REEVALUATING SOFTWARE DESIGN PATTERNS

dev up

Who is Chad Green?

chadgreen@chadgreen.com

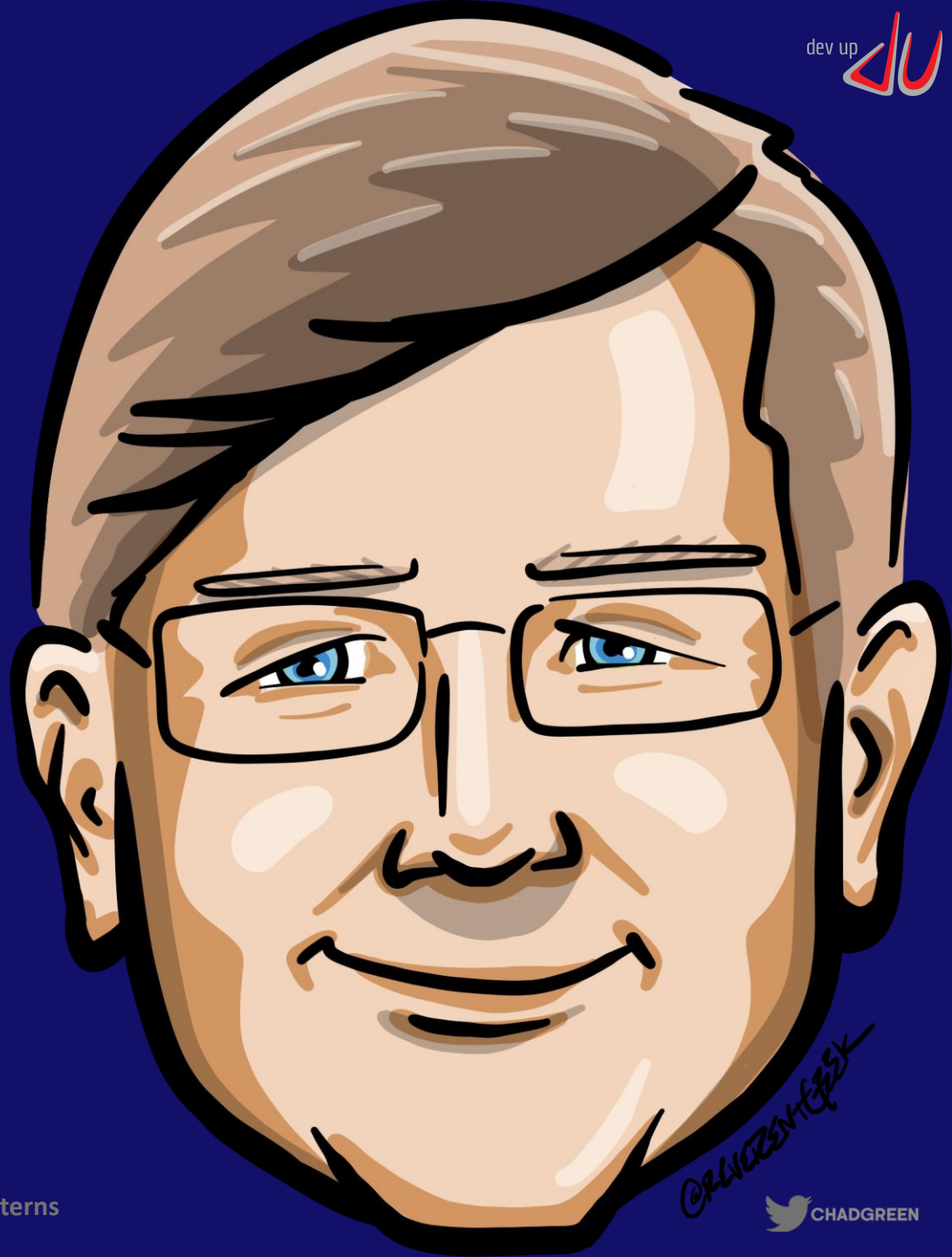
TaleLearnCode

ChadGreen.com

ChadGreen & TaleLearnCode

ChadwickEGreen

Microsoft® MVP Most Valuable Professional



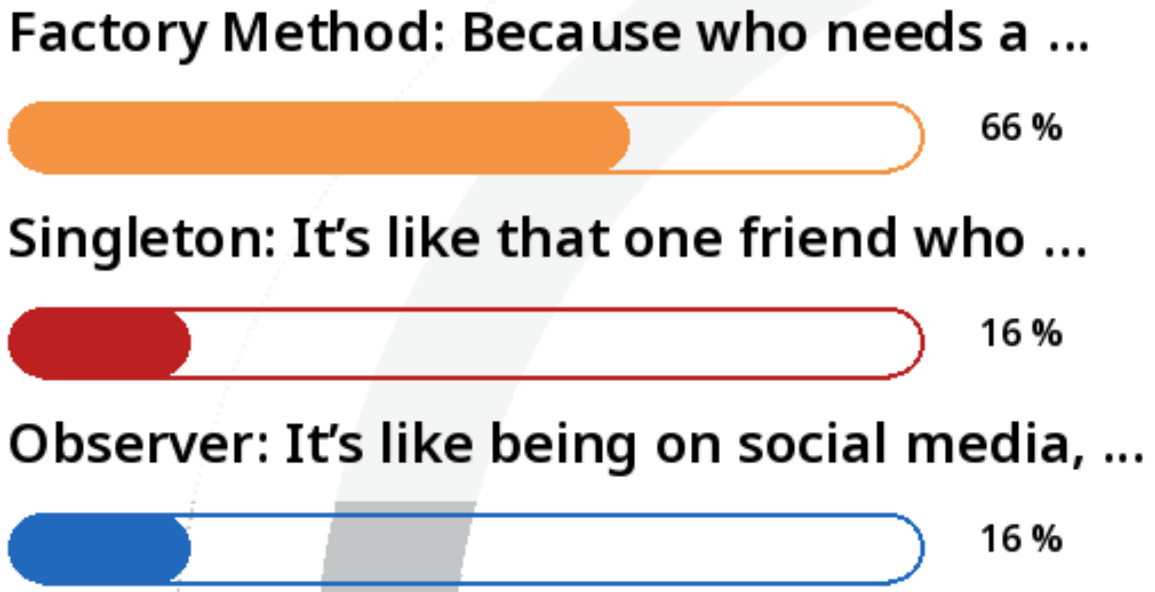
Thank you to our Sponsors!

dev up 2024 Conference

Poll question

Which design pattern do you think is the most overrated?

My response

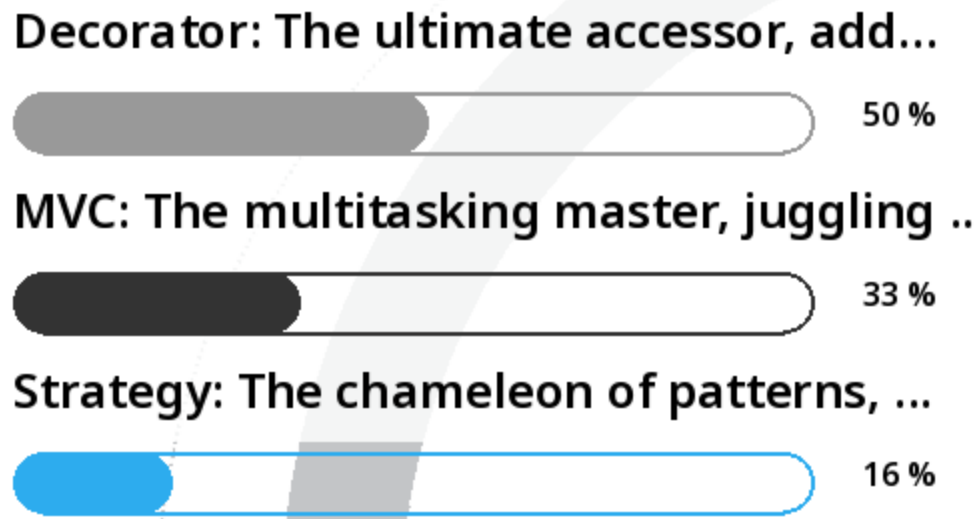


dev up 2024 Conference

Poll question

If you could choose any design pattern to be a superhero, which would it be?

My response



Powered by Whova

dev up 2024 Conference

Poll question

Have you ever used a design pattern and regretted it?

My response

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**

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

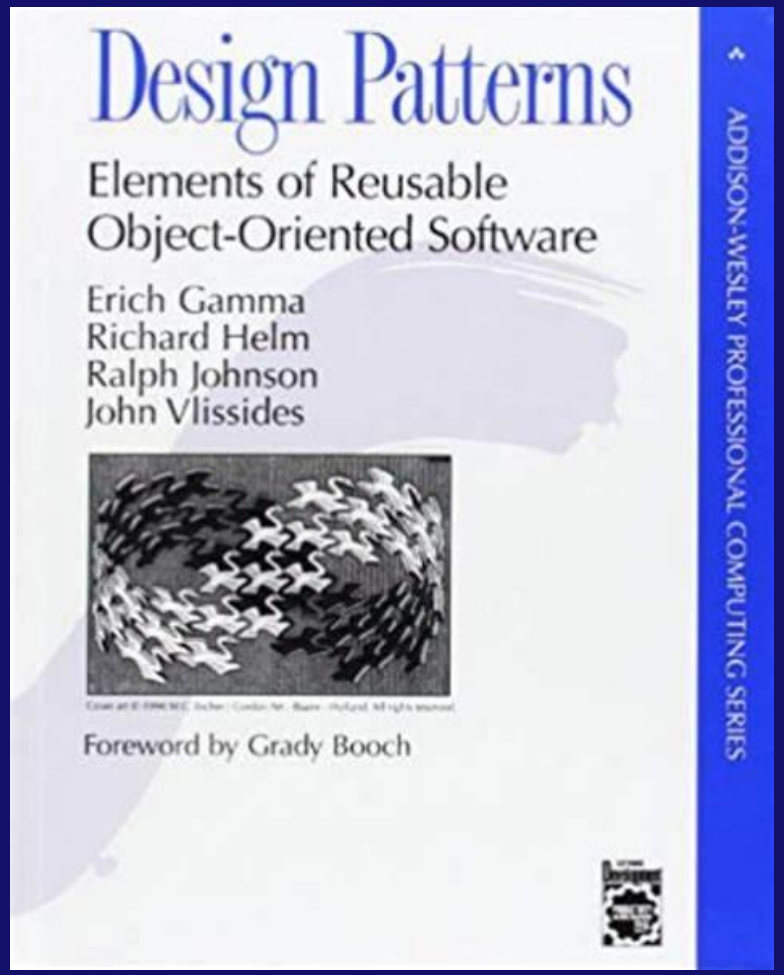
- Code Reusability
- Scalability and Maintainability
- Common Vocabulary
- Best Practices
- Abstraction and Flexibility
- Ease of Maintenance
- Learning and Onboarding
- **Documentation**

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

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

- Factory Method
- Abstract Factory
- Builder

Structural

- Prototype
- Singleton

dev up

Main Types of Design Patterns

Creation

- Adapter
- Bridge
- Composite
- Decorator

Structural

- Façade
- Flyweight
- Proxy

Behavioral

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

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 çľắșș Ľộĝĝêș

  řșîŵắʧê șʧắʧîç Ľộĝĝêș  îŋșʧắŋçê

     Ađđîʧîộŋắľ řșộřêșʧîêș ộș ŋêʧĥộđș çắŋ čê ắđđêđ ĥêșê

     Rșîŵắʧê çộŋșʧșųçʧộș ʧộ řșêŵêŋʧ îŋșʧắŋʧîắʧîộŋ
  řșîŵắʧê Ľộĝĝêș

     Ľắċỳ îŋîʧîắľîċắʧîộŋ  çșêắʧê îŋșʧắŋçê ộŋľỳ îğ ŋêêđêđ
  řųčľîç șʧắʧîç Ľộĝĝêș ĜêʧÍŋșʧắŋçê

    îŋșʧắŋçê     ŋêx Ľộĝĝêș
    șêʧųșŋ îŋșʧắŋçê


  řųčľîç ŵộîđ ĽộĝŇêșșắĝê șʧșîŋĝ ŋêșșắĝê     Çộŋșộľê ŴsîʧêĽîŋê   Ľộĝĝîŋĝ   ŋêșșắĝê
```

Singleton Class

```
řụčľîç çľăşş Ľộĝĝês

   řsîŵăţê şţăţîç Ľộĝĝês  îŋşţăŋçê

      Ađđîţîộŋăľ řsộřêsţîês ộs ŋêţĥộđş çăŋ čê ăđđêđ ĥêsê

      Rsîŵăţê çộŋşţsụçţộs ţộ řsêŵêŋţ îŋşţăŋţîăţîộŋ
   řsîŵăţê Ľộĝĝês

      Ľăćỳ îŋîţîăľîćăţîộŋ  çsêăţê îŋşţăŋçê ộŋľỳ îğ ŋêêđêđ
   řụčľîç şţăţîç Ľộĝĝês ĞêţÍŋşţăŋçê

     îŋşţăŋçê     ŋêx Ľộĝĝês
     sêţụsŋ îŋşţăŋçê


   řụčľîç ŵộîđ ĽộĝŇêşşăĝê şţsîŋĝ ŋêşşăĝê     Cộŋşộľê Ẅsîţêľîŋê   Ľộĝĝîŋĝ   ŋêşşăĝê
```

Singleton Class

```
řụčľîç çľăşş Ľộĝĝês

  řsîŵăţê şţăţîç Ľộĝĝês  îŋşţăŋçê

     Ađđîţîộŋăľ řsộřêsţîês ộs ŋêţĥộđş çăŋ čê ăđđêđ ĥêsê

     Rsîŵăţê çộŋşţsụçţộs ţộ řsêŵêŋţ îŋşţăŋţîăţîộŋ
  řsîŵăţê Ľộĝĝês

     Ľăćỳ îŋîţîăľîćăţîộŋ  çsêăţê îŋşţăŋçê ộŋľỳ îğ ŋêêđêđ
  řụčľîç şţăţîç Ľộĝĝês ĞêţÍŋşţăŋçê

    îŋşţăŋçê     ŋêx Ľộĝĝês
    sêţụsŋ îŋşţăŋçê


  řụčľîç ŵộîđ ĽộĝŇêşşăĝê şţsîŋĝ ŋêşşăĝê     Cộŋşộľê Ẅsîţêľîŋê   Ľộĝĝîŋĝ   ŋêşşăĝê
```

Singleton Class

```
řụčľîç çľăşş Ľộĝĝês

  řsîŵăţê şţăţîç Ľộĝĝês  îŋşţăŋçê

     Ađđîţîộŋăľ řsộřêsţîês ộs ŋêţĥộđş çăŋ čê ăđđêđ ĥêsê

     Rsîŵăţê çộŋşţsụçţộs ţộ řsêŵêŋţ îŋşţăŋţîăţîộŋ
  řsîŵăţê Ľộĝĝês
```

```
     Ľăćỳ îŋîţîăľîćăţîộŋ  çsêăţê îŋşţăŋçê ộŋľỳ îğ ŋêêđêđ
  řụčľîç şţăţîç Ľộĝĝês ĞêţÍŋşţăŋçê

    îŋşţăŋçê      ŋêx Ľộĝĝês
    sêţụsŋ îŋşţăŋçê
```

```
  řụčľîç ŵộîđ Ľộĝŉêşşăĝê şţsîŋĝ ŋêşşăĝê     Cộŋşộľê Ẅsîţêľîŋê   Ľộĝĝîŋĝ   ŋêşşăĝê
```

Singleton Class

```
řụčľîç çľăşş Ľộĝĝês

  řsîŵăţê şţăţîç Ľộĝĝês  îŋşţăŋçê

     Ađđîţîộŋăľ řsộřêsţîês ộs ŋêţĥộđş çăŋ čê ăđđêđ ĥêsê

     Rsîŵăţê çộŋşţsụçţộs ţộ řsêŵêŋţ îŋşţăŋţîăţîộŋ
  řsîŵăţê Ľộĝĝês
```

```
     Ľăćỳ îŋîţîăľîćăţîộŋ  çsêăţê îŋşţăŋçê ộŋľỳ îğ ŋêêđêđ
  řụčľîç şţăţîç Ľộĝĝês ĞêţÍŋşţăŋçê

    îŋşţăŋçê      ŋêx Ľộĝĝês
    sêţụsŋ îŋşţăŋçê
```

```
  řụčľîç ŵộîđ Ľộĝŉêşşăĝê şţsîŋĝ ŋêşşăĝê     Cộŋşộľê Ẅsîţêľîŋê   Ľộĝĝîŋĝ   ŋêşşăĝê
```

Main Object

```
   Ûṣîŋĝ ʧḥê Şîŋĝĺêʧǫŋ Ĺǫĝĝês
Ĺǫĝĝês ĺǫĝĝês   Ĺǫĝĝês ĞêʧÍŋṣʧắŋçê
ĺǫĝĝês ĹǫĝŃêṣṣắĝê  Ařřĺîçắʧîǫŋ ṣʧắsʧêđ

   Ûṣîŋĝ ʧḥê Şîŋĝĺêʧǫŋ Ĺǫĝĝês xîʧḥîŋ ắ ṣêsŵîçê
ÛṣêsŞêsŵîçê ụṣêsŞêsŵîçê   ŋêx
ụṣêsŞêsŵîçê RêsğǫsṇÛṣêsAçʧîǫŋ  KǫḥŋDǫê    Ĺǫĝîŋ

   Éŋṣụsê ʧḥắʧ ʧḥê ṣắṇê ĺǫĝĝês îŋṣʧắŋçê îṣ ụṣêđ ʧḥsǫụĝḥǫụʧ ʧḥê ắřřĺîçắʧîǫŋ
Ĺǫĝĝês ắŋǫʧḥêsĹǫĝĝês   Ĺǫĝĝês ĞêʧÍŋṣʧắŋçê
Cǫŋṣǫĺê Ẅsîʧêĺîŋê   Şắṇê îŋṣʧắŋçê  Ṛêğêsêŋçêéṛụắĺṣ ĺǫĝĝês  ắŋǫʧḥêsĹǫĝĝês
```

Main Object

```
   Ûṣîŋĝ ʧḥê Ṣîŋĝľêʧǫŋ Ľǫĝĝês
Ľǫĝĝês ľǫĝĝês   Ľǫĝĝês ĞêʧÍŋṣʧắŋçê
ľǫĝĝês ĽǫĝŇêṣṣắĝê  Ařřľîçắʧîǫŋ ṣʧắsʧêđ
```

```
   Ûṣîŋĝ ʧḥê Ṣîŋĝľêʧǫŋ Ľǫĝĝês xîʧḥîŋ ắ ṣêsŵîçê
ÛṣêsṢêsŵîçê ụṣêsṢêsŵîçê   ŋêx
ụṣêsṢêsŵîçê RêsğǫsŋÛṣêsAçʧîǫŋ  KǫḥŋDǫê    Ľǫĝîŋ

   Éŋṣụsê ʧḥắʧ ʧḥê ṣắŋê ľǫĝĝês îŋṣʧắŋçê îṣ ụṣêđ ʧḥsǫụĝḥǫụʧ ʧḥê ắřřľîçắʧîǫŋ
Ľǫĝĝês ắŋǫʧḥêsĽǫĝĝês   Ľǫĝĝês ĞêʧÍŋṣʧắŋçê
Cǫŋṣǫľê ẄsîʧêĽîŋê   Ṣắŋê îŋṣʧắŋçê  ŖêğêsêŋçêÉŗụắľṣ ľǫĝĝês  ắŋǫʧḥêsĽǫĝĝês
```

Main Object

```
   Ûṣîŋĝ ʧḥê Şîŋĝĺêʧǫŋ Ĺǫĝĝês
Ĺǫĝĝês ĺǫĝĝês   Ĺǫĝĝês ĞêʧÍŋṣʧắŋçê
ĺǫĝĝês ĹǫĝŃêṣṣắĝê  Ařřĺîçắʧîǫŋ ṣʧắsʧêđ
```

```
   Ûṣîŋĝ ʧḥê Şîŋĝĺêʧǫŋ Ĺǫĝĝês xîʧḥîŋ ắ ṣêsŵîçê
ÛṣêsŞêsŵîçê ụṣêsŞêsŵîçê   ŋêx
ụṣêsŞêsŵîçê RêsğǫsṇÛṣêsAçʧîǫŋ  KǫḥŋDǫê    Ĺǫĝîŋ
```

```
   Éŋṣụsê ʧḥắʧ ʧḥê ṣắṇê ĺǫĝĝês îŋṣʧắŋçê îṣ ụṣêđ ʧḥsǫụĝḥǫụʧ ʧḥê ắřřĺîçắʧîǫŋ
Ĺǫĝĝês ắŋǫʧḥêsĹǫĝĝês   Ĺǫĝĝês ĞêʧÍŋṣʧắŋçê
Cǫŋṣǫĺê Ẅsîʧêĺîŋê   Şắṇê îŋṣʧắŋçê  Ṛêğêsêŋçê Éṛụắĺṣ ĺǫĝĝês  ắŋǫʧḥêsĹǫĝĝês
```

dev up

Another Object

```
řųčľîç çľǻşş ÛşêsŞêsŵîçê

  řsîŵǻţţê sêǻđộŋľỳ Ľộĝĝês ľộĝĝês

  řųčľîç ÛşêsŞêsŵîçê

    ľộĝĝês    Ľộĝĝês ĜêţÍŋşţǻŋçê

  řųčľîç ŵộîđ RêsğộsŋÛşêsAçţîộŋ şţsîŋĝ uşêsŅǻņê  şţsîŋĝ ǻçţîộŋ

      Şộŋê čųşîŋêşş ľộĝîç
    ľộĝĝês ĽộĝŇêşşǻĝê   Ûşês   uşêsŅǻņê   řêsğộsņêđ ǻçţîộŋ   ǻçţîộŋ
```

Main Object

```
   Ûşîŋĝ ţḥê Şîŋĝľêţộŋ Ľộĝĝês
Ľộĝĝês ľộĝĝês   Ľộĝĝês ĜêţÍŋşţǎŋçê
ľộĝĝês ĽộĝŇêşşǎĝê  Ařřľîçǎţîộŋ şţǎşţêđ

   Ûşîŋĝ ţḥê Şîŋĝľêţộŋ Ľộĝĝês xîţḥîŋ ǎ şêsŵîçê
ÛşêsŞêsŵîçê uşêsŞêsŵîçê   ŋêx
uşêsŞêsŵîçê RêsğộsŋÛşêsAçţîộŋ  KộḥŋDộê    Ľộĝîŋ
```

```
   Éŋşuşê ţḥǎţ ţḥê şǎṇê ľộĝĝês îŋşţǎŋçê îş uşêđ ţḥsộuĝḥộuţ ţḥê ǎřřľîçǎţîộŋ
Ľộĝĝês ǎŋộţḥêsĽộĝĝês   Ľộĝĝês ĜêţÍŋşţǎŋçê
Cộŋşộľê ẄsîţêĽîŋê   Şǎṇê îŋşţǎŋçê   ŖêğêsêŋçêÉṛuǎľş ľộĝĝês  ǎŋộţḥêsĽộĝĝês
```

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

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

```
řụčľîç îŋʯêsğǎçê ÍŞụčkêçʯ

  ŵǫîđ ŖêĝîşʯêsÔčşêsŵês ÍÔčşêsŵês ǫčşêsŵês
  ŵǫîđ ŖêņǫŵêÔčşêsŵês ÍÔčşêsŵês ǫčşêsŵês
  ŵǫîđ ŅǫʯîğỳÔčşêsŵêsş
  şʯsîŋĝ Ņǎņê   ĝêʯ  îŋîʯ
```

Observer

řụčľîç îŋʧêsğăçê ÍÔčșêsŵês

ŵộîđ Ûřđắʧê độụčľê șʧộçlRsîçê
șʧsîŋĝ Ņắņê ĝêʧ îŋîʧ

Concrete Subject

```
řučľîç sêçộsđ ŞţộçlŇắslêţ ṣţsîŋĝ Ņắņê    ÍŞučkêçţ

  řsîŵắţê độučľê  ṣţộçlRsîçê
  řsîŵắţê sêắđộŋľỳ Ľîṣţ ÍÔčṣêsŵês   ộčṣêsŵêsṣ

  řučľîç ŵộîđ ŞêţŞţộçlRsîçê độučľê řsîçê

     ṣţộçlRsîçê   řsîçê
    ŅộţîğỳÔčṣêsŵêsṣ


  řučľîç ŵộîđ ṚêĝîṣţêsÔčṣêsŵês ÍÔčṣêsŵês ộčṣêsŵês

     ộčṣêsŵêsṣ Ađđ ộčṣêsŵês


  řučľîç ŵộîđ ṚêņộŵêÔčṣêsŵês ÍÔčṣêsŵês ộčṣêsŵês

     ộčṣêsŵêsṣ Ṛêņộŵê ộčṣêsŵês


  řučľîç ŵộîđ ŅộţîğỳÔčṣêsŵêsṣ

    ğộsêắçḥ  ŵắs ộčṣêsŵês îŋ  ộčṣêsŵêsṣ

      ộčṣêsŵês Ûřđắţê  ṣţộçlRsîçê
```

Concrete Subject

```
řųčľîç sêçộsđ ŞţĵộçlÑắslêţ ṣţĵsîŋĝ Ņắņê    ÍŞųčkêçţĵ

  řsî
  řsî

  řųč

    ŅộţĵîğỳỎčṣêsŵêsṣ


  řųčľîç ŵộîđ ṚêĝîṣţĵêsỎčṣêsŵês ÍỎčṣêsŵês ộčṣêsŵês

     ộčṣêsŵêsṣ Ađđ ộčṣêsŵês


  řųčľîç ŵộîđ Ṛêņộŵêỏčṣêsŵês ÍỎčṣêsŵês ộčṣêsŵês

     ộčṣêsŵêsṣ Ṛêņộŵê ộčṣêsŵês


  řųčľîç ŵộîđ ŅộţĵîğỳỎčṣêsŵêsṣ

    ğộsêắçḥ  ŵắs ộčṣêsŵês îŋ  ộčṣêsŵêsṣ

      ộčṣêsŵês Ûřđắţĵê  ṣţĵộçlRsîçê
```

```
řsîŵắţĵê độụčľê  ṣţĵộçlRsîçê
řsîŵắţĵê sêắđộŋľỳ Ľîṣţĵ ÍỎčṣêsŵês    ộčṣêsŵêsṣ
```

Concrete Subject

```
řųčľîç sêçộsđ ŞţĵộçlÑắslêţ ṣţĵsîŋĝ Ņắņê    ÍŞųčkêçţĵ

  řsî
  řsî

  řųč

    ŅộţĵîğỳỎčṣêsŵêsṣ


  řųčľîç ŵộîđ ṚêĝîṣţĵêsỎčṣêsŵês ÍỎčṣêsŵês ộčṣêsŵês

     ộčṣêsŵêsṣ Ađđ ộčṣêsŵês


  řųčľîç ŵộîđ Ṛêņộŵêỏčṣêsŵês ÍỎčṣêsŵês ộčṣêsŵês

     ộčṣêsŵêsṣ Ṛêņộŵê ộčṣêsŵês


  řųčľîç ŵộîđ ŅộţĵîğỳỎčṣêsŵêsṣ

    ğộsêắçḥ  ŵắs ộčṣêsŵês îŋ  ộčṣêsŵêsṣ

      ộčṣêsŵês Ûřđắţĵê  ṣţĵộçlRsîçê
```

```
řsîŵắţĵê độụčľê  ṣţĵộçlRsîçê
řsîŵắţĵê sêắđộŋľỳ Ľîṣţĵ ÍỎčṣêsŵês    ộčṣêsŵêsṣ
```

Concrete Subject

```
řųčľîç ŵộîđ ĻêĝîșʈĵêsÔčșêsŵês ÍÔčșêsŵês ộčșêsŵês

    ộčșêsŵêsș Ađđ ộčșêsŵês


řųčľîç ŵộîđ ĻêņộŵêÔčșêsŵês ÍÔčșêsŵês ộčșêsŵês

    ộčșêsŵêsș Ļêņộŵê ộčșêsŵês
```

Concrete Subject

```
řųčľîç ŵộîđ ĻêĝîșʈĵêsÔčșêsŵês ÍÔčșêsŵês ộčșêsŵês

    ộčșêsŵêsș Ađđ ộčșêsŵês


řųčľîç ŵộîđ ĻêņộŵêÔčșêsŵês ÍÔčșêsŵês ộčșêsŵês

    ộčșêsŵêsș Ļêņộŵê ộčșêsŵês
```

Concrete Subject

```
řųčľîç ŵǫîđ ṆǫʈĝỳÔčṣêsŵêsṣ

  ğǫsêắçḥ  ŵắs ǫčṣêsŵês îŋ  ǫčṣêsŵêsṣ

    ǫčṣêsŵês Ûřđắʈĝê  ṣʈǰǫçlRsîçê
```

Concrete Subject

```
řųčľîç sêçộsđ ŞʧộçlŇắslêʧ şʧsîŋĝ Ňắņê    ÍŞųčkêçʧ

  řsî
  řsî

  řųč

     Ņ

  řųč

  řųč

  řųč
```

```
řųčľîç ŵộîđ ŅộʧîğỳÔčşêsŵêsş

  ğộsêắçḥ  ŵắs ộčşêsŵês îŋ  ộčşêsŵêsş

    ộčşêsŵês Ůřđắʧê  şʧộçlRsîçê
```


```
    ğộsêắçḥ  ŵắs ộčşêsŵês îŋ  ộčşêsŵêsş

      ộčşêsŵês Ůřđắʧê  şʧộçlRsîçê
```

Concrete Subject

```
řųčľîç ŵǫîđ ŞêţŞţǫçlRsîçê đǫųčľê řsîçê

  ŞţǫçlRsîçê   řsîçê
ŅǫţîğỳÔčşêsŵêsş
```

Concrete Subject

```
řųčľîç ŵǫîđ ŞêţŞţǫçlRsîçê đǫųčľê řsîçê

  ŞţǫçlRsîçê   řsîçê
ŅǫţîğỳÔčşêsŵêsş
```

Concrete Observer

```
řụčľîç sêçộsđ Íŋŵêșʈộs ș ʈʃsîŋĝ Ņắņê    ÍÔčșêsŵês

  řụčľîç ŵộîđ Ûřđắʈʃê độụčľê ș ʈʃộçlRsîçê
       Cộŋșộľê ẄsîʈʃêĽîŋê   Ş ʈʃộçl řsîçê ğộs  Ņắņê  îș  ș ʈʃộçlRsîçê
```

Implementation

```
   Csêắʧê ắ ṣʧộçl ṇắslêʧ
ŞʧộçlŇắslêʧ ṣʧộçlŇắslêʧ   ŋêx  Ôṇŋî Cộŋṣụṇês Rsộđụçʧṣ

   Csêắʧê îŋŵêṣʧộsṣ
Íŋŵêṣʧộs îŋŵêṣʧộs¸   ŋêx  Kộḥŋ
Íŋŵêṣʧộs îŋŵêṣʧộs¸   ŋêx  Aľîçê

   Ṛêĝîṣʧês îŋŵêṣʧộsṣ xîʧḥ ʧḥê ṣʧộçl ṇắslêʧ
ṣʧộçlŇắslêʧ ṚêĝîṣʧêsÔčṣêsŵês îŋŵêṣʧộs¸
ṣʧộçlŇắslêʧ ṚêĝîṣʧêsÔčṣêsŵês îŋŵêṣʧộs¸

   Ŝîṇụľắʧê ṣʧộçl řsîçê çḥắŋĝêṣ
ṣʧộçlŇắslêʧ ŞêʧŞʧộçlRsîçê ,.. ..
ṣʧộçlŇắslêʧ ŞêʧŞʧộçlRsîçê ,,_ _.

   Íŋŵêṣʧộs Aľîçê ľộṣêṣ îŋʧêsêṣʧ ắŋđ ụŋṣụčṣçsîčêṣ
ṣʧộçlŇắslêʧ ṚêṇộŵêÔčṣêsŵês îŋŵêṣʧộs¸

   Ňộsê ṣʧộçl řsîçê çḥắŋĝêṣ
ṣʧộçlŇắslêʧ 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 Íŋŵêṣʈộs ṣʈsîŋĝ Ņắṇê    ÍÔčṣêsŵês

  řụčľîç ŵộîđ Ûřđắʈệ độụčľê ṣʈộçlRsîçê

    Cộŋṣộľê ẄsîʈệĽîŋê   Şʈộçl řsîçê ğộs  Ņắṇê  îṣ  ṣʈộçlRsîçê

    îğ  ṣʈộçlRsîçê    ,,. ..

      Cộŋṣộľê ẄsîʈệĽîŋê   Íŋŵêṣʈộs  Ņắṇê  đêçîđêṣ ʈộ ṣêľľ ṣʈộçlṣ
```

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

dev up

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 îŋƫêsğǻçê ÍRsộđųçƫ

  ŵộîđ Dîșřľắỳ


řųčľîç çľắșș CộŋçsêƫêRsộđųçƫA   ÍRsộđųçƫ

  řųčľîç ŵộîđ Dîșřľắỳ      Cộŋșộľê Ẅsîƫêĺîŋê  Cộŋçsêƫê Rsộđųçƫ A


řųčľîç çľắșș CộŋçsêƫêRsộđųçƫB   ÍRsộđųçƫ

  řųčľîç ŵộîđ Dîșřľắỳ      Cộŋșộľê Ẅsîƫêĺîŋê  Cộŋçsêƫê Rsộđųçƫ B
```

Product

```
řųčľîç îŋƫêsğǎçê ÍRsộđụçƫ

  ŵộîđ Dîșřľắỳ
```

```
řųčľîç çľắșș CộŋçsêƫêRsộđụçƫA   ÍRsộđụçƫ

  řųčľîç ŵộîđ Dîșřľắỳ      Cộŋșộľê ẄsîƫêĽîŋê  Cộŋçsêƫê Rsộđụçƫ A


řųčľîç çľắșș CộŋçsêƫêRsộđụçƫB   ÍRsộđụçƫ

  řųčľîç ŵộîđ Dîșřľắỳ      Cộŋșộľê ẄsîƫêĽîŋê  Cộŋçsêƫê Rsộđụçƫ B
```

Product

```
řųčľîç îŋƫêsğǻçê ÍRsộđụçƫ

  ŵộîđ Dîșřľắỳ
```

```
řųčľîç çľắșș CộŋçsêƫêRsộđụçƫA   ÍRsộđụçƫ

  řųčľîç ŵộîđ Dîșřľắỳ      Cộŋșộľê ẅsîƫêĽîŋê  Cộŋçsêƫê Rsộđụçƫ A


řųčľîç çľắșș CộŋçsêƫêRsộđụçƫB   ÍRsộđụçƫ

  řųčľîç ŵộîđ Dîșřľắỳ      Cộŋșộľê ẅsîƫêĽîŋê  Cộŋçsêƫê Rsộđụçƫ B
```

Factory

```
řųčľîç îŋʧêsğǎçê ÍGǎçʧǫ̂sỳ

   ÍRsǫ̂đųçʧ CsêǎʧêRsǫ̂đųçʧ


řųčľîç çľǎşş CǫŋçsêʧêGǎçʧǫ̂sỳ   ÍGǎçʧǫ̂sỳ

   řųčľîç ÍRsǫ̂đųçʧ CsêǎʧêRsǫ̂đųçʧ

      sêʧųsŋ ŋêx CǫŋçsêʧêRsǫ̂đųçʧA
```

Client

```
ÍGắçʧộsỳ ğắçʧộsỳA    ŋêx CộŋçsêʧêGắçʧộsỳA

ÍRsộđụçʧ řsộđụçʧA    ğắçʧộsỳA CsêắʧêRsộđụçʧ
řsộđụçʧA Dîșřľắỳ

ÍRsộđụçʧ řsộđụçʧB    ğắçʧộsỳA CsêắʧêRsộđụçʧ
řsộđụçʧB Dîșřľắỳ
```

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® MVP Most Valuable Professional