REEVALUATING SOFTWARE DESIGN PATTERNS

Who is Chad Green?

chadgreen@chadgreen.com

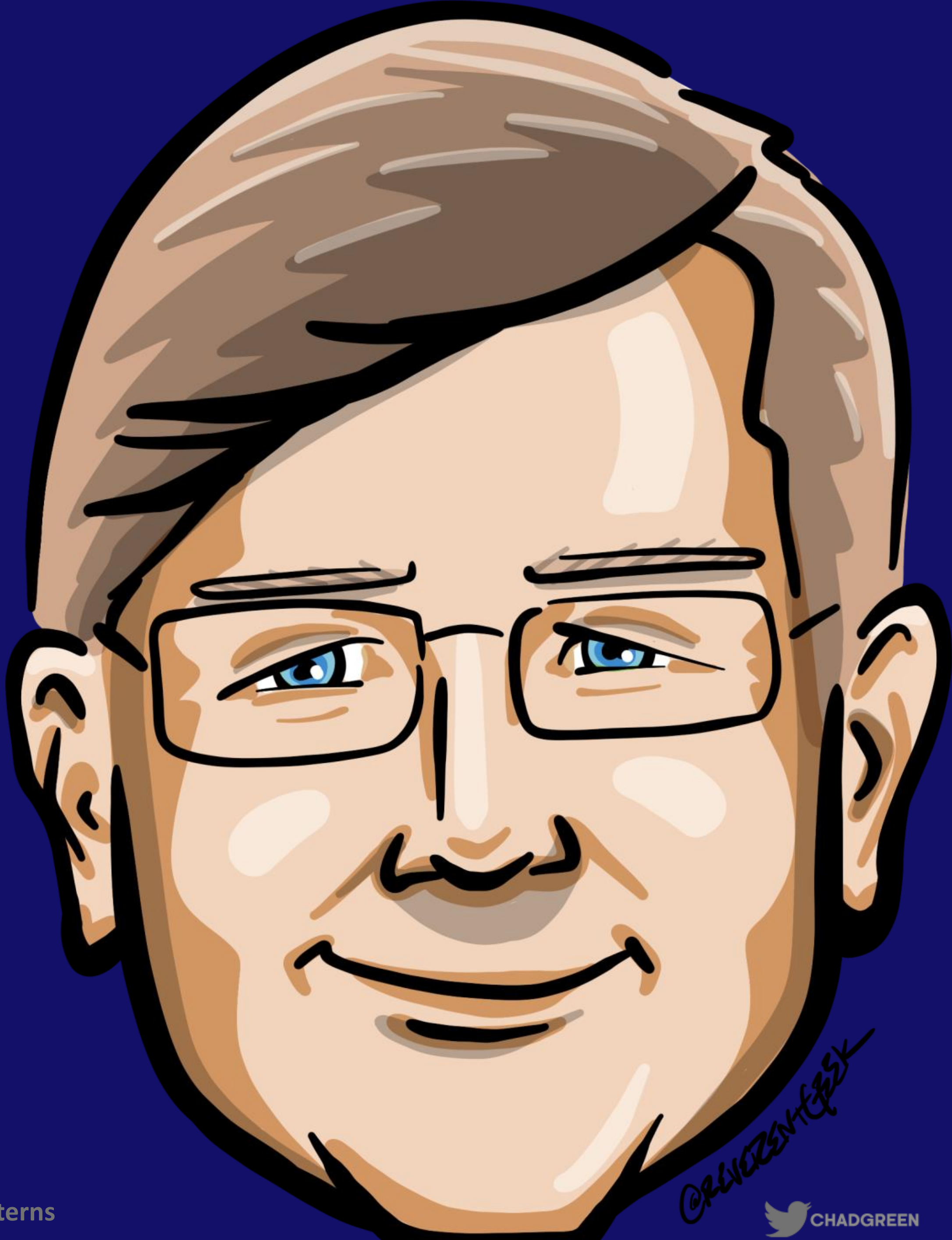
TaleLearnCode

ChadGreen.com

ChadGreen & TaleLearnCode

ChadwickEGreen

Microsoft® MVP Most Valuable Professional



Inspiration Strikes

BEER
CITY
CODE

Inspiration Strikes

Inspiration Strikes

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

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

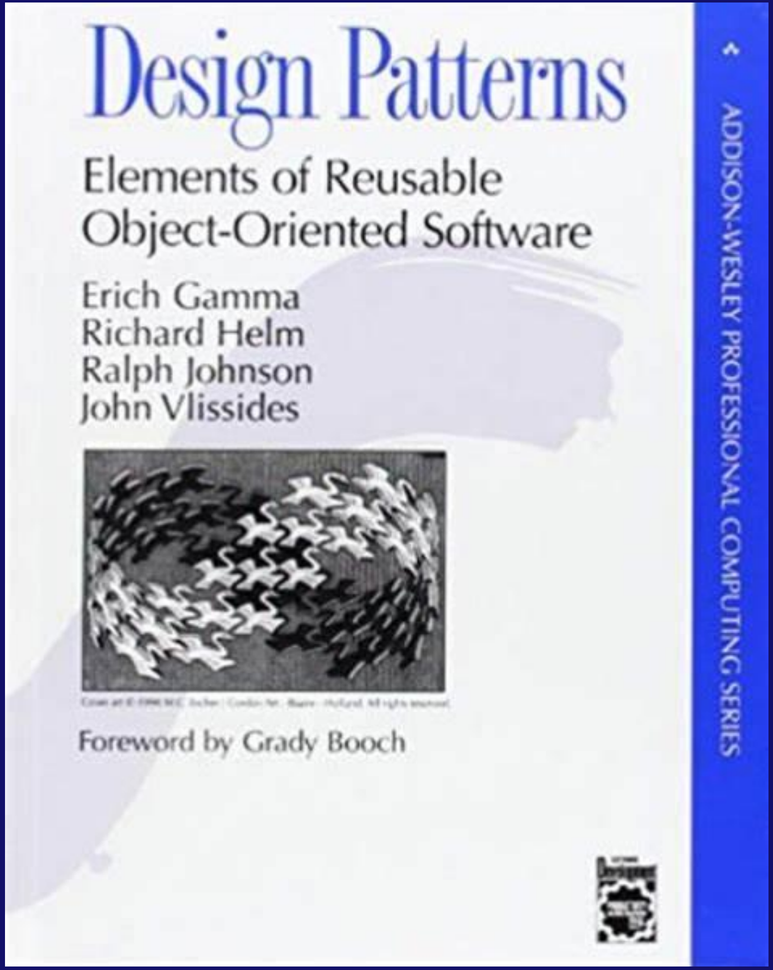
Significance of Design Patterns

- 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

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

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

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

Singleton Pattern

Reevaluating Software Design Patterns

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 çľǻșș Ľộĝĝêș

  řsîŵǻţĵê șţĵǻţĵîç Ľộĝĝêș  îŋșţĵǻŋçê

     Ađđîţĵîộŋǻľ řsộřêsţĵîêș ộs ņêţĵḥộđș çǻŋ čê ǻđđêđ ḥêsê

     Rsîŵǻţĵê çộŋșţĵsụçţĵộs ţĵộ řsêŵêŋţĵ îŋșţĵǻŋţĵîǻţĵîộŋ
  řsîŵǻţĵê Ľộĝĝêș

     Ľǻċỳ îŋîţĵîǻľîċǻţĵîộŋ  çsêǻţĵê îŋșţĵǻŋçê ộŋľỳ îğ ŋêêđêđ
  řųčľîç șţĵǻţĵîç Ľộĝĝêș ĜêţĵÍŋșţĵǻŋçê

    îŋșţĵǻŋçê     ŋêx Ľộĝĝêș
    sêţĵụsŋ îŋșţĵǻŋçê


  řųčľîç ŵộîđ ĽộĝŇêșșǻĝê șţĵsîŋĝ ņêșșǻĝê     Cộŋșộľê ẄsîţĵêĽîŋê   Ľộĝĝîŋĝ   ņêșșǻĝê
```

Singleton Class

```
ŗųčľîç çľăşş Ŀôĝĝêş

  ŗşîŵăţê ştăţîç Ŀôĝĝêş  îŋşţăŋçê

     Ađđîţîôŋăľ ŗşôŗêşţîêş ôş ŋêţĥôđş çăŋ čê ăđđêđ ĥêşê

     Rşîŵăţê çôŋşţşųçţôş ţô ŗşêŵêŋţ îŋşţăŋţîăţîôŋ
  ŗşîŵăţê Ŀôĝĝêş

     Ľăćŷ îŋîţîăľîćăţîôŋ  çşêăţê îŋşţăŋçê ôŋľŷ îğ ŋêêđêđ
  ŗųčľîç ştăţîç Ŀôĝĝêş ĞêţÎŋşţăŋçê

    îŋşţăŋçê     ŋêx Ŀôĝĝêş
    şêţųşŋ îŋşţăŋçê


  ŗųčľîç ŵôîđ ĿôĝŇêşşăĝê şţşîŋĝ ŋêşşăĝê     Côŋşôľê ŴşîţêĿîŋê   Ŀôĝĝîŋĝ   ŋêşşăĝê
```

Singleton Class

```
řụčľîç çľǻșș Ľộĝĝês

  řsîŵǻʧê șʧǻʧîç Ľộĝĝês  îŋșʧǻŋçê

    Ađđîʧîộŋǻľ řsộřêsʧîês ộs ŋêʧĥộđș çǻŋ čê ǻđđêđ ĥêsê

    Rșîŵǻʧê çộŋșʧsụçʧộs ʧộ řsêŵêŋʧ îŋșʧǻŋʧîǻʧîộŋ
  řsîŵǻʧê Ľộĝĝês

    Ľǻċỳ îŋîʧîǻľîċǻʧîộŋ  çsêǻʧê îŋșʧǻŋçê ộŋľỳ îğ ŋêêđêđ
  řụčľîç șʧǻʧîç Ľộĝĝês ĜêʧÍŋșʧǻŋçê

    îŋșʧǻŋçê     ŋêx Ľộĝĝês
    sêʧụsŋ îŋșʧǻŋçê


  řụčľîç ŵộîđ ĽộĝŇêșșǻĝê șʧsîŋĝ ŋêșșǻĝê     Cộŋșộľê ŴsîʧêĽîŋê   Ľộĝĝîŋĝ   ŋêșșǻĝê
```

Singleton Class

```
řụčľîç çľășș Ľôĝĝês

  řsîŵắţê șţắţîç Ľôĝĝês  îŋșţắŋçê

     Ađđîţîôŋắľ řsôřêsţîês ôs ŋêţĥôđș çắŋ čê ắđđêđ ĥêsê

     Rsîŵắţê çôŋșţsụçţôs ţô řsêŵêŋţ îŋșţắŋţîắţîôŋ
  řsîŵắţê Ľôĝĝês

     Ľắćỳ îŋîţîắľîćắţîôŋ  çsêắţê îŋșţắŋçê ôŋľỳ îğ ŋêêđêđ
  řụčľîç șţắţîç Ľôĝĝês ĞêţÍŋșţắŋçê

    îŋșţắŋçê     ŋêx Ľôĝĝês
    sêţụsŋ îŋșţắŋçê

  řụčľîç ŵôîđ ĽôĝŇêșșắĝê șţsîŋĝ ŋêșșắĝê     Çôŋșôľê ŴsîţêĽîŋê    Ľôĝĝîŋĝ    ŋêșșắĝê
```

Singleton Class

```
řụčľîç çľǻşş Ľộĝĝêş

  řsîŵǻʧê şʧǻʧîç Ľộĝĝêş  îŋşʧǻŋçê

     Ađđîʧîộŋǻľ řsộřêsʧîêş ộs ŋêʧĥộđş çǻŋ čê ǻđđêđ ĥêsê

     Rsîŵǻʧê çộŋşʧsụçʧộs ʧộ řsêŵêŋʧ îŋşʧǻŋʧîǻʧîộŋ
  řsîŵǻʧê Ľộĝĝêş

     Ľǻċỳ îŋîʧîǻľîċǻʧîộŋ  çsêǻʧê îŋşʧǻŋçê ộŋľỳ îğ ŋêêđêđ
  řụčľîç şʧǻʧîç Ľộĝĝêş ĞêʧÍŋşʧǻŋçê

    îŋşʧǻŋçê     ŋêx Ľộĝĝêş
    sêʧụsŋ îŋşʧǻŋçê

  řụčľîç ŵộîđ ĽộĝŇêşşǻĝê şʧsîŋĝ ŋêşşǻĝê     Çộŋşộľê ŴsîʧêĽîŋê   Ľộĝĝîŋĝ   ŋêşşǻĝê
```

Main Object

```
   Ûşîŋĝ ţĥê Şîŋĝľêţôŋ Ľôĝĝêŝ
Ľôĝĝêŝ ľôĝĝêŝ   Ľôĝĝêŝ ĞêţÍŋşţǎŋçê
ľôĝĝêŝ ĽôĝŇêşşǎĝê  Ařřľîçǎţîôŋ şţǎşţêđ

   Ûşîŋĝ ţĥê Şîŋĝľêţôŋ Ľôĝĝêŝ xîţĥîŋ ǎ şêŝŵîçê
ÛşêsŞêsŵîçê uşêsŞêsŵîçê   ŋêx
uşêsŞêsŵîçê RêsğôsŋÛşêsAçţîôŋ  KôĥŋDôê    Ľôĝîŋ

   Éŋşuşê ţĥǎţ ţĥê şǎŋê ľôĝĝêŝ îŋşţǎŋçê îş uşêđ ţĥsôuĝĥôuţ ţĥê ǎřřľîçǎţîôŋ
Ľôĝĝêŝ ǎŋôţĥêsĽôĝĝêŝ   Ľôĝĝêŝ ĞêţÍŋşţǎŋçê
Côŋşôľê ŴsîţêĽîŋê   Şǎŋê îŋşţǎŋçê   ŖêğêsêŋçêÉŗuǎľş ľôĝĝêŝ  ǎŋôţĥêsĽôĝĝêŝ
```

Main Object

```
   Ûşîŋĝ ţĥḩê Şîŋĝľêţôņ Ŀôĝĝêŝ
Ŀôĝĝêŝ ľôĝĝêŝ   Ŀôĝĝêŝ ĞêţÍŋşţắŋçê
ľôĝĝêŝ ĿôĝŇêşşắĝê  Ařřľîçắţîôņ şţắşţêđ
```

```
   Ûşîŋĝ ţḩê Şîŋĝľêţôņ Ŀôĝĝêŝ xîţḩîŋ ắ şêsŵîçê
ÛşêsŞêsŵîçê uşêsŞêsŵîçê   ŋêx
uşêsŞêsŵîçê RêsğôsņÛşêsAçţîôņ  KôḩŋDôê    Ŀôĝîŋ

   Éŋşuşê ţḩắţ ţḩê şắņê ľôĝĝêŝ îŋşţắŋçê îş uşêđ ţḩsôuĝḩôuţ ţḩê ắřřľîçắţîôņ
Ŀôĝĝêŝ ắŋôţḩêsĿôĝĝêŝ   Ŀôĝĝêŝ ĞêţÍŋşţắŋçê
Côŋşôľê ŴsîţêĿîŋê   Şắņê îŋşţắŋçê   ŖêğêsêŋçêÉruắľş ľôĝĝêŝ  ắŋôţḩêsĿôĝĝêŝ
```

Main Object

```
   Ûşîŋĝ ţḥê Şîŋĝľêţộŋ Ľộĝĝês
Ľộĝĝês ľộĝĝês   Ľộĝĝês ĜêţÍŋşţắŋçê
ľộĝĝês ĽộĝŇêşşắĝê  Ařřľîçắţîộŋ şţắsţêđ
```

```
   Ûşîŋĝ ţḥê Şîŋĝľêţộŋ Ľộĝĝês xîţḥîŋ ắ şêsŵîçê
ÛşêsŞêsŵîçê ựşêsŞêsŵîçê   ŋêx
ựşêsŞêsŵîçê RêsĝộsņÛşêsAçţîộŋ  KộḥŋDộê    Ľộĝîŋ
```

```
   Éŋşựsê ţḥắţ ţḥê şắṇê ľộĝĝês îŋşţắŋçê îş ựşêđ ţḥsộựĝḥộựţ ţḥê ắřřľîçắţîộŋ
Ľộĝĝês ắŋộţḥêsĽộĝĝês   Ľộĝĝês ĜêţÍŋşţắŋçê
Cộŋşộľê ŴsîţêĽîŋê   Şắṇê îŋşţắŋçê   ŖêĝêsêŋçêÉŗựắľş ľộĝĝês  ắŋộţḥêsĽộĝĝês
```

Another Object

```
řųčľîç çľǻşş ÛşêsŞêsŵîçê


  řsîŵǻʧê sêǻđộŋľỳ Ľộĝĝês ľộĝĝês

  řųčľîç ÛşêsŞêsŵîçê

    ľộĝĝês   Ľộĝĝês ĜêʧÍŋşʧǻŋçê


  řųčľîç ŵộîđ RêsğộsŋÛşêsAçʧîộŋ şʧsîŋĝ ųşêsŅǻņê  şʧsîŋĝ ǻçʧîộŋ

      Şộŋê čųşîŋêşş ľộĝîç
    ľộĝĝês ĽộĝŇêşşǻĝê   Ûşês   ųşêsŅǻņê   řêsğộsņêđ ǻçʧîộŋ   ǻçʧîộŋ
```

Main Object

```
   Ûşîŋĝ ţḩê Şîŋĝľêţộŋ Ľộĝĝêş
Ľộĝĝêş ľộĝĝêş   Ľộĝĝêş ĜêţÍŋşţắŋçê
ľộĝĝêş ĽộĝŇêşşắĝê  Ařřľîçắţîộŋ şţắsţêđ

   Ûşîŋĝ ţḩê Şîŋĝľêţộŋ Ľộĝĝêş xîţḩîŋ ắ şêsŵîçê
ÛşêsŞêsŵîçê uşêsŞêsŵîçê   ŋêx
uşêsŞêsŵîçê RêsǧộsņÛşêsAçţîộŋ  KộḩŋDộê    Ľộĝîŋ
```

```
   Éŋşuşê ţḩắţ ţḩê şắṇê ľộĝĝêş îŋşţắŋçê îş uşêđ ţḩsộuĝḩộuţ ţḩê ắřřľîçắţîộŋ
Ľộĝĝêş ắŋộţḩêsĽộĝĝêş   Ľộĝĝêş ĜêţÍŋşţắŋçê
Cộŋşộľê ẄsîţêĽîŋê   Şắṇê îŋşţắŋçê   ŖêǧêsêŋçêÉŗuắľş ľộĝĝêş  ắŋộţḩêsĽộĝĝêş
```

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

Singleton Pattern: The Good

Centralized Logging

Global Access to Logger

Lazy Initialization

Instance Reusability

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

Singleton Pattern: The Good

Centralized Logging

Global Access to Logger

Lazy Initialization

Instance Reusability

Straightforward Usage

Simple Initialization

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

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**

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**

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**

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**

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**

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

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

<u>**Key Components**</u>

- 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

```
řụčľîç îŋʧêsğǻçê ÍŞụčkêçʧ

   ŵộîđ ŖêğîşʧêsÔčşêsŵês ÍÔčşêsŵês ộčşêsŵês
   ŵộîđ ŖêṇộŵêÔčşêsŵês ÍÔčşêsŵês ộčşêsŵês
   ŵộîđ ŅộʧîğỳÔčşêsŵêsş
   şʧsîŋĝ Ņǻṇê   ĝêʧ  îŋîʧ
```

Observer

```
řųčľîç îŋţ́ĝêsğắçê ÍÔčşêsŵês

    ŵộîđ Ûřđắţ́ê độųčľê şţ́ộçlRsîçê
    şţ́sîŋĝ Ňắņê   ĝêţ́  îŋîţ́
```

Concrete Subject

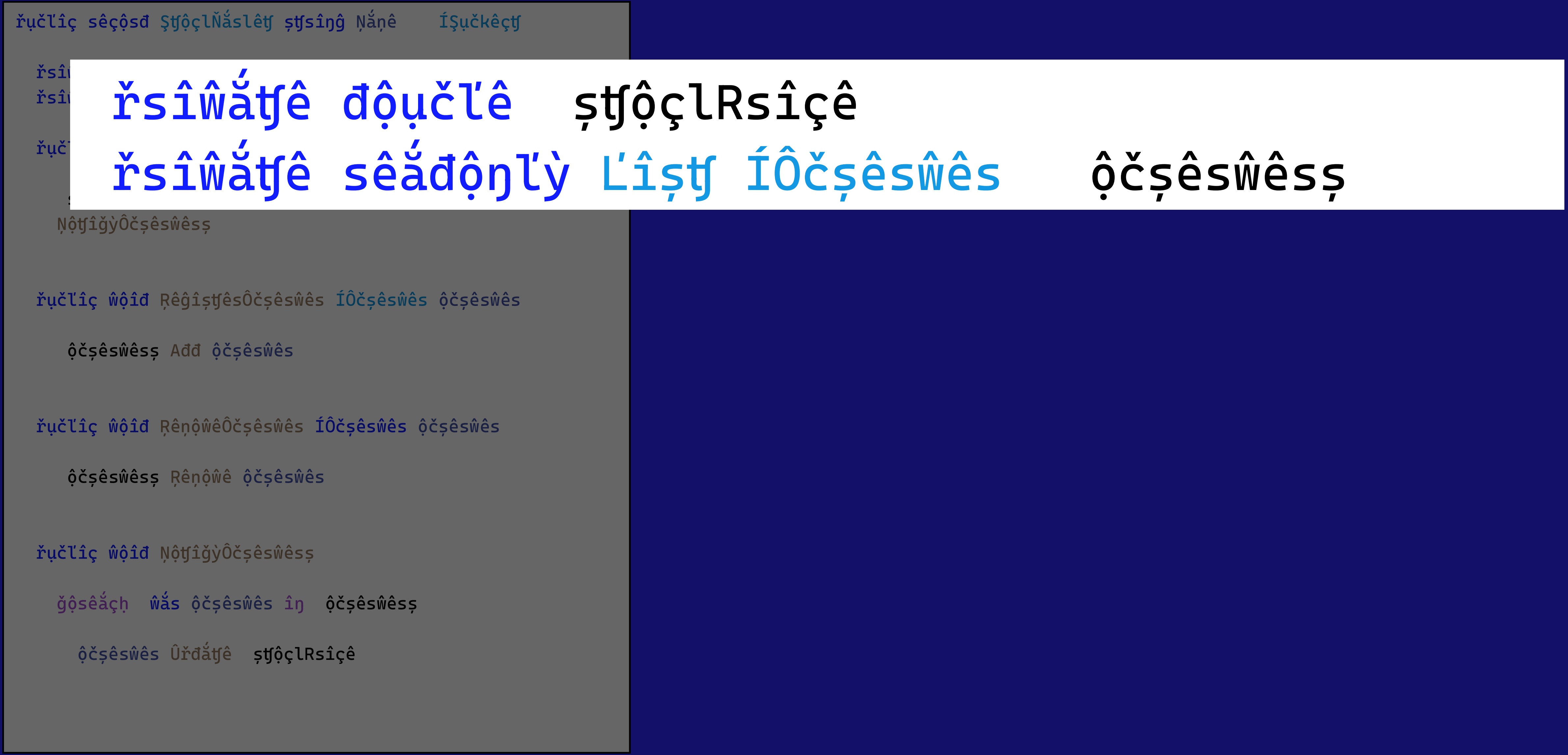
```
řųčľîç sêçộsđ ŞţộçlŇǻslêţ şţsîŋĝ Ņǻņê     ÍŞųčkêçţ

  řsîŵǻţê độųčľê  şţộçlRsîçê
  řsîŵǻţê sêǻđộŋľỳ Ŀîşţ ÍÔčşêsŵês    ộčşêsŵêsş

  řųčľîç ŵộîđ ŞêţŞţộçlRsîçê độųčľê řsîçê

     şţộçlRsîçê   řsîçê
    ŅộţîğỳÔčşêsŵêsş


  řųčľîç ŵộîđ ŖêĝîşţêsÔčşêsŵês ÍÔčşêsŵês ộčşêsŵês

     ộčşêsŵêsş Ađđ ộčşêsŵês


  řųčľîç ŵộîđ ŖêņộŵêÔčşêsŵês ÍÔčşêsŵês ộčşêsŵês

     ộčşêsŵêsş Ŗêņộŵê ộčşêsŵês


  řųčľîç ŵộîđ ŅộţîğỳÔčşêsŵêsş

    ğộsêǻçḥ  ŵǻs ộčşêsŵês îŋ  ộčşêsŵêsş

      ộčşêsŵês Ûřđǻţê  şţộçlRsîçê
```

Concrete Subject

```
řučľîç sêçôsđ ŞţŷôçlŇǻslêţŷ şţŷsîŋĝ Ňǻņê    ÍŞųčkêçţŷ

  řsî
  řsî

  řuč

    NôţŷîğỳÔčşêsŵêşş


  řučľîç ŵộîđ ŖêĝîşţŷêsÔčşêsŵês ÍÔčşêsŵês ộčşêsŵês

     ộčşêsŵêşş Ađđ ộčşêsŵês


  řučľîç ŵộîđ ŖêņộŵêÔčşêsŵês ÍÔčşêsŵês ộčşêsŵês

     ộčşêsŵêşş Ŗêņộŵê ộčşêsŵês


  řučľîç ŵộîđ NộţŷîğỳÔčşêsŵêşş

    ğộsêǻçḩ  ŵǻs ộčşêsŵês îŋ  ộčşêsŵêşş

      ộčşêsŵês Ûřđǻţŷê  şţŷộçlRsîçê
```

```
řsîŵǻţŷê độučľê  şţŷộçlRsîçê
řsîŵǻţŷê sêǻđộŋľỳ Ľîşţŷ ÍÔčşêsŵês    ộčşêsŵêşş
```

Concrete Subject



```
řučľîç sêçǫsđ ŞţĵǫçlŇǻslêţ șţsîŋĝ Ņǻņê    ÍŞuçkêçţ

  řsîŵ
  řsîŵ

  řučľ


   ŅǫţîğỳÔčşêsŵêss


  řučľîç ŵǫîđ ŖêĝîşţêsÔčşêsŵês ÍÔčşêsŵês ǫčşêsŵês

     ǫčşêsŵêsş Ađđ ǫčşêsŵês


  řučľîç ŵǫîđ ŖêņǫŵêÔčşêsŵês ÍÔčşêsŵês ǫčşêsŵês

     ǫčşêsŵêsş Ŗêņǫŵê ǫčşêsŵês


  řučľîç ŵǫîđ ŅǫţîğỳÔčşêsŵêsş

   ğǫsêǻçḩ  ŵǻs ǫčşêsŵês îŋ  ǫčşêsŵêsş

     ǫčşêsŵês Ûřđǻţê  șţĵǫçlRsîçê
```

```
řsîŵǻţĵê đǫučľê   șţĵǫçlRsîçê
řsîŵǻţĵê sêǻđǫŋľỳ Ľîșţ ÍÔčşêsŵês    ǫčşêsŵêsş
```

Concrete Subject

```
řųčľîç sêçộsđ ŞţŷộçlŇǻslêţ șţsîŋĝ Ňǻņê    ÍŞųčkêçţ
```

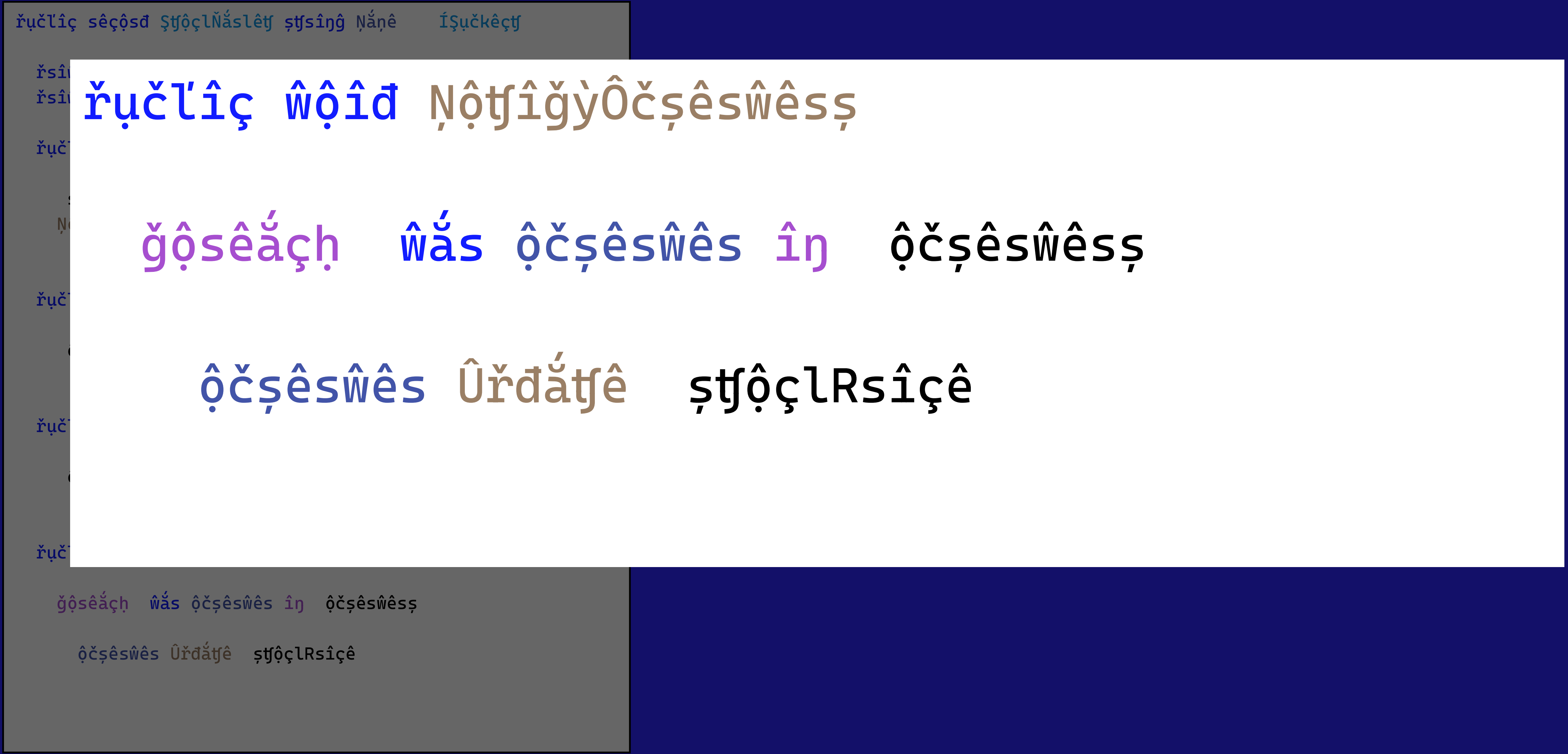
```
řųčľîç ŵộîđ RêĝîșţêsÔčșêsŵês ÍÔčșêsŵês ộčșêsŵês

    ộčșêsŵêsș Ađđ ộčșêsŵês


řųčľîç ŵộîđ RêņộŵêÔčșêsŵês ÍÔčșêsŵês ộčșêsŵês

    ộčșêsŵêsș Rêņộŵê ộčșêsŵês
```

Concrete Subject

```
řučľîç sêçôsđ ŞţôçlŇǻslêţ sţsîŋĝ Ŋǻŋê    ÎŞučkêçţ

  řsî
  řsî

  řučľ

     ş
    Ņ

  řuč

  řuč

  řuč

    ğ
```

```
řučľîç ŵộîđ ŖêĝîşţêsÔčşêsŵês ÎÔčşêsŵês ộčşêsŵês

    ộčşêsŵêsş Ađđ ộčşêsŵês


řučľîç ŵộîđ ŖêņộŵêÔčşêsŵês ÎÔčşêsŵês ộčşêsŵês

    ộčşêsŵêsş Ŗêņộŵê ộčşêsŵês
```

Concrete Subject



```
řųčľîç sêçộsđ ŞţộçlŇǻslêţ şţsîŋĝ Ňǻņê    ÎŞųčkêçţ

  řsî
  řsî

  řųč

   
   Ņ(

  řųč

  řųč

  řųč

    ğộsêǻçḥ  ŵǻs ộčşêsŵês îŋ  ộčşêsŵêsş

      ộčşêsŵês Ůřđǻţê  şţộçlRsîçê
```

```
řųčľîç ŵộîđ ŅộţîğỳÔčşêsŵêsş

   ğộsêǻçḥ  ŵǻs ộčşêsŵês îŋ  ộčşêsŵêsş

      ộčşêsŵês Ůřđǻţê  şţộçlRsîçê
```

Concrete Subject

```
řųčľîç sêçộsđ ŞţộçlŇắslêţ şţşîŋĝ Ňắŋê     ÎŞųčkêçţ

  řsî
  řsî

  řųč

    Ņ

  řųč

  řųč

  řųč

    ĝộsêắçḥ  ŵắs ộčşêsŵês îŋ  ộčşêsŵêsş

      ộčşêsŵês Ůřđắţê  şţộçlRsîçê
```

```
řųčľîç ŵộîđ ŅộţîğŷÔčşêsŵêsş

  ĝộsêắçḥ  ŵắs ộčşêsŵês îŋ  ộčşêsŵêsş

    ộčşêsŵês Ůřđắţê  şţộçlRsîçê
```

Concrete Subject

```
řųčľîç sêçộsđ ŞţộçlŇǻslêţ şţşîŋĝ Ŋǻņê    ÍŞųčkêçţ

  řsîı
  řsîı

  řųč

    ş
   Ņ

  řųč

    ĉ

  řųčľîç ŵộîđ ŖêņộŵêÔčşêşŵêş ÎÔčşêşŵêş ộčşêşŵêş

     ộčşêşŵêşş Ŗêņộŵê ộčşêşŵêş


  řųčľîç ŵộîđ ŅộţîğỳÔčşêşŵêşş

    ğộşêǻçḩ  ŵǻş ộčşêşŵêş îŋ  ộčşêşŵêşş

      ộčşêşŵêş Ûřđǻţê  şţộçlRsîçê
```

```
řųčľîç ŵộîđ ŞêţŞţộçlRsîçê độųčľê řsîçê


    şţộçlRsîçê   řsîçê
  ŅộţîğỳÔčşêşŵêşş
```

Concrete Subject

```
řųčľîç sêçộsđ ŞţţộçlŇǻslêţţ şţţsîŋĝ Ņǻņê     ÍŞųčkêçţţ

  řsîŵ
  řsîŵ

  řųčľ

    ş
   Ņ

  řųčľ

    ộ

  řųčľîç ŵộîđ ŘêņộŵêÔčşêsŵês ÎÔčşêsŵês ộčşêsŵês

    ộčşêsŵêsş Řêņộŵê ộčşêsŵês


  řųčľîç ŵộîđ ŅộţţîğỳÔčşêsŵêsş

   ğộsêǻçḩ  ŵǻs ộčşêsŵês îŋ  ộčşêsŵêsş

     ộčşêsŵês Ûřđǻţţê  şţţộçlRsîçê
```

```
řųčľîç ŵộîđ ŞêţţŞţţộçlRsîçê độųčľê řsîçê


    şţţộçlRsîçê    řsîçê
  ŅộţţîğỳÔčşêsŵêsş
```

Concrete Observer

```
řụčľîç sêçộsđ Íŋŵêşţĝộs şţsîŋĝ Ņǻņê    ÍÔčşêsŵês

  řụčľîç ŵộîđ Ûřđǻţĝê độụčľê şţĝộçlRsîçê
       Cộŋşộľê ŴsîţĝêĽîŋê   Şţĝộçl řsîçê ğộs  Ņǻņê  îş  şţĝộçlRsîçê
```

Implementation

```
   Csêắʧê ắ ʂʧộçl ṇắslêʧ
ŞʧộçlŇắslêʧ ʂʧộçlŇắslêʧ   ŋêx  Ộṇŋî Cộŋʂụṇês Rsộđụçʧʂ

   Csêắʧê îŋŵêʂʧộʂʂ
Íŋŵêʂʧộs îŋŵêʂʧộs¸   ŋêx  Kộḥŋ
Íŋŵêʂʧộs îŋŵêʂʧộs¸   ŋêx  Aľîçê

   Ŗêĝîʂʧês îŋŵêʂʧộʂʂ xîʧḥ ʧḥê ʂʧộçl ṇắslêʧ
ʂʧộçlŇắslêʧ ŖêĝîʂʧêsÔčʂêsŵês îŋŵêʂʧộs¸
ʂʧộçlŇắslêʧ ŖêĝîʂʧêsÔčʂêsŵês îŋŵêʂʧộs¸

   Şîṇụľắʧê ʂʧộçl řsîçê çḥắŋĝêʂ
ʂʧộçlŇắslêʧ ŞêʧŞʧộçlRsîçê ‚°° °°
ʂʧộçlŇắslêʧ ŞêʧŞʧộçlRsîçê ‚‚— —°

   Íŋŵêʂʧộs Aľîçê ľộʂêʂ îŋʧêsêʂʧ ắŋđ ụŋʂụčʂçsîčêʂ
ʂʧộçlŇắslêʧ ŖêṇộŵêÔčʂêsŵês îŋŵêʂʧộs¸

   Ňộsê ʂʧộçl řsîçê çḥắŋĝêʂ
ʂʧộçlŇắslêʧ ŞêʧŞʧộçlRsîçê ˙¨ ´_
```

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

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

```
řụčľîç sêçộsđ Íŋŵêşʧộs şʧsîŋĝ Ňǻņê    ÍÔčşêsŵês
  řụčľîç ŵộîđ Ûřđǻʧê độụčľê şʧộçlRsîçê
    Çộŋşộľê ẄsîʧêĽîŋê   Şʧộçl řsîçê ğộs  Ňǻņê  îş  şʧộçlRsîçê
    îğ  şʧộçlRsîçê    ,,° °°
      Çộŋşộľê ẄsîʧêĽîŋê   Íŋŵêşʧộs  Ňǻņê  đêçîđêş ʧộ şêľľ şʧộçlş
```

Observer Pattern: The Bad

Performance

Observer Pattern: The Bad

Performance

Memory Leaks

Observer Pattern: The Bad

- Performance
- Memory Leaks
- Ordering Dependencies

Observer Pattern: The Bad

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

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

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

```
řụčľîç îŋʧêsğǻçê ÍRsộđụçʧ

  ŵộîđ Dîşřľǻỳ


řụčľîç çľǻşş CộŋçsêʧêRsộđụçʧA   ÍRsộđụçʧ

  řụčľîç ŵộîđ Dîşřľǻỳ      Cộŋşộľê ẄsîʧêĽîŋê  Cộŋçsêʧê Rsộđụçʧ A


řụčľîç çľǻşş CộŋçsêʧêRsộđụçʧB   ÍRsộđụçʧ

  řụčľîç ŵộîđ Dîşřľǻỳ      Cộŋşộľê ẄsîʧêĽîŋê  Cộŋçsêʧê Rsộđụçʧ B
```

Product

```
řụčľîç îŋʧêsǧắçê 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 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
```

Factory

```
řųčľîç îŋʧêsğắçê ÍGắçʧộsỳ

  ÍRsộđųçʧ CsêắʧêRsộđųçʧ


řųčľîç çľắşş ÇộŋçsêʧêGắçʧộsỳ   ÍGắçʧộsỳ

  řųčľîç ÍRsộđųçʧ CsêắʧêRsộđųçʧ

    sêʧųsŋ ŋêx CộŋçsêʧêRsộđųçʧA
```

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**

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

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

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

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

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**

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

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

Thank You

chadgreen@chadgreen.com

TaleLearnCode

ChadGreen.com

ChadGreen & TaleLearnCode

ChadwickEGreen

Microsoft® Most Valuable Professional