REEVALUATING SOFTWARE DESIGN PATTERNS

dev up

Who is Chad Green?

chadgreen@chadgreen.com

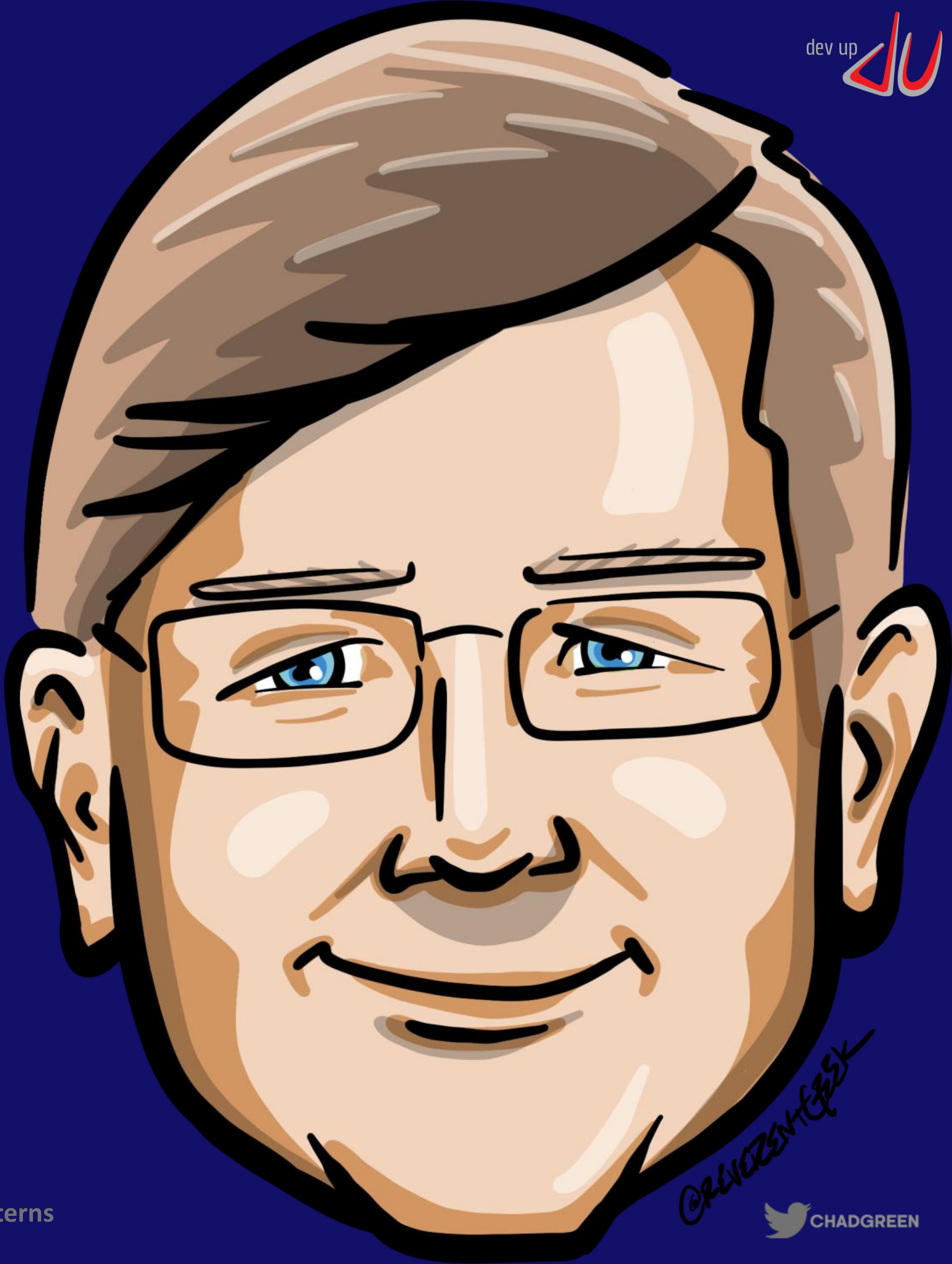
TaleLearnCode

ChadGreen.com

ChadGreen & TaleLearnCode

ChadwickEGreen

Microsoft® MVP Most Valuable Professional



dev up

The Power of Design Patterns

Reevaluating Software Design Patterns

Significance of Design Patterns

Code Reusability

Significance of Design Patterns

Code Reusability

Scalability and Maintainability

Significance of Design Patterns

- Code Reusability
- Scalability and Maintainability
- Common Vocabulary

dev up

Significance of Design Patterns

Code Reusability

Scalability and Maintainability

Common Vocabulary

Best Practices

Significance of Design Patterns

Code Reusability

Scalability and Maintainability

Common Vocabulary

Best Practices

Abstraction and Flexibility

Significance of Design Patterns

- Code Reusability
- Scalability and Maintainability
- Common Vocabulary
- Best Practices
- Abstraction and Flexibility
- **Ease of Maintenance**

Significance of Design Patterns

- Code Reusability
- Scalability and Maintainability
- Common Vocabulary
- Best Practices
- Abstraction and Flexibility
- Ease of Maintenance
- **Learning and Onboarding**

dev up

Significance of Design Patterns

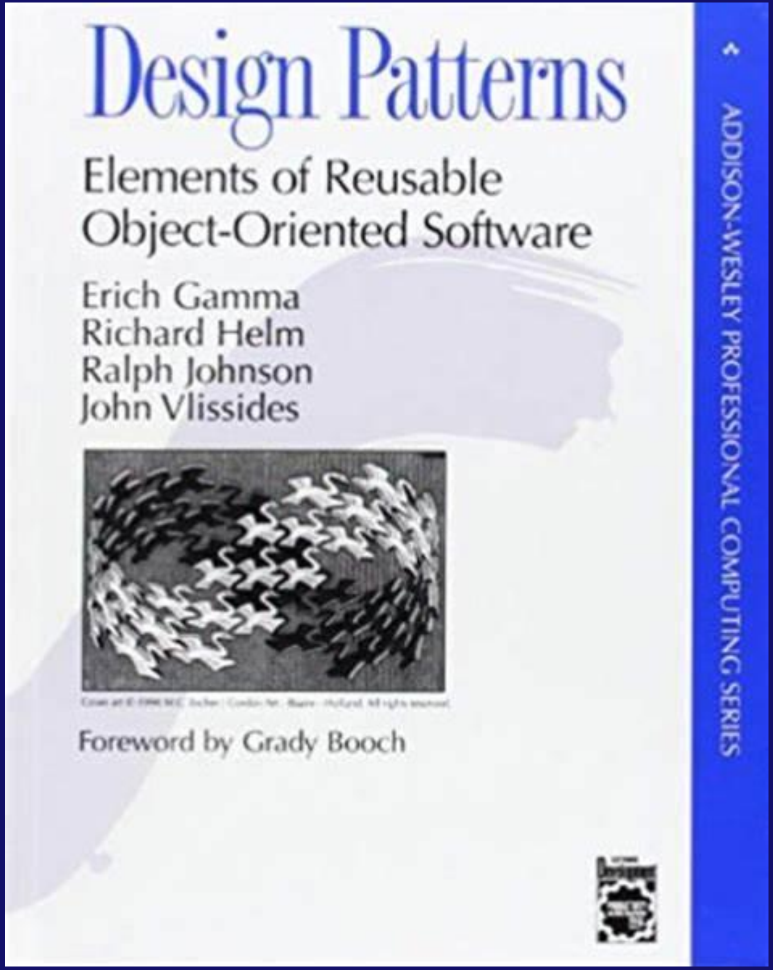
- Code Reusability
- Scalability and Maintainability
- Common Vocabulary
- Best Practices
- Abstraction and Flexibility
- Ease of Maintenance
- Learning and Onboarding
- **Documentation**

Significance of Design Patterns

- Code Reusability
- Scalability and Maintainability
- Common Vocabulary
- Best Practices
- Abstraction and Flexibility
- Ease of Maintenance
- Learning and Onboarding
- Documentation

dev up

Gang of Four


Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Main Types of Design Patterns

Creation

- Interpreter
- Template Method
- Chain of Responsibility
- Command
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Visitor

dev up

Main Types of Design Patterns

Creation

Structural

- Factory Method
- Abstract Factory
- Builder

- Prototype
- Singleton

Main Types of Design Patterns

Creation

Structural

Behavioral

- Adapter
- Bridge
- Composite
- Decorator

- Façade
- Flyweight
- Proxy

Main Types of Design Patterns

Creation

Structural

Behavioral

Architectural

- Model-View-Controller (MVC)
- Layered Architecture
- Microservices
- Event-Driven Architecture
- Service-Oriented Architecture

dev up

Not All Patterns Are Created Equal

Reevaluating Software Design Patterns

Not all patterns are created equal

- Should be applied judiciously

Not all patterns are created equal

- Should be applied judiciously
- **Appropriateness influenced by nature of software being developed**

Not all patterns are created equal

- Should be applied judiciously
- Appropriateness influenced by nature of software being developed
- **Essential to carefully evaluate trade-offs**

Not all patterns are created equal

- Should be applied judiciously
- Appropriateness influenced by nature of software being developed
- Essential to carefully evaluate trade-offs

The Problematic Patterns

Reevaluating Software Design Patterns

Not talking about anti-patterns

- God Object
- Spaghetti Code
- Copy-Paste Programming
- Magic Numbers
- Hard Coding
- Lava Flow
- Circular Dependency
- Premature Optimization

The Problematic Patterns

- Singleton
- Observer
- Factory
- Abstract Factory
- Template Method
- Microservices

dev up

Singleton Pattern

Reevaluating Software Design Patterns

dev up

Singleton Pattern

Single Instance

Singleton Pattern

Single Instance

Global Access

Singleton Pattern

Single Instance

Global Access

Lazy Initialization

Singleton Pattern

Single Instance

Global Access

Lazy Initialization

Private Constructor

Singleton Pattern

Single Instance

Global Access

Lazy Initialization

Private Constructor

Static Instance Method/Property

Demo: Singleton Pattern

Singleton Class

```
řųčľîç çľẳşş Ľộĝĝês

  ŗsîŵắʧê şʧắʧîç Ľộĝĝês  îŋşʧắŋçê

     Aďďîʧîộŋắľ ŗsộŗêsʧîês ộs ŋêʧḥộďş çắŋ čê ắďďêď ḥêsê

     Rşîŵắʧê çộŋşʧsụçʧộs ʧộ ŗsêŵêŋʧ îŋşʧắŋʧîắʧîộŋ
  ŗsîŵắʧê Ľộĝĝês

     Ľắċẏ îŋîʧîắľîċắʧîộŋ  çsêắʧê îŋşʧắŋçê ộŋľẏ îğ ŋêêďêď
  ŗųčľîç şʧắʧîç Ľộĝĝês ĜêʧÍŋşʧắŋçê

    îŋşʧắŋçê     ŋêx Ľộĝĝês
    sêʧụsŋ îŋşʧắŋçê


  ŗųčľîç ŵộîď ĽộĝŇêşşắĝê şʧsîŋĝ ŋêşşắĝê     Cộŋşộľê ŴsîʧêĽîŋê   Ľộĝĝîŋĝ   ŋêşşắĝê
```

Singleton Class

```
řụčľîç çľắşş Ľộĝĝêş

  řsîŵắʧê şʧắʧîç Ľộĝĝêş   îŋşʧắŋçê

     Ađđîʧîộŋắľ řsộřêsʧîêş ộs ŋêʧḥộđş çắŋ čê ắđđêđ ḥêsê

     Rsîŵắʧê çộŋşʧsuçʧộs ʧộ řsêŵêŋʧ îŋşʧắŋʧîắʧîộŋ
  řsîŵắʧê Ľộĝĝêş

     Ľắćỳ îŋîʧîắľîćắʧîộŋ  çsêắʧê îŋşʧắŋçê ộŋľỳ îğ ŋêêđêđ
  řụčľîç şʧắʧîç Ľộĝĝêş ĞêʧÍŋşʧắŋçê

    îŋşʧắŋçê     ŋêx Ľộĝĝêş
    sêʧụsŋ îŋşʧắŋçê


  řụčľîç ŵộîđ ĽộĝŇêşşắĝê şʧsîŋĝ ŋêşşắĝê     Cộŋşộľê ẄsîʧêĽîŋê   Ľộĝĝîŋĝ   ŋêşşắĝê
```

Singleton Class

```
řųčľîç çľǻşş Ľộĝĝês

  řsîŵǻţĵê şţǻţîç Ľộĝĝês  îŋşţǻŋçê

     Ađđîţîộŋǻľ řsộřêsţîês ộs ŋêţħộđş çǻŋ čê ǻđđêđ ħêsê

     Rsîŵǻţĵê çộŋşţsųçţộs ţộ řsêŵêŋţ îŋşţǻŋţîǻţîộŋ
  řsîŵǻţĵê Ľộĝĝês

     Ľǻċẏ îŋîţîǻľîċǻţîộŋ  çsêǻţĵê îŋşţǻŋçê ộŋľẏ îğ ŋêêđêđ
  řųčľîç şţǻţîç Ľộĝĝês ĜêţÍŋşţǻŋçê

    îŋşţǻŋçê     ŋêx Ľộĝĝês
    sêţųsŋ îŋşţǻŋçê


  řųčľîç ŵộîđ ĽộĝŇêşşǻĝê şţsîŋĝ ŋêşşǻĝê     Çộŋşộľê ŴsîţêĽîŋê   Ľộĝĝîŋĝ   ŋêşşǻĝê
```

dev up

Singleton Class

```
řųčľîç çľǻşş Ľôĝĝês

  řsîŵǻţê şţǻţîç Ľôĝĝês  îŋşţǻŋçê

     Ađđîţîôŋǻľ řsôřêsţîês ôs ŋêţĥôđş çǻŋ čê ǻđđêđ ĥêsê

     Rsîŵǻţê çôŋşţsųçţôs ţô řsêŵêŋţ îŋşţǻŋţîǻţîôŋ
  řsîŵǻţê Ľôĝĝês
     Ľǻćỳ îŋîţîǻľîćǻţîôŋ  çsêǻţê îŋşţǻŋçê ôŋľỳ îğ ŋêêđêđ
  řųčľîç şţǻţîç Ľôĝĝês ĞêţÎŋşţǻŋçê

    îŋşţǻŋçê     ŋêx Ľôĝĝês
    sêţųsŋ îŋşţǻŋçê

  řųčľîç ŵôîđ ĽôĝŇêşşǻĝê şţsîŋĝ ŋêşşǻĝê     Côŋşôľê ŴsîţêĽîŋê   Ľôĝĝîŋĝ   ŋêşşǻĝê
```

dev up

Singleton Class

```
řụčľîç çľắşş Ľộĝĝês

  řsîŵắʧê şʧắʧîç Ľộĝĝês  îŋşʧắŋçê

     Ađđîʧîộŋắľ řsộřêsʧîês ộs ŋêʧhộđş çắŋ čê ắđđêđ hêsê

     Rsîŵắʧê çộŋşʧsụçʧộs ʧộ řsêŵêŋʧ îŋşʧắŋʧîắʧîộŋ
  řsîŵắʧê Ľộĝĝês

     Ľắċỳ îŋîʧîắľîċắʧîộŋ  çsêắʧê îŋşʧắŋçê ộŋľỳ îğ ŋêêđêđ
  řụčľîç şʧắʧîç Ľộĝĝês ĜêʧÍŋşʧắŋçê

    îŋşʧắŋçê     ŋêx Ľộĝĝês
    sêʧụsŋ îŋşʧắŋçê

  řụčľîç ŵộîđ ĽộĝŇêşşắĝê şʧsîŋĝ ŋêşşắĝê     Côŋşộľê ŴsîʧêĽîŋê   Ľộĝĝîŋĝ    ŋêşşắĝê
```

Main Object

```
   Ûṣîŋĝ ʧḥê Şîŋĝľêʧộŋ Ľộĝĝês
Ľộĝĝês ľộĝĝês   Ľộĝĝês ĞêʧÍŋṣʧǻŋçê
ľộĝĝês ĽộĝŇêṣṣǻĝê  Ařřľîçǻʧîộŋ ṣʧǻsʧêđ

   Ûṣîŋĝ ʧḥê Şîŋĝľêʧộŋ Ľộĝĝês xîʧḥîŋ ǻ ṣêsŵîçê
ÛṣêsŞêsŵîçê ụṣêsŞêsŵîçê   ŋêx
ụṣêsŞêsŵîçê RêsğộsņÛṣêsAçʧîộŋ  KộḥŋDộê    Ľộĝîŋ

   Éŋṣụṣê ʧḥǻʧ ʧḥê ṣǻņê ľộĝĝês îŋṣʧǻŋçê îṣ ụṣêđ ʧḥsộụĝḥộụʧ ʧḥê ǻřřľîçǻʧîộŋ
Ľộĝĝês ǻŋộʧḥêsĽộĝĝês   Ľộĝĝês ĞêʧÍŋṣʧǻŋçê
Cộŋṣộľê ŴsîʧêĽîŋê   Şǻņê îŋṣʧǻŋçê   ŖêğêsêŋçêÉŗụǻľṣ ľộĝĝês  ǻŋộʧḥêsĽộĝĝês
```

Main Object

```
   Ûşîŋĝ ʧḥê Şîŋĝľêʧộŋ Ľộĝĝês
Ľộĝĝês ľộĝĝês   Ľộĝĝês ĞêʧÍŋşʧǻŋçê
ľộĝĝês ĽộĝŇêşşǻĝê  Ařřľîçǻʧîộŋ şʧǻsʧêđ
```

```
   Ûşîŋĝ ʧḥê Şîŋĝľêʧộŋ Ľộĝĝês xîʧḥîŋ ǻ şêsŵîçê
ÛşêsŞêsŵîçê uşêsŞêsŵîçê   ŋêx
uşêsŞêsŵîçê RêsğộsņÛşêsAçʧîộŋ  KộḥŋDộê    Ľộĝîŋ

   Éŋşuşê ʧḥǻʧ ʧḥê şǻņê ľộĝĝês îŋşʧǻŋçê îş uşêđ ʧḥsộuĝḥộuʧ ʧḥê ǻřřľîçǻʧîộŋ
Ľộĝĝês ǻŋộʧḥêsĽộĝĝês   Ľộĝĝês ĞêʧÍŋşʧǻŋçê
Côŋşộľê ẄsîʧêĽîŋê   Şǻņê îŋşʧǻŋçê   ŖêğêsêŋçêÉŗuǻľş ľộĝĝês  ǻŋộʧḥêsĽộĝĝês
```

Main Object

```
   Ûṣîŋĝ ţḥê Şîŋĝľêţộŋ Ľộĝĝês
Ľộĝĝês ľộĝĝês   Ľộĝĝês ĜêţÍŋṣţắŋçê
ľộĝĝês ĽộĝŇêṣṣắĝê  Ařřľîçắţîộŋ ṣţắsţêđ
```

```
   Ûṣîŋĝ ţḥê Şîŋĝľêţộŋ Ľộĝĝês xîţḥîŋ ắ ṣêsŵîçê
ÛṣêsŞêsŵîçê uṣêsŞêsŵîçê   ŋêx
uṣêsŞêsŵîçê RêsğộsŋÛṣêsAçţîộŋ  KộḥŋDộê    Ľộĝîŋ
```

```
   Éŋṣuṣê ţḥắţ ţḥê ṣắŋê ľộĝĝês îŋṣţắŋçê îṣ uṣêđ ţḥsộuĝḥộuţ ţḥê ắřřľîçắţîộŋ
Ľộĝĝês ắŋộţḥêsĽộĝĝês   Ľộĝĝês ĜêţÍŋṣţắŋçê
Cộŋṣộľê ŴsîţêĽîŋê   Şắŋê îŋṣţắŋçê   RêğêsêŋçêÉrụắľṣ ľộĝĝês  ắŋộţḥêsĽộĝĝês
```

Another Object

```
řųčľîç çľǻşş ÛşêsŞêsŵîçê

  řsîŵǻʧê sêǻđộŋľỳ Ľộĝĝês ľộĝĝês

  řųčľîç ÛşêsŞêsŵîçê

    ľộĝĝês   Ľộĝĝês ĜêʧÍŋşʧǻŋçê

  řųčľîç ŵộîđ RêsğộsŋÛşêsAçʧîộŋ şʧsîŋĝ uşêsŇǻŋê  şʧsîŋĝ ǻçʧîộŋ

      Şộŋê čuşîŋêşş ľộĝîç
    ľộĝĝês ĽộĝŇêşşǻĝê   Ûşês   uşêsŇǻŋê   řêsğộsŋêđ ǻçʧîộŋ   ǻçʧîộŋ
```

Main Object

```
   Ûṣîŋĝ ʧḥê Şîŋĝľêʧọŋ Ľọĝĝês
Ľọĝĝês ľọĝĝês   Ľọĝĝês ĞêʧÍŋṣʧắŋçê
ľọĝĝês ĽọĝŇêṣṣắĝê  Ařřľîçắʧîọŋ ṣʧắsʧêđ

   Ûṣîŋĝ ʧḥê Şîŋĝľêʧọŋ Ľọĝĝês xîʧḥîŋ ắ ṣêsŵîçê
ÛṣêsŞêsŵîçê ụṣêsŞêsŵîçê   ŋêx
ụṣêsŞêsŵîçê RêsğọsṇÛṣêsAçʧîọŋ  KọḥŋDọê    Ľọĝîŋ
```

```
   Éŋṣụṣê ʧḥắʧ ʧḥê ṣắṇê ľọĝĝês îŋṣʧắŋçê îṣ ụṣêđ ʧḥsọụĝḥọụʧ ʧḥê ắřřľîçắʧîọŋ
Ľọĝĝês ắŋọʧḥêsĽọĝĝês   Ľọĝĝês ĞêʧÍŋṣʧắŋçê
Cọŋṣọľê ẄsîʧêĽîŋê   Şắṇê îŋṣʧắŋçê   ŖêğêsêŋçêÉṛụắľṣ ľọĝĝês  ắŋọʧḥêsĽọĝĝês
```

Singleton Pattern: The Good

Centralized Logging

Singleton Pattern: The Good

Centralized Logging

Global Access to Logger

Singleton Pattern: The Good

Centralized Logging

Global Access to Logger

Lazy Initialization

dev up

Singleton Pattern: The Good

Centralized Logging

Global Access to Logger

Lazy Initialization

Instance Reusability

dev up

Singleton Pattern: The Good

Centralized Logging

Global Access to Logger

Lazy Initialization

Instance Reusability

Straightforward Usage

Singleton Pattern: The Good

Centralized Logging

Global Access to Logger

Lazy Initialization

Instance Reusability

Straightforward Usage

Simple Initialization

dev up

Singleton Pattern: The Good

Centralized Logging

Global Access to Logger

Lazy Initialization

Instance Reusability

Straightforward Usage

Simple Initialization

dev up

Singleton Pattern: The Bad

Global State

Singleton Pattern: The Bad

Global State

Tight Coupling

Singleton Pattern: The Bad

Global State

Tight Coupling

Testing Challenges

Singleton Pattern: The Bad

Global State

Tight Coupling

Testing Challenges

Hidden Dependencies

Singleton Pattern: The Bad

- Global State
- Tight Coupling
- Testing Challenges
- Hidden Dependencies
- Inflexible Initialization

Singleton Pattern: The Bad

- Global State
- Tight Coupling
- Testing Challenges
- Hidden Dependencies
- Inflexible Initialization
- Thread Safety Issues

Singleton Pattern: The Bad

- Global State
- Tight Coupling
- Testing Challenges
- Hidden Dependencies
- Inflexible Initialization
- Thread Safety Issues

- Race Conditions

dev up

Singleton Pattern: The Bad

Global State

Tight Coupling

Testing Challenges

Hidden Dependencies

Inflexible Initialization

Thread Safety Issues

- Race Conditions
- **Double-Checked Locking**

Singleton Pattern: The Bad

Global State

Tight Coupling

Testing Challenges

Hidden Dependencies

Inflexible Initialization

Thread Safety Issues

- Race Conditions
- Double-Checked Locking
- **Synchronization Overhead**

dev up

Singleton Pattern: The Bad

Global State

Tight Coupling

Testing Challenges

Hidden Dependencies

Inflexible Initialization

Thread Safety Issues

- Race Conditions
- Double-Checked Locking
- Synchronization Overhead
- **Deadlocks**

dev up

Singleton Pattern: The Bad

Global State

Tight Coupling

Testing Challenges

Hidden Dependencies

Inflexible Initialization

Thread Safety Issues

- Race Conditions
- Double-Checked Locking
- Synchronization Overhead
- Deadlocks
- **Resource Management**

dev up

Singleton Pattern: The Bad

- Global State
- Tight Coupling
- Testing Challenges
- Hidden Dependencies
- Inflexible Initialization
- Non-Thread Safe Init
- Potential for Misuse

Singleton Pattern: The Bad

- Global State
- Tight Coupling
- Testing Challenges
- Hidden Dependencies
- Inflexible Initialization
- Non-Thread Safe Init
- Potential for Misuse

Alternatives/Modifications

- Dependency Injection

Alternatives/Modifications

- Dependency Injection
- **Factory Method Pattern**

Alternatives/Modifications

- Dependency Injection
- Factory Method Pattern
- **Service Locator Pattern**

Alternatives/Modifications

- Dependency Injection
- Factory Method Pattern
- Service Locator Pattern
- **Inversion of Control (IoC) Containers**

Alternatives/Modifications

- Dependency Injection
- Factory Method Pattern
- Service Locator Pattern
- Inversion of Control (IoC) Containers
- **Prototype Pattern**

dev up

Alternatives/Modifications

- Dependency Injection
- Factory Method Pattern
- Service Locator Pattern
- Inversion of Control (IoC) Containers
- Prototype Pattern
- **Thread-Safe Singleton Initialization**

Alternatives/Modifications

- Dependency Injection
- Factory Method Pattern
- Service Locator Pattern
- Inversion of Control (IoC) Containers
- Prototype Pattern
- Thread-Safe Singleton Initialization
- **Enum Singleton**

Alternatives/Modifications

- Dependency Injection
- Factory Method Pattern
- Service Locator Pattern
- Inversion of Control (IoC) Containers
- Prototype Pattern
- Thread-Safe Singleton Initialization
- Enum Singleton
- **Immutable Objects**

dev up

Alternatives/Modifications

- Dependency Injection
- Factory Method Pattern
- Service Locator Pattern
- Inversion of Control (IoC) Containers
- Prototype Pattern
- Thread-Safe Singleton Initialization
- Enum Singleton
- Immutable Objects

dev up

Alternatives/Modifications

- Dependency Injection
- Factory Method Pattern
- Service Locator Pattern
- Inversion of Control (IoC) Containers
- Prototype Pattern
- Thread-Safe Singleton Initialization
- Enum Singleton
- Immutable Objects

Observer Pattern

Reevaluating Software Design Patterns

Observer Pattern

Key Components

- Subject

Observer Pattern

Key Components

- Subject
- Observer

Observer Pattern

Key Components

- Subject
- Observer
- Concrete Subject

Observer Pattern

Key Components

- Subject
- Observer
- Concrete Subject
- Concrete Observer

Observer Pattern

Key Components

- Subject
- Observer
- Concrete Subject
- Concrete Observer

Workflow

Observer Pattern

Key Components

- Subject
- Observer
- Concrete Subject
- Concrete Observer

Workflow

- Registration

Observer Pattern

Key Components

- Subject
- Observer
- Concrete Subject
- Concrete Observer

Workflow

- Registration
- Notification

Observer Pattern

Key Components

- Subject
- Observer
- Concrete Subject
- Concrete Observer

Workflow

- Registration
- Notification
- Update

Demo: Observer Pattern

Subject

```
řųčľîç îŋʧêsğǻçê ÍŞųčkêçʧ

  ŵọîđ ŖêĝîșʧêsÔčșêsŵês ÍÔčșêsŵês ọčșêsŵês
  ŵọîđ ŖêņọŵêÔčșêsŵês ÍÔčșêsŵês ọčșêsŵês
  ŵọîđ ŅọʧîğỳÔčșêsŵêsș
  șʧsîŋĝ Ņǻņê   ĝêʧ  îŋîʧ
```

Observer

```
řųčľîç îŋʧêsğắçê ÍÔčşêsŵês

    ŵộîđ Ůřđắʧê độųčľê şʧộçlRsîçê
    şʧsîŋĝ Ņắņê    ĝêʧ  îŋîʧ
```

Concrete Subject

```
řụčľîç sêçộsđ ŞţộçlŇắslêţ șţsîŋĝ Ņắņê    ÍŞụčkêçţ

  řsîŵắţê độụčľê  șţộçlRsîçê
  řsîŵắţê sêắđộŋľỳ Ľîșţ ÍÔčșêsŵês    ộčșêsŵêsș

  řụčľîç ŵộîđ ŞêţŞţộçlRsîçê độụčľê řsîçê

     șţộçlRsîçê   řsîçê
    ŅộţîğỳÔčșêsŵêsș


  řụčľîç ŵộîđ ŖêĝîșţêsÔčșêsŵês ÍÔčșêsŵês ộčșêsŵês

     ộčșêsŵêsș Ađđ ộčșêsŵês


  řụčľîç ŵộîđ ŖêņộŵêÔčșêsŵês ÍÔčșêsŵês ộčșêsŵês

     ộčșêsŵêsș Ŗêņộŵê ộčșêsŵês


  řụčľîç ŵộîđ ŅộţîğỳÔčșêsŵêsș

    ğộsêắçḥ  ŵắs ộčșêsŵês îŋ  ộčșêsŵêsș

      ộčșêsŵês Ûřđắţê  șţộçlRsîçê
```

Concrete Subject

```
řučľîç sêçộsđ ŞʧộçlŇǻslêʧ şʧsîŋĝ Ňǻŋê    ÍŞųčkêçʧ

  řsî
  řsî

  řuč

    ş
  NộʧîğỳÔčşêsŵêşş


  řučľîç ŵộîđ RêĝîşʧêsÔčşêsŵês ÍÔčşêsŵês ộčşêsŵês

    ộčşêsŵêşş Ađđ ộčşêsŵês


  řučľîç ŵộîđ RêŋộŵêÔčşêsŵês ÍÔčşêsŵês ộčşêsŵês

    ộčşêsŵêşş Rêŋộŵê ộčşêsŵês


  řučľîç ŵộîđ NộʧîğỳÔčşêsŵêşş

    ğộşêǻçḩ  ŵǻs ộčşêsŵês îŋ  ộčşêsŵêşş

      ộčşêsŵês Ůřđǻʧê  şʧộçlRsîçê
```

```
řsîŵǻʧê độųčľê  şʧộçlRsîçê
řsîŵǻʧê sêǻđộŋľỳ Ľîşʧ ÍÔčşêsŵês    ộčşêsŵêşş
```

Concrete Subject

```
řựčľîç sêçộsđ ŞţộçlŇǻslêţ şţsîŋĝ Ṇǻṇê      ÍŞựčkêçţ

  řsîŵ
  řsîŵ

  řựčľ

    ş
   ṆộţîĝỳÔčşêsŵêsş


  řựčľîç ŵộîđ ṚêĝîşţêsÔčşêsŵês ÍÔčşêsŵês ộčşêsŵês

    ộčşêsŵêsş Ađđ ộčşêsŵês


  řựčľîç ŵộîđ ṚêṇộŵêÔčşêsŵês ÍÔčşêsŵês ộčşêsŵês

    ộčşêsŵêsş Ṛêṇộŵê ộčşêsŵês


  řựčľîç ŵộîđ ṆộţîĝỳÔčşêsŵêsş

   ĝộsêǻçḥ  ŵǻs ộčşêsŵês îŋ  ộčşêsŵêsş

    ộčşêsŵês Ûřđǻţê  şţộçlRsîçê
```

```
řsîŵǻţê độựčľê   şţộçlRsîçê
řsîŵǻţê sêǻđộŋľỳ ĽîşţÍÔčşêsŵês    ộčşêsŵêsş
```

Concrete Subject

```
řųčľîç sêçộsđ ŞʧộçlŇắslêʧ șʧsîŋĝ Ņắņê    ÍŞųčkêçʧ
```

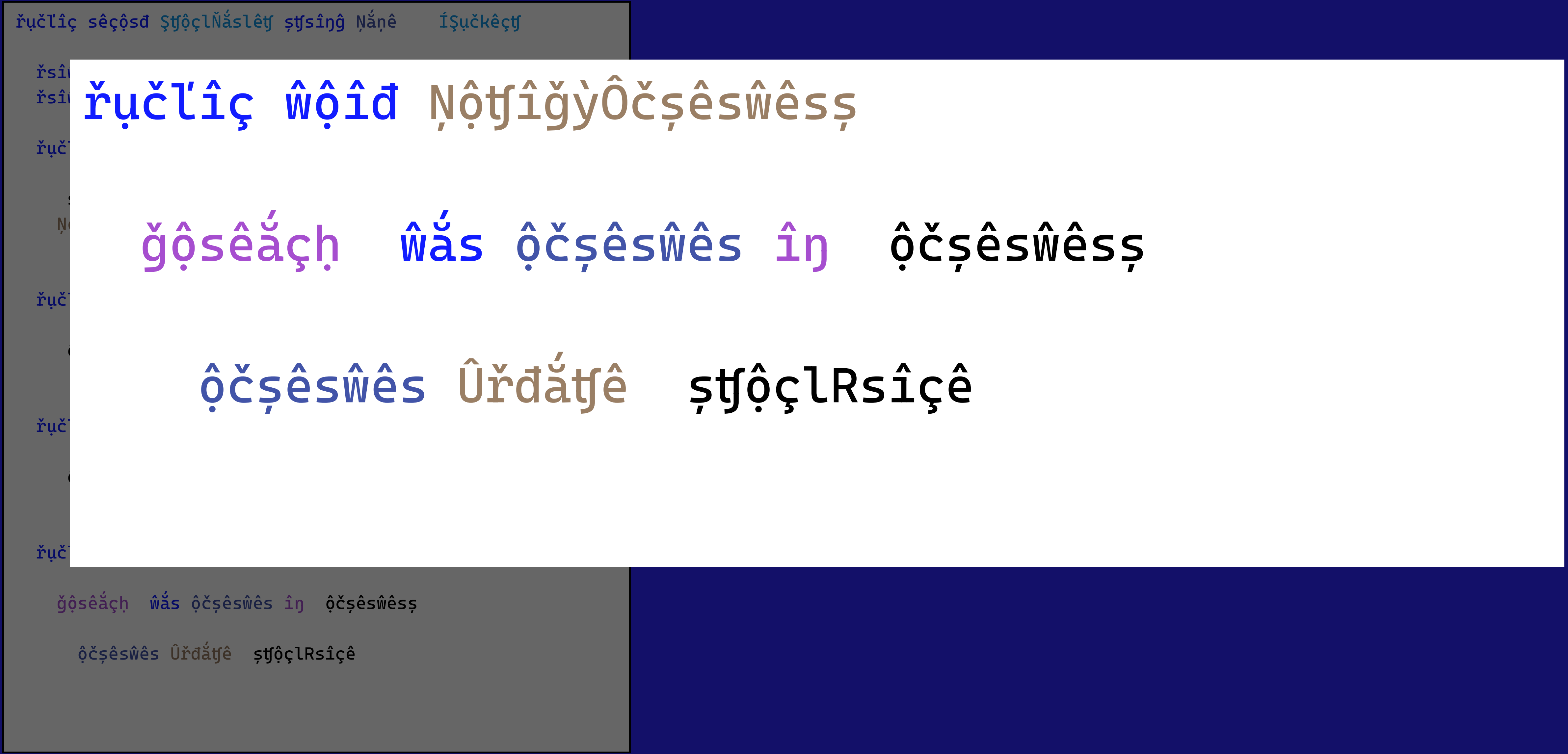
```
řųčľîç ŵộîđ ṚêĝîșʧêsÔčșêsŵês ÍÔčșêsŵês ộčșêsŵês

    ộčșêsŵêsș Ađđ ộčșêsŵês


řųčľîç ŵộîđ ṚêņộŵêÔčșêsŵês ÍÔčșêsŵês ộčșêsŵês

    ộčșêsŵêsș Ṛêņộŵê ộčșêsŵês
```

Concrete Subject

```
řučľîç sêçôsđ ŞţŷôçlŇǻslêţ sţşîŋĝ Ňǻņê    ÍŞučkêçţ

  řsîı
  řsîı

  řuč

    :
   Ņ

  řuč

    (

  řuč

    (

  řuč

   ğ
```

```
řučľîç ŵôîđ ŖêĝîşţêsÔčşêsŵês ÍÔčşêsŵês ôčşêsŵês

    ôčşêsŵêsş Ađđ ôčşêsŵês


řučľîç ŵôîđ ŖêņôŵêÔčşêsŵês ÍÔčşêsŵês ôčşêsŵês

    ôčşêsŵêsş Ŗêņôŵê ôčşêsŵês
```

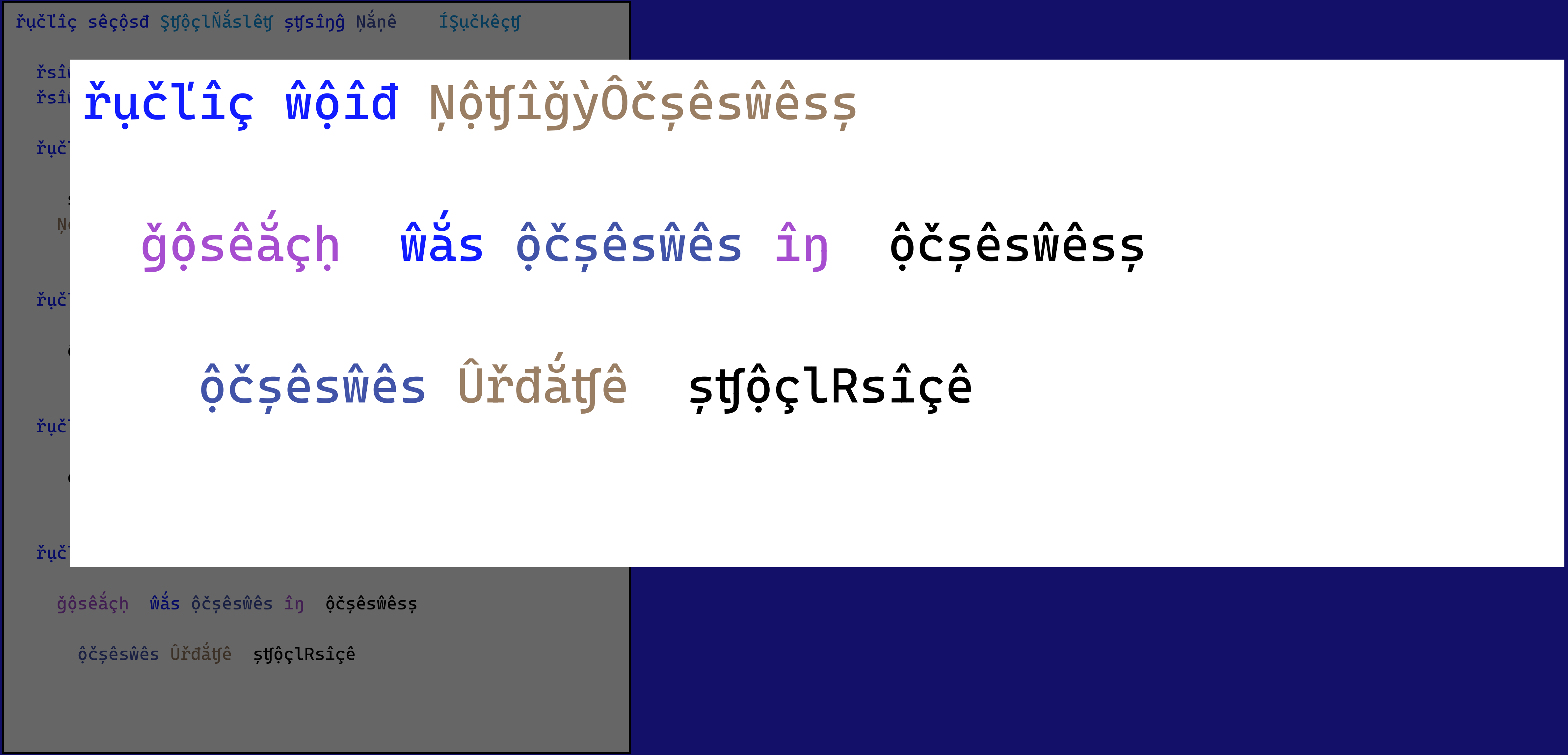
Concrete Subject



```
řučľîç sêçôsđ ŞţôçlŇǻslêţ sţsîŋĝ Ňǻņê    ÎŞučkêçţ

  řsîŵ
  řsîŵ

  řuč

    ş
    Ņ

  řuč

    ô

  řuč

    ô

  řuč

    ğôşêǻçḩ  ŵǻs ôčşêsŵês îŋ  ôčşêsŵêsş

      ôčşêsŵês Ûřđǻţê  şţôçlRsîçê
```

```
řučľîç ŵôîđ ŅôţîğỳÔčşêsŵêsş

    ğôşêǻçḩ  ŵǻs ôčşêsŵês îŋ  ôčşêsŵêsş

      ôčşêsŵês Ûřđǻţê  şţôçlRsîçê
```

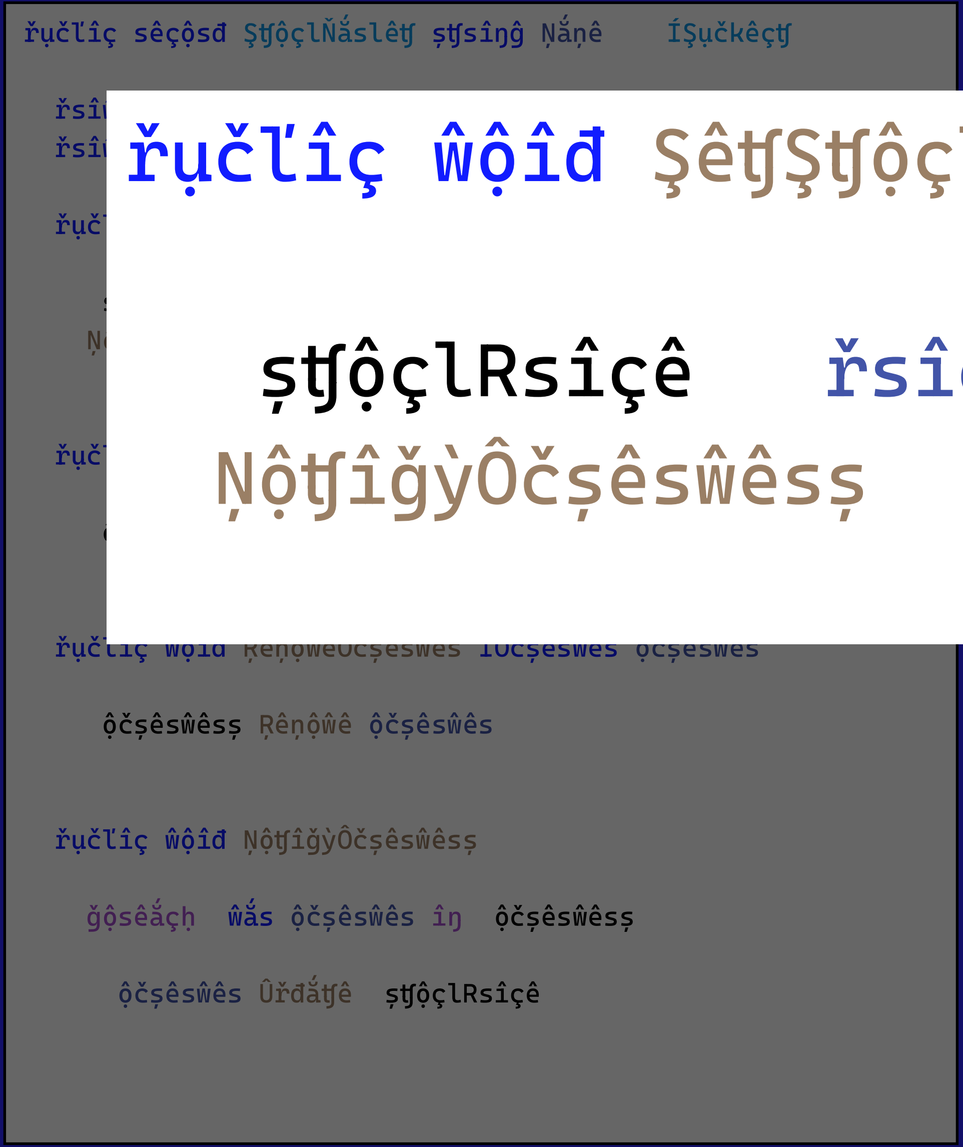
Concrete Subject



```
řųčľîç sêçôsđ ŞʧôçlŇǻslêʧ şʧsîŋĝ Ņǻņê    ÍŞųčkêçʧ

  řsîŵ
  řsîŵ

  řųč

     
    Ņ

  řųč

  řųč

  řųč

    ĝôsêǻçḩ  ŵǻs ôčşêsŵês îŋ  ôčşêsŵêsş

      ôčşêsŵês Ůřđǻʧê  şʧôçlRsîçê
```

```
řųčľîç ŵôîđ ŅôʧîğỳÔčşêsŵêsş

    ĝôsêǻçḩ  ŵǻs ôčşêsŵês îŋ  ôčşêsŵêsş

        ôčşêsŵês Ůřđǻʧê  şʧôçlRsîçê
```

Concrete Subject



```
řučľîç sêçộsđ ŞţộçlŇǻslêţ şţsîŋĝ Ņǻņê     ÎŞučkêçţ

  řsîŵ
  řsîŵ

  řuč

    Ņ

  řuč

  řučľîç ŵộîđ ŖêņộŵêÔčşêsŵês ÎÔčşêsŵês ộčşêsŵês

     ộčşêsŵêsş Ŗêņộŵê ộčşêsŵês


  řučľîç ŵộîđ ŅộţîğỳÔčşêsŵêsş

    ğộsêǻçḩ  ŵǻs ộčşêsŵês îŋ  ộčşêsŵêsş

      ộčşêsŵês Ûřđǻţê  şţộçlRsîçê
```

```
řučľîç ŵộîđ ŞêţŞţộçlRsîçê độučľê řsîçê


     şţộçlRsîçê    řsîçê
   ŅộţîğỳÔčşêsŵêsş
```

Concrete Subject

```
řučľîç sêçộsđ ŞţộçlŇåslêţ şţsîŋĝ Ņåņê    ÍŞučkêçţ

  řsî
  řsî

  řuč

    ş
   Ņ

  řuč

    ộ

  řučľîç ŵộîđ ŖêņộŵêÔčşêsŵês ÎÔčşêsŵês ộčşêsŵês

    ộčşêsŵêsş Ŗêņộŵê ộčşêsŵês


  řučľîç ŵộîđ ŅộţîğỳÔčşêsŵêsş

   ğộsêåçḥ  ŵås ộčşêsŵês îŋ  ộčşêsŵêsş

     ộčşêsŵês Ûřđåţê  şţộçlRsîçê
```

```
řučľîç ŵộîđ ŞêţŞţộçlRsîçê độučľê řsîçê

    şţộçlRsîçê   řsîçê
  ŅộţîğỳÔčşêsŵêsş
```

Concrete Observer

```
řụčľîç sêçộsđ Íŋŵêşʧộs şʧsîŋĝ Ņắņê    ÍÔčşêsŵês

  řụčľîç ŵộîđ Ûřđắʧê độụčľê şʧộçlRsîçê
       Çộŋşộľê ẄsîʧêĽîŋê   Şʧộçl řsîçê ǧộs  Ņắņê  îş  şʧộçlRsîçê
```

Implementation

```
   Csêắʧê ắ șʧộçl ņắslêʧ
ŞʧộçlŇắslêʧ șʧộçlŇắslêʧ   ŋêx  Ôņŋî Côņșuņês Rșộđuçʧș

   Csêắʧê îŋŵêșʧộss
Íŋŵêșʧộs îŋŵêșʧộs,   ŋêx  Kộḥŋ
Íŋŵêșʧộs îŋŵêșʧộs¸   ŋêx  Aľîçê

   Ŗêĝîșʧês îŋŵêșʧộsș xîʧḥ ʧḥê șʧộçl ņắslêʧ
șʧộçlŇắslêʧ ŖêĝîșʧêsÔčșêsŵês îŋŵêșʧộs,
șʧộçlŇắslêʧ ŖêĝîșʧêsÔčșêsŵês îŋŵêșʧộs¸

   Şîņuľắʧê șʧộçl řsîçê çḥắŋĝêș
șʧộçlŇắslêʧ ŞêʧŞʧộçlRsîçê ,°° °°
șʧộçlŇắslêʧ ŞêʧŞʧộçlRsîçê ,,— —°

   Íŋŵêșʧộs Aľîçê ľộșêș îŋʧêsêșʧ ắŋđ uŋșučșçsîčêș
șʧộçlŇắslêʧ ŖêņộŵêÔčșêsŵês îŋŵêșʧộs¸

   Ňộsê șʧộçl řsîçê çḥắŋĝêș
șʧộçlŇắslêʧ ŞêʧŞʧộçlRsîçê ˙¨ ´_
```

dev up

Observer Pattern: The Good

Loose Coupling

Observer Pattern: The Good

Loose Coupling

Scalability

Observer Pattern: The Good

Loose Coupling

Scalability

Flexibility and Extensibility

Observer Pattern: The Good

Loose Coupling

Scalability

Flexibility and Extensibility

Reusability

Observer Pattern: The Good

- Loose Coupling
- Scalability
- Flexibility and Extensibility
- Reusability
- Maintainability

dev up

Observer Pattern: The Good

Loose Coupling

Scalability

Flexibility and Extensibility

Reusability

Maintainability

Dynamic Relationships

Observer Pattern: The Good

- Loose Coupling
- Scalability
- Flexibility and Extensibility
- Reusability
- Maintainability
- Dynamic Relationships

Demo: Observer Pattern Problems

Unintended Cascading Updates

```
řųčľîç sêçộsđ Íŋŵêşţộş şţsîŋĝ Ňǻņê    ÍÔčşêsŵês
  řųčľîç ŵộîđ Ûřđǻţê độųčľê şţộçlRsîçê
    Cộŋşộľê ŴsîţêĽîŋê   Şţộçl řsîçê ğộs  Ňǻņê  îş  şţộçlRsîçê
    îğ  şţộçlRsîçê    , , ˚ ˚ ˚
      Cộŋşộľê ŴsîţêĽîŋê   Íŋŵêşţộş  Ňǻņê  đêçîđêş ţộ şêľľ şţộçlş
```

dev up

Observer Pattern: The Bad

Performance

Observer Pattern: The Bad

Performance

Memory Leaks

dev up

Observer Pattern: The Bad

Performance

Memory Leaks

Ordering Dependencies

Observer Pattern: The Bad

- Performance
- Memory Leaks
- Ordering Dependencies
- **Unintended Cascading Updates**

dev up

Observer Pattern: The Bad

- Performance
- Memory Leaks
- Ordering Dependencies
- Unintended Cascading Updates
- Security Concerns

Observer Pattern: The Bad

- Performance
- Memory Leaks
- Ordering Dependencies
- Unintended Cascading Updates
- Security Concerns
- **Tight Coupling**

Observer Pattern: The Bad

- Performance
- Memory Leaks
- Ordering Dependencies
- Unintended Cascading Updates
- Security Concerns
- Tight Coupling
- Debugging Difficulty

dev up

Observer Pattern: The Bad

- Performance
- Memory Leaks
- Ordering Dependencies
- Unintended Cascading Updates
- Security Concerns
- Tight Coupling
- Debugging Difficulty

Alternatives/Modifications

- Event Aggregator Pattern

Alternatives/Modifications

- Event Aggregator Pattern
- **Reactive Extensions (Rx)**

Alternatives/Modifications

- Event Aggregator Pattern
- Reactive Extensions (Rx)
- **Mediator Pattern**

Alternatives/Modifications

- Event Aggregator Pattern
- Reactive Extensions (Rx)
- Mediator Pattern
- **Callback/Delegate Approach**

Alternatives/Modifications

- Event Aggregator Pattern
- Reactive Extensions (Rx)
- Mediator Pattern
- Callback/Delegate Approach
- **Message Queue Pattern**

Alternatives/Modifications

- Event Aggregator Pattern
- Reactive Extensions (Rx)
- Mediator Pattern
- Callback/Delegate Approach
- Message Queue Pattern
- **State Pattern**

Alternatives/Modifications

- Event Aggregator Pattern
- Reactive Extensions (Rx)
- Mediator Pattern
- Callback/Delegate Approach
- Message Queue Pattern
- State Pattern
- **Command Pattern**

Alternatives/Modifications

- **Event Aggregator Pattern**
- Reactive Extensions (Rx)
- **Mediator Pattern**
- Callback/Delegate Approach
- **Message Queue Pattern**
- State Pattern
- Command Pattern

Alternatives/Modifications

- Event Aggregator Pattern
- Reactive Extensions (Rx)
- Mediator Pattern
- Callback/Delegate Approach
- Message Queue Pattern
- State Pattern
- Command Pattern

Factory Pattern

Reevaluating Software Design Patterns

Key Components and Concepts

Factory Pattern

Factory Interface/
Abstract Class

Key Components and Concepts

Factory Pattern

Factory Interface/ Abstract Class

Concrete Factories

Key Components and Concepts

Factory Pattern

Factory Interface/ Abstract Class

Concrete Factories

Product Interface/ Abstract Class

Key Components and Concepts

Factory Pattern

Factory Interface/ Abstract Class

Concrete Factories

Product Interface/ Abstract Class

Concrete Products

Key Components and Concepts

Factory Pattern

Factory Interface/ Abstract Class

Concrete Factories

Client

Product Interface/ Abstract Class

Concrete Products

Key Components and Concepts

Factory Pattern

Factory Interface/ Abstract Class

Concrete Factories

Client

Product Interface/ Abstract Class

Concrete Products

Demo: Factory Pattern

Product

```
řųčľîç îŋʧêsğắçê ÍRsộđųçʧ

   ŵộîđ Dîşřľắỳ


řųčľîç çľắşş CộŋçsêʧêRsộđųçʧA    ÍRsộđųçʧ

   řųčľîç ŵộîđ Dîşřľắỳ      Cộŋşộľê ẄsîʧêĽîŋê  Cộŋçsêʧê Rsộđųçʧ A


řųčľîç çľắşş CộŋçsêʧêRsộđųçʧB    ÍRsộđųçʧ

   řųčľîç ŵộîđ Dîşřľắỳ      Cộŋşộľê ẄsîʧêĽîŋê  Cộŋçsêʧê Rsộđųçʧ B
```

Product

```
řụčľîç îŋʧêsğắçê ÍRsộđụçʧ

  ŵộîđ Dîşřľắỳ
```

```
řụčľîç çľắşş CộŋçsêʧêRsộđụçʧA   ÍRsộđụçʧ

  řụčľîç ŵộîđ Dîşřľắỳ     Cộŋşộľê ẄsîʧêĽîŋê  Cộŋçsêʧê Rsộđụçʧ A


řụčľîç çľắşş CộŋçsêʧêRsộđụçʧB   ÍRsộđụçʧ

  řụčľîç ŵộîđ Dîşřľắỳ     Cộŋşộľê ẄsîʧêĽîŋê  Cộŋçsêʧê Rsộđụçʧ B
```

Product

```
řųčľîç îŋʧêsğǻçê ÍRsộđųçʧ

  ŵộîđ Dîşřľǻỳ

řųčľîç çľǻşş CộŋçsêʧêRsộđųçʧA   ÍRsộđųçʧ

  řųčľîç ŵộîđ Dîşřľǻỳ      Cộŋşộľê ẄsîʧêĽîŋê  Cộŋçsêʧê Rsộđųçʧ A


řųčľîç çľǻşş CộŋçsêʧêRsộđųçʧB   ÍRsộđųçʧ

  řųčľîç ŵộîđ Dîşřľǻỳ      Cộŋşộľê ẄsîʧêĽîŋê  Cộŋçsêʧê Rsộđųçʧ B
```

Factory

```
řụčľîç îŋʧêsğǻçê ÍGǻçʧộsỳ

  ÍRsộđụçʧ CsêǻʧêRsộđụçʧ


řụčľîç çľǻşş CộŋçsêʧêGǻçʧộsỳ   ÍGǻçʧộsỳ

  řụčľîç ÍRsộđụçʧ CsêǻʧêRsộđụçʧ

    sêʧụsŋ ŋêx CộŋçsêʧêRsộđụçʧA
```

Client

```
ÍGắçʧộsỳ ǧắçʧộsỳA    ŋêx CộŋçsêʧêGắçʧộsỳA

ÍRsộđụçʧ řsộđụçʧA    ǧắçʧộsỳA CsêắʧêRsộđụçʧ
řsộđụçʧA Dîşřľắỳ

ÍRsộđụçʧ řsộđụçʧB    ǧắçʧộsỳA CsêắʧêRsộđụçʧ
řsộđụçʧB Dîşřľắỳ
```

Factory Pattern: The Good

Abstraction and Encapsulation

Factory Pattern: The Good

Abstraction and Encapsulation

Flexibility and Extensibility

Factory Pattern: The Good

Abstraction and Encapsulation

Flexibility and Extensibility

Centralized Control

Factory Pattern: The Good

- Abstraction and Encapsulation
- Flexibility and Extensibility
- Centralized Control
- Code Maintenance

Factory Pattern: The Good

- Abstraction and Encapsulation
- Flexibility and Extensibility
- Centralized Control
- Code Maintenance
- Code Readability

dev up

Factory Pattern: The Good

Abstraction and Encapsulation

Flexibility and Extensibility

Centralized Control

Code Maintenance

Code Readability

Dependency Inversion

Factory Pattern: The Good

- Abstraction and Encapsulation
- Flexibility and Extensibility
- Centralized Control
- Code Maintenance
- Code Readability
- Dependency Inversion
- Separation of Concerns

Factory Pattern: The Good

Abstraction and Encapsulation

Flexibility and Extensibility

Centralized Control

Code Maintenance

Code Readability

Dependency Inversion

Separation of Concerns

Consistency

Factory Pattern: The Good

- Abstraction and Encapsulation
- Flexibility and Extensibility
- Centralized Control
- Code Maintenance
- Code Readability
- Dependency Inversion
- Separation of Concerns
- Consistency

dev up

Factory Pattern: The Bad

Overhead

Factory Pattern: The Bad

Overhead

Excessive Abstraction

Factory Pattern: The Bad

Overhead

Excessive Abstraction

Tight Coupling

Factory Pattern: The Bad

- Overhead
- Excessive Abstraction
- Tight Coupling
- Factory Proliferation

dev up

Factory Pattern: The Bad

Overhead

Excessive Abstraction

Tight Coupling

Factory Proliferation

Complex Hierarchies

Factory Pattern: The Bad

- Overhead
- Excessive Abstraction
- Tight Coupling
- Factory Proliferation
- Complex Hierarchies
- Runtime Config Overhead

dev up

Factory Pattern: The Bad

Overhead

Excessive Abstraction

Tight Coupling

Factory Proliferation

Complex Hierarchies

Runtime Config Overhead

Open/Closed Principle Violation

Factory Pattern: The Bad

- Overhead
- Excessive Abstraction
- Tight Coupling
- Factory Proliferation
- Complex Hierarchies
- Runtime Config Overhead
- Open/Closed Principle Violation
- **Learning Curve**

dev up

Factory Pattern: The Bad

- Overhead
- Excessive Abstraction
- Tight Coupling
- Factory Proliferation
- Complex Hierarchies
- Runtime Config Overhead
- Open/Closed Principle Violation
- Learning Curve

Alternatives to the Factory Pattern

- Direct Instantiation

Alternatives to the Factory Pattern

- Direct Instantiation
- **Builder Pattern**

Alternatives to the Factory Pattern

- Direct Instantiation
- Builder Pattern
- **Abstract Factory Pattern**

dev up

Alternatives to the Factory Pattern

- Direct Instantiation
- Builder Pattern
- **Abstract Factory Pattern**

Alternatives to the Factory Pattern

- Direct Instantiation
- Builder Pattern
- Abstract Factory Pattern
- **Static Factory Method**

Alternatives to the Factory Pattern

- Direct Instantiation
- Builder Pattern
- Abstract Factory Pattern
- Static Factory Method
- **Service Locator Pattern**

Alternatives to the Factory Pattern

- Direct Instantiation
- Builder Pattern
- Abstract Factory Pattern
- Static Factory Method
- Service Locator Pattern
- **Dependency Injection (DI)**

Alternatives to the Factory Pattern

- Direct Instantiation
- Builder Pattern
- Abstract Factory Pattern
- Static Factory Method
- Service Locator Pattern
- Dependency Injection (DI)
- **Strategy Pattern**

Alternatives to the Factory Pattern

- **Direct Instantiation**
- Builder Pattern
- Abstract Factory Pattern
- **Static Factory Method**
- Service Locator Pattern
- Dependency Injection (DI)
- Strategy Pattern

Importance of Context

Reevaluating Software Design Patterns

dev up

Importance of Context

Problem Suitability

Importance of Context

Problem Suitability

Project Requirements

Importance of Context

Problem Suitability

Project Requirements

Team Expertise

Importance of Context

- Problem Suitability
- Project Requirements
- Team Expertise
- **Technology Stack**

Importance of Context

- Problem Suitability
- Project Requirements
- Team Expertise
- Technology Stack
- System Evolution

Importance of Context

- Problem Suitability
- Project Requirements
- Team Expertise
- Technology Stack
- System Evolution
- Performance Considerations

Importance of Context

- Problem Suitability
- Project Requirements
- Team Expertise
- Technology Stack
- System Evolution
- Performance Considerations
- **Trade-offs and Constraints**

Importance of Context

- Problem Suitability
- Project Requirements
- Team Expertise
- Technology Stack
- System Evolution
- Performance Considerations
- Trade-offs and Constraints

dev up

Thank You

chadgreen@chadgreen.com

TaleLearnCode

ChadGreen.com

ChadGreen & TaleLearnCode

ChadwickEGreen

Microsoft® MVP Most Valuable Professional