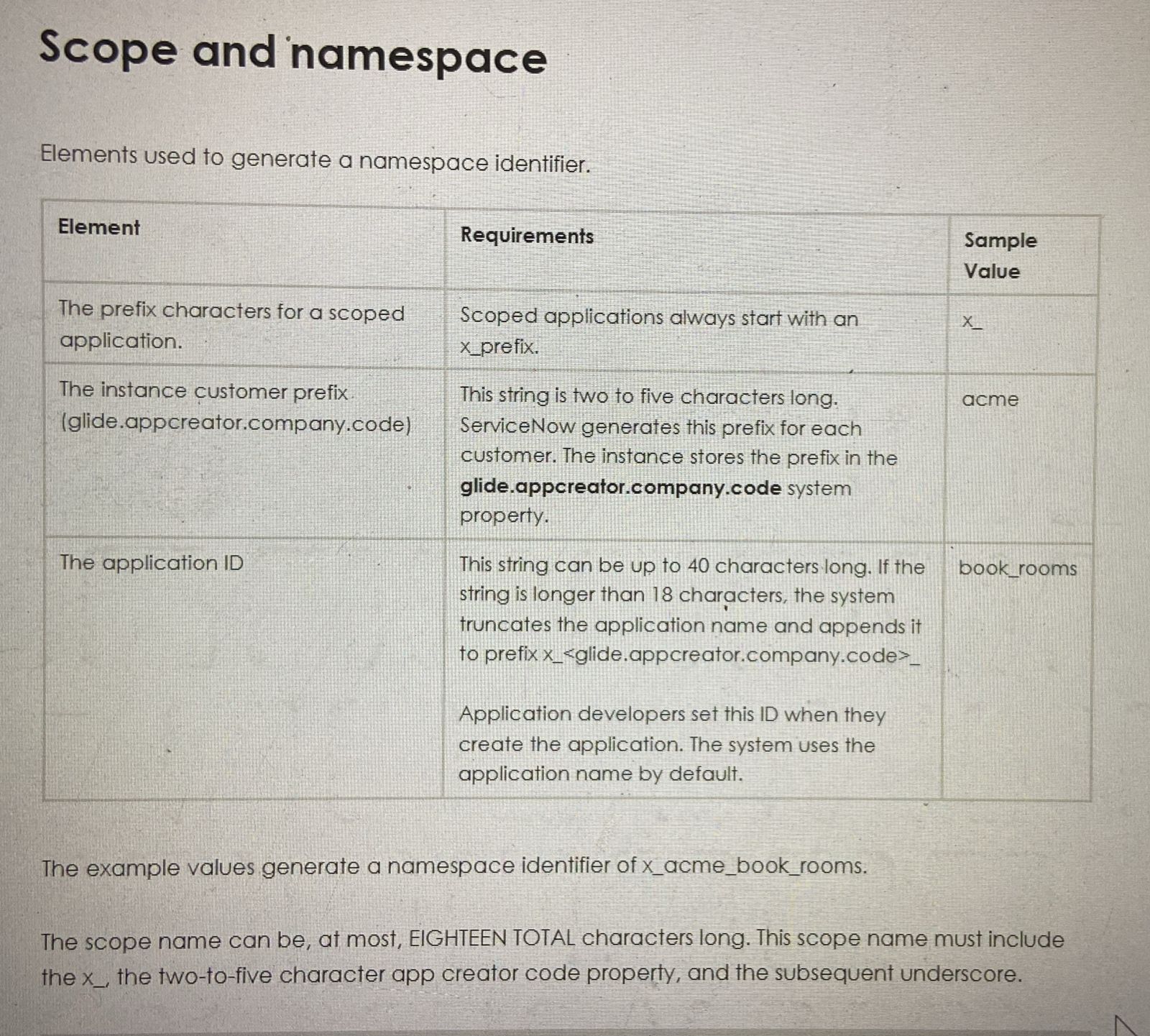
* Everything in Global
* Source Control integration
* Publish to app repository
* Easy file management

Scoped applications are 'sandboxed into their own application namespace, thereby heavily restricting their ability to cause any performance issues to other running applications. They are also more secure in terms of who can access them. For example, the new Scoped HR Application is locked down so only HR users are able to access HR Data; even an Admin cannot access the scoped application.

**Best practice: It is a best practice to work in a private scoped application.**

Artifacts are all of the application files comprising an application. These exist within the application's scope. Artifacts include but are not limited to: Access Controls, Business Rules, Script Includes, Tables, and Client Scripts. In the baseline case, custom applications have read access to each other's table records but cannot perform any other database operations.

**in general, new applications created by ServiceNow will be scoped applications. Scoped application capabilities will continue to mature, expand, and evolve.**



**Scope and delegated development**

Administrators can extend application development to other employees while maintaining control and governance over the platform, developer privileges, application resources, and data access.

Admins can grant non-admin users the ability to develop, deploy, and delete scoped applications.

- **Application-specific permissions.**

* Specify which application file types the developer can access.

- **Instance-specific user roles.**

* Grant the developer access to security records.
* Grant the developer access to script fields.

Admins use Studio to manage Delegated Developers and the content they can access.

Access can be granted to only scoped applications.

Administrators must be familiar with application files and the system table structure to set developer permissions. For example, a developer expected to create advanced business rules needs both the All File Types and Allow Scripting developer permissions.

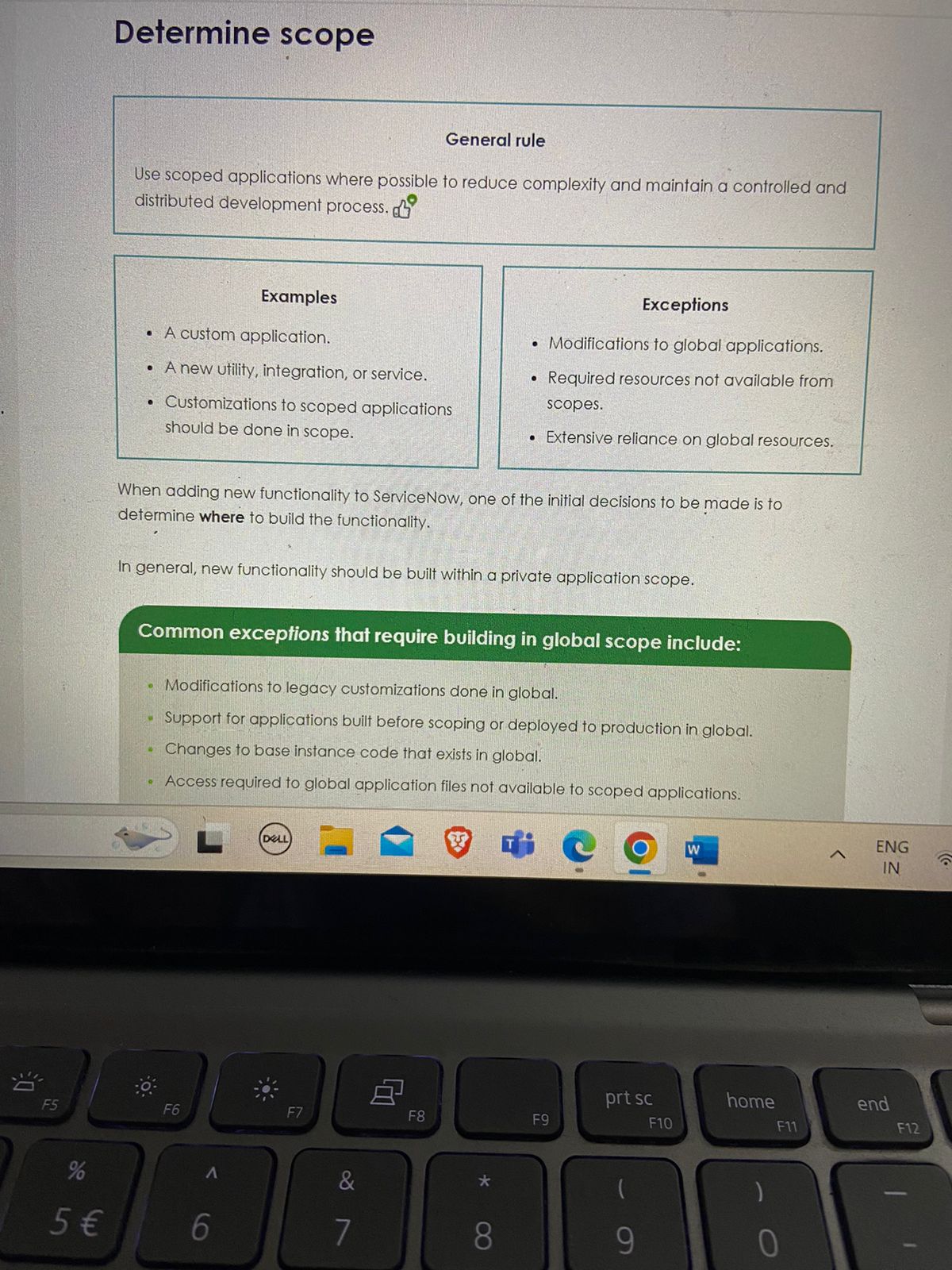
Application-specific deployment permissions allows an administrator to assign a non-administrator as a developer or deployment resource for an application. Administrators can set which specific actions the assigned user (non-administrator) can perform. For example, the admin may enable a user to publish update sets, but prevent publishing to the application repository and ServiceNow Store.

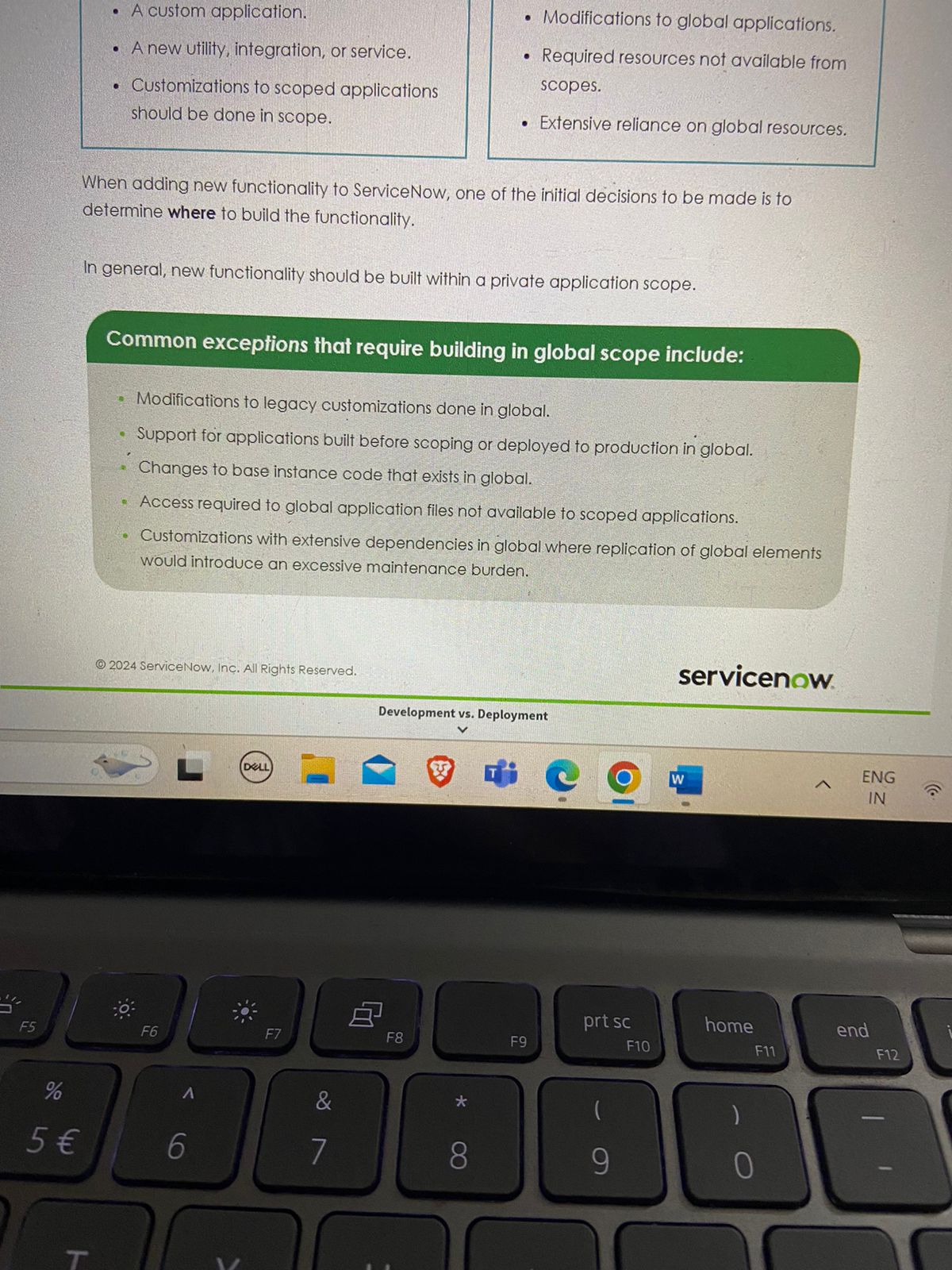
Another example would be a developer who has permission to access all file types for one application does not necessarily have any developer permissions for another application.

Administrators must set developer permissions for each application. In Studio, navigate to File >

Manage Developers to set permissions for individual users and/or groups.

Instance-specific deployment user roles allows an administrator to assign roles that enable non-admin users to install or upgrade all applications in specific instances.





**Development vs. Deployment**

**Development**

* Update sets
* Construction of the next unit of deployment.
* Individuals and teams often manage multiple work streams.

**Deployment**

* Update sets or Application repository
* Application of completed units of development to production.
* Execution of a change management process.

**Update sets**

Update sets are used to manage and store changes (versions) to an application and produce a file for export. Administrators can manually transfer versions between instances with Update Sets.

**Application repository**

The application repository is used for installing and updating applications on all company instances.

To use the application repository, an instance must have a valid subscription, matching application scope, and network access. As a result, personal developer instances cannot connect to the application repository.

**What is an update set?**

Update sets let you apply changes in bulk, and help you force an instance to load the file versions you need.

An update set is:

A container for capturing customizations.

Used to move customizations between instances.

Essential for:

* Customizing baseline applications.
* Customizing applications purchased from the store.

Useful for:

* + Keeping track of why a change was made.
* Exporting work in progress ahead of a clone.

Associating changes with SDLC artifacts.

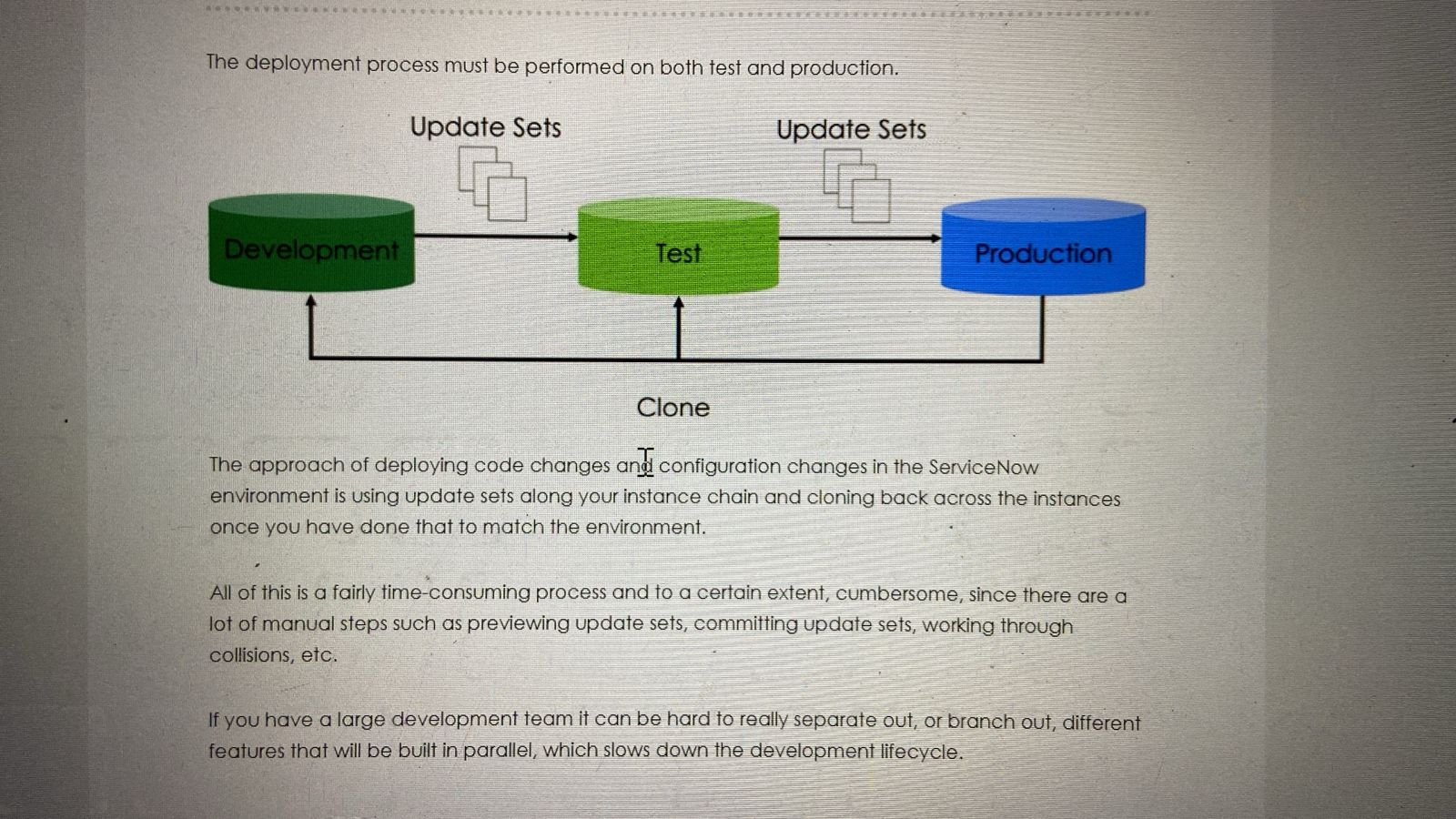
- Stories, problems, bugs, enhancements, etc.

Deploying an application if the app repo is unavailable.

**Update set deployment**

You can always load an update set created on an older family release on an instance running a newer family release. Loading an update set created on a newer family release on an instance running an older family release requires additional testing to determine compatibility.

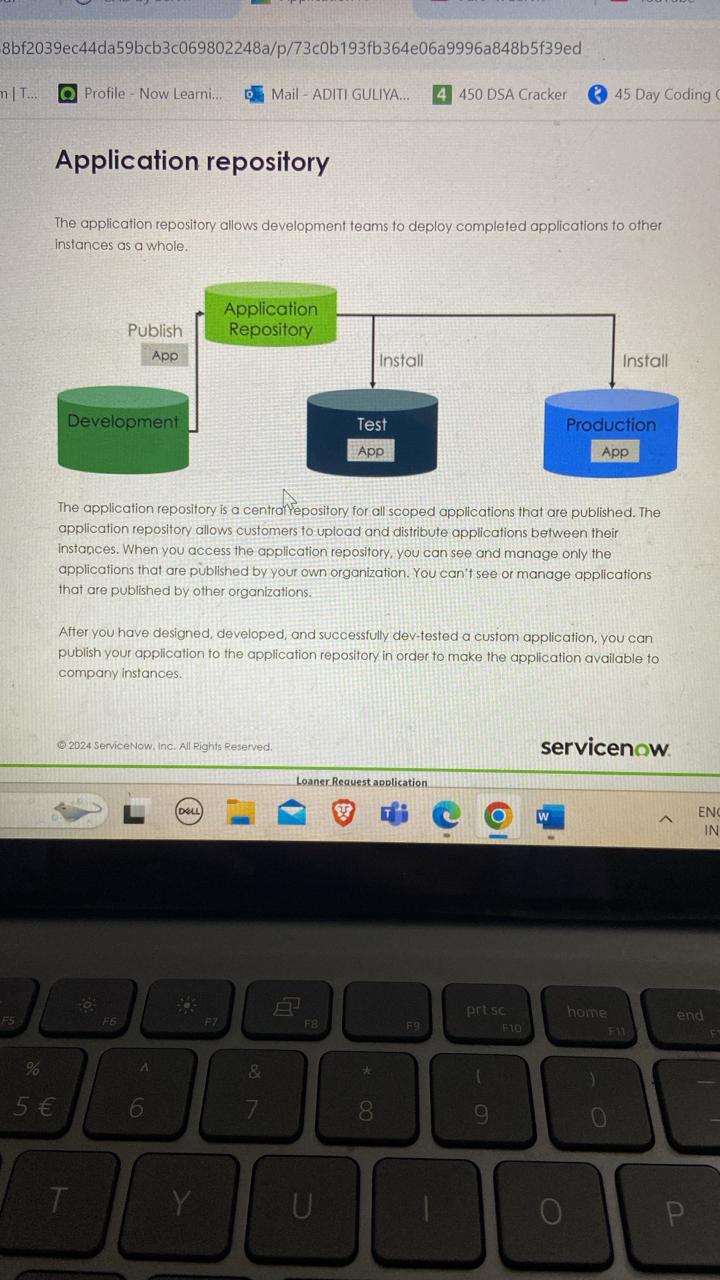
Updates from newer family releases may not produce the same functionality when moved to older family releases. In extreme cases, newer family release updates may cause outages or data loss on an older family release instance. Where possible, avoid moving updates from newer family releases to older family releases. Similar constraints apply to moving updates between instances running different versions of ServiceNow Store apps.

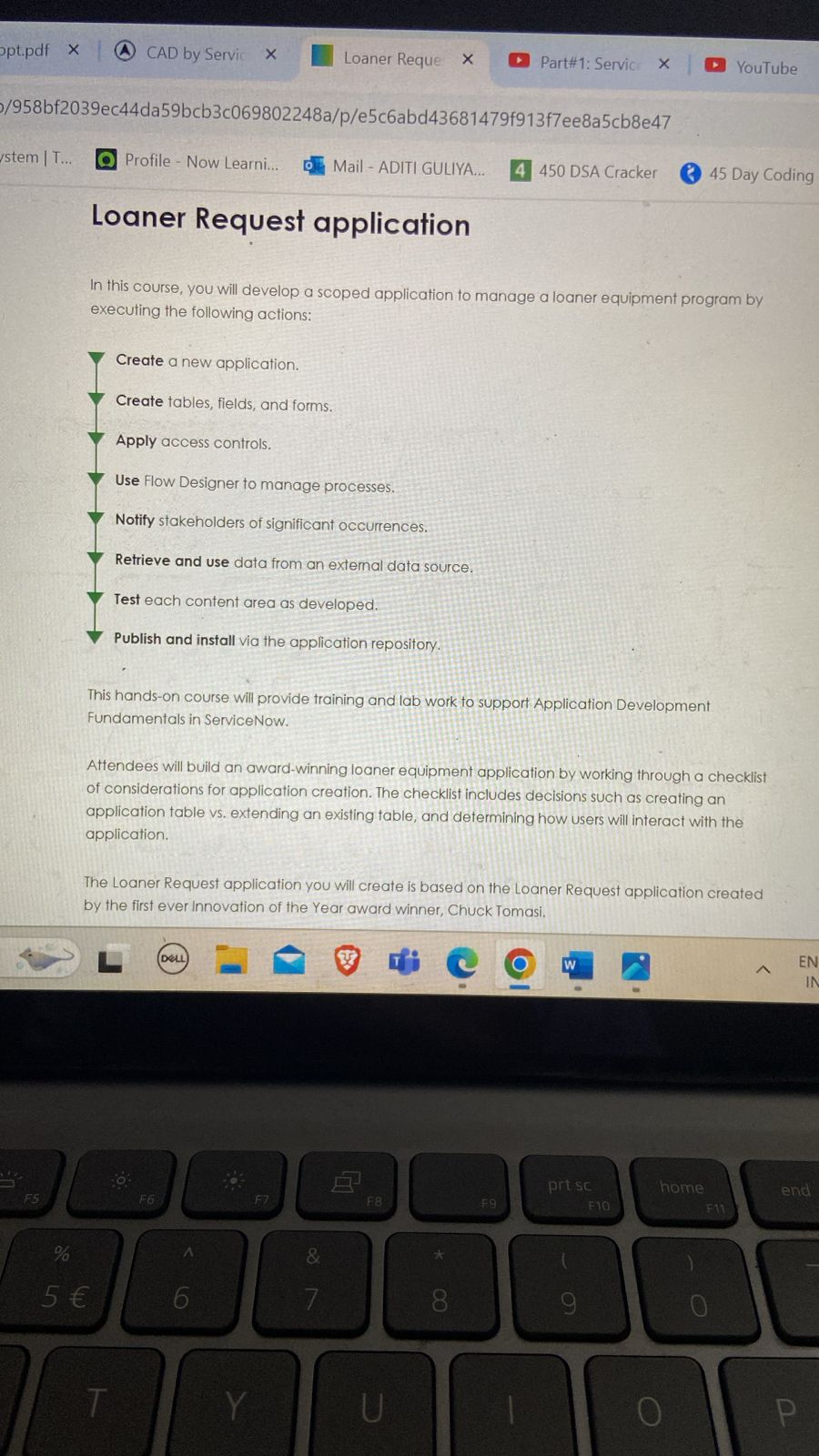


**What is the ServiceNow application repository?**

The ServiceNow application repository:

* Enables companies to store published applications for installation on any instance belonging to that company.
  + **glide.appcreator.company.code value must be the same on all instances (maint only property).**
* Standardizes applications and versions installet on instances.
* Makes it easy to install/uninstall/update apps.
  + Admin role required.





**Scoped Development Process**

**Analyse , design , develop, test deploy**

**JavaScript in ServiceNow: JavaScript mode**

To support existing scripts and new scripts developed to the ECMAScript 2021 standard, the JavaScript engine has three modes:

1. **Compatibility Mode**
2. **ES5 Standards Mode**
3. **ECMAScript 2021 (ES12)**

**ECMAScript 2021 (ES12)**

ECMAScript 2021 (ES12) is the default mode. It allows server-side scripts for scoped applications to use the EMAScript 2021 (ES12) standard. This mode does not preserve the legacy behaviors in the pre-Tokyo JavaScript engine or work with global scripts. This mode supports a subset of ECMAScript syntax and features such as:

* + Let declaration.
* Default function parameters.
* For-of loops.
* Const declaration.
* Additional ECMAScript 2021 (ES12) syntax and features can be found at docs.servicenow.com.

**ES5 standards mode**

ES5 Standards Mode supports ECMAScript5 syntax, extensions and features including:

* The 'use strict declaration.
* Control over extensibility of objects.
* Get and set properties on objects.
* Control overwrite-ability, configurability, and innumerability of object properties.
* New Array and Date methods.
* Native JSON support.

**Compatibility mode**

Compatibility mode is used for all scripts developed prior to the Helsinki release and all global scripts.

Compatibility mode has some differences from the old JavaScript engine.

JSON support changes:

* JSON.stringify() and JSON.parse() are implemented using the ES5 Native JSON object.
* The new JSON().encode() and JSON() .decode () are still supported but should only be used when the legacy behavior is required.
* The use of third-party JavaScript libraries is not supported in Compatibility mode.

**Application Engine Studio (AES)** is the Integrated Development Environment (IDE) for continuing development on applications.

App Engine Studio is a guided, low-code tool for developing rich web applications to store information, automate business processes, and solve business problems. This means that you can delegate development work once assigned to administrators to employees with little to no training.

**Service Now Studio provides an Integrated Development Environment (IDE)** interface for application developers to work on custom applications in one centralized location.

ServiceNow Studio offers a simple way to create, review, and update application files from a tabbed environment. The system opens Studio whenever you edit a custom application.

**Guided Application Creator (GAC)** is an intuitive development interface for building applications on the Now Platform, which provides a step-by-step process to guide you through your initial application development.

The Guided Application Creator (GAC) is enabled via the com.glide.sn-guided-app-creator plugin by default on new and upgraded instances. **Users with the role sn g\_app\_creator.app\_creator have access**. **This feature is intended to be utilized by System Administrators, Developers, and Business Analysts.**

Guided Application Creator (GAC) has two access points:

1. In the main ServiceNow browser window, navigate to System Applications > My Company Applications and select Create New.

2. In Studio, select the Create Application button.

**The welcome screen will appear the first time GAC is used. To see the screen again, remove user preference sn g\_app\_creator.has\_viewed\_gac.**

**App creation process using GAC**

Guided Application Creator (GAC) guides users through the application creation process.

**Note: If the system property sn\_g\_app\_creator.allow\_global is set it to true, any developer or user with the sn g\_app\_creator.app\_creator role has the option to create a global app in GAC. If the property is false, only admins and users with the sn g\_app\_creator.global role can create global apps.**

Source Control integration supports all custom applications. You can use different tools like GitLab, GitHub, Bitbucket, or others. Git is a cloud-based source control management system. It uses repositories and branching.

In the labs for this class, GitLab is used because it allows the use of private repositories at no charge.

If you have an account with another Git tool, feel free to use it for class.

**Follow these steps to link a ServiceNow application to Git:**

**1. Create a repository in the Git tool of your choice.**

**2. Open the application of interest in studio.**

**3. Open Source Control > Link to Source Control.**

**4. Configure the link: repository URL, user name, and password.**

**5. Select the Link To Source Control button.**