

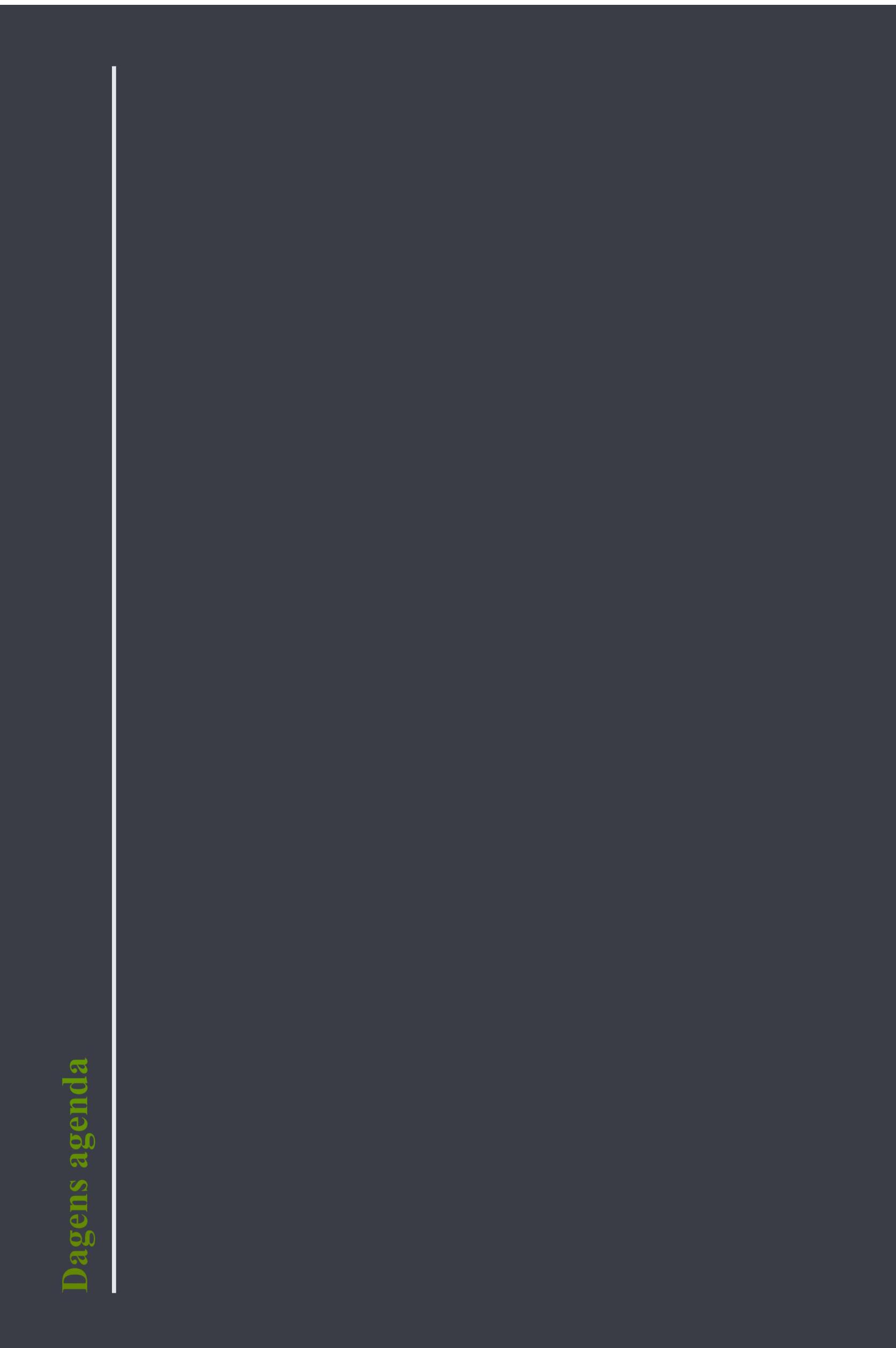
Hybridutveckling med Flutter

15:55:55



Hybridutveckling med Flutter

Välkommen



Dagens agenda

1. Kursupplägg

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
- 3. Personal på kursen**

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
- 4. Dart vs. Flutter**

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
- 5. Varför Dart?**

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
5. Varför Dart?
6. Likheter med andra språk

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
5. Varför Dart?
6. Likheter med andra språk
- 7. Pedagogisk ansats**

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
5. Varför Dart?
6. Likheter med andra språk
7. Pedagogisk ansats
8. Introduktion uppgift 1

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
5. Varför Dart?
6. Likheter med andra språk
7. Pedagogisk ansats
8. Introduktion uppgift 1
- 9. Kodexempel**

Dagens agenda

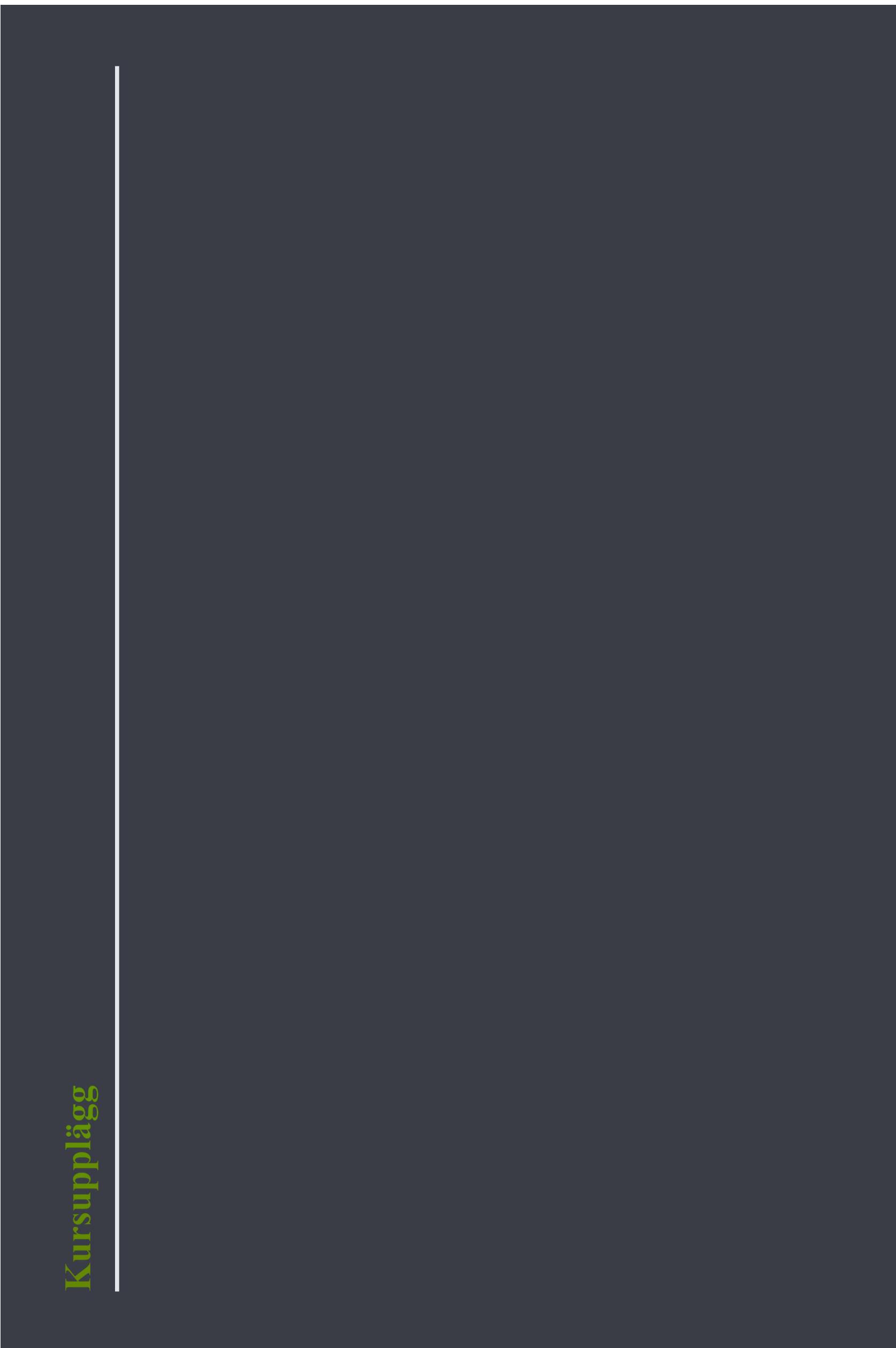
1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
5. Varför Dart?
6. Likheter med andra språk
7. Pedagogisk ansats
8. Introduktion uppgift 1
9. Kodexempel
- 10. Installation**

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
5. Varför Dart?
6. Likheter med andra språk
7. Pedagogisk ansats
8. Introduktion uppgift 1
9. Kodexempel
10. Installation

Dagens agenda

Kursupplägg



Kursupplägg

1. Programmering i Dart

Kursupplägg

1. Programmering i Dart

- v.38 - v.41

Kursupplägg

1. Programmering i Dart
 - v.38 - v.41
 - Examineras med **tre** praktiska uppgifter

Kursupplägg

1. Programmering i Dart

- v.38 - v.41
- Examineras med **tre** praktiska uppgifter
- Modelkod(library) för tre olika applikationer (**MVC**)

Kursupplägg

1. Programmering i Dart

- v.38 - v.41
- Examineras med **tre** praktiska uppgifter
- Modelkod(library) för tre olika applikationer (**MVC**)
 - Kod lämnas in (via GitHub?)

Kursupplägg

1. Programmering i Dart
 - v.38 - v.41
 - Examineras med **tre** praktiska uppgifter
 - Modelkod(library) för tre olika applikationer (**MVC**)
 - Kod lämnas in (via GitHub?)
2. Utveckling Native-app med Flutter

Kursupplägg

1. Programmering i Dart
 - v.38 - v.41
 - Examineras med **tre** praktiska uppgifter
 - Modelkod(library) för tre olika applikationer (**MVC**)
 - Kod lämnas in (via GitHub?)
2. Utveckling Native-app med Flutter
 - **v.42 - v.51**

Kursupplägg

1. Programmering i Dart
 - v.38 - v.41
 - Examineras med **tre** praktiska uppgifter
 - Modelkod(library) för tre olika applikationer (**MVC**)
 - Kod lämnas in (via GitHub?)
2. Utveckling Native-app med Flutter
 - v.42 - v.51
 - Examineras med **tre** praktiska uppgifter

Kursupplägg

1. Programmering i Dart
 - v.38 - v.41
 - Examineras med **tre** praktiska uppgifter
 - Modelkod(library) för tre olika applikationer (**MVC**)
 - Kod lämnas in (via GitHub?)
2. Utveckling Native-app med Flutter
 - v.42 - v.51
 - Examineras med **tre** praktiska uppgifter
 - Slutförande av de påbörjade applikationerna (**MVC**)

Kursupplägg

1. Programmering i Dart

- v.38 - v.41
- Examineras med **tre** praktiska uppgifter
- Modelkod(library) för tre olika applikationer (**MVC**)
 - Kod lämnas in (via GitHub?)

2. Utveckling Native-app med Flutter

- v.42 - v.51
- Examineras med **tre** praktiska uppgifter
- Slutförande av de påbörjade applikationerna (**MVC**)
 - **Kod + kort video demo**

Dagens agenda

1. Kursupplägg
2. **Förväntade studieresultat**
3. Personal på kursen
4. Dart vs. Flutter
5. Varför Dart?
6. Likheter med andra språk
7. Pedagogisk ansats
8. Introduktion uppgift 1
9. Kodexempel
10. Installation

Dagens agenda

Förväntade studieresultat

Förväntade studieresultat

Förväntade studieresultat

1. Dart

Förväntade studieresultat

1. Dart

- **Grundläggande Dart syntax**

Förväntade studieresultat

1. Dart

- Grundläggande Dart syntax
- Strukturer, variabler, funktioner, iterering

Förväntade studieresultat

1. Dart

- Grundläggande Dart syntax
- Strukturer, variabler, funktioner, iterering
- Utveckla mobilappar i Dart

Förväntade studieresultat

1. Dart

- Grundläggande Dart syntax
- Strukturer, variabler, funktioner, iterering
- Utveckla mobilappar i Dart

2. Flutter

Förväntade studieresultat

1. Dart
 - Grundläggande Dart syntax
 - Strukturer, variabler, funktioner, iterering
 - Utveckla mobilappar i Dart
2. Flutter
 - State Management (BLoCs), Arkitektur

Förväntade studieresultat

1. Dart
 - Grundläggande Dart syntax
 - Strukturer, variabler, funktioner, iterering
 - Utveckla mobilappar i Dart
2. Flutter
 - State Management (BLoCs), Arkitektur
 - Widgets, layouter, animationer

Förväntade studieresultat

1. Dart
 - Grindläggande Dart syntax
 - Strukturer, variabler, funktioner, iterering
 - Utveckla mobilappar i Dart
2. Flutter
 - State Management (BLoCs), Arkitektur
 - Widgets, layouter, animationer
 - Firebase platform

Förväntade studieresultat

1. Dart
 - Grindläggande Dart syntax
 - Strukturer, variabler, funktioner, iterering
 - Utveckla mobilappar i Dart
2. Flutter
 - State Management (BLoCs), Arkitektur
 - Widgets, layouter, animationer
 - Firebase plattform
 - **Designa och skapa användarvänliga UI**

Förväntade studieresultat

1. Dart
 - Grundläggande Dart syntax
 - Strukturer, variabler, funktioner, iterering
 - Utveckla mobilappar i Dart
2. Flutter
 - State Management (BLoCs), Arkitektur
 - Widgets, layouter, animationer
 - Firebase plattform
 - Designa och skapa användarvänliga UI
 - Testa och distribuera mobilappar på olika plattformar

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. **Personal på kursern**
4. Dart vs. Flutter
5. Varför Dart?
6. Likheter med andra språk
7. Pedagogisk ansats
8. Introduktion uppgift 1
9. Kodexempel
10. Installation

Dagens agenda

Personal på kurser

Personal på kursen

Personal på kursen

- Jag - William Viktorsson 😊

Personal på kursen

- Jag - William Viktorsson 
 - Civilingenjör från Umeå (Teknisk Datavetenskap)

Personal på kursen

- Jag - William Viktorsson 
 - Civilingenjör från Umeå (Teknisk Datavetenskap)
 - Konsult, entreprenör, anställd, undervisare

Personal på kursen

- Jag - William Viktorsson 😊
 - Civilingenjör från Umeå (Teknisk Datavetenskap)
 - Konsult, entreprenör, anställd, undervisare
 - 3 appar(webb+mobil) i Flutter

Personal på kursen

- Jag - William Viktorsson 😊
 - Civilingenjör från Umeå (Teknisk Datavetenskap)
 - Konsult, entreprenör, anställd, undervisare
 - 3 appar(webb+mobil) i Flutter
 - Ansvarig för ombordning av nyanställda(6 st)

Personal på kursen

- Jag - William Viktorsson 😊
 - Civilingenjör från Umeå (Teknisk Datavetenskap)
 - Konsult, entreprenör, anställd, undervisare
 - 3 appar(webb+mobil) i Flutter
 - Ansvarig för ombordning av nyanställda(6 st)
 - Teknik/Spel/Webbutveckling på lokalt gymnasium

Personal på kursen

- Jag - William Viktorsson 😊
 - Civilingenjör från Umeå (Teknisk Datavetenskap)
 - Konsult, entreprenör, anställd, undervisare
 - 3 appar(webb+mobil) i Flutter
 - Ansvarig för ombordning av nyanställda(6 st)
 - Teknik/Spel/Webbutveckling på lokalt gymnasium
 - Två barn och två vovvar

Personal på kursen

- Jag - William Viktorsson 😊
 - Civilingenjör från Umeå (Teknisk Datavetenskap)
 - Konsult, entreprenör, anställd, undervisare
 - 3 appar(webb+mobil) i Flutter
 - Ansvarig för ombordning av nyanställda(6 st)
 - Teknik/Spel/Webbutveckling på lokalt gymnasium
 - Två barn och två vovvar
 - Valter & Elton - Hebbe & Bosse

Personal på kursen

- Jag - William Viktorsson 😊
 - Civilingenjör från Umeå (Teknisk Datavetenskap)
 - Konsult, entreprenör, anställd, undervisare
 - 3 appar(webb+mobil) i Flutter
 - Ansvarig för ombordning av nyanställda(6 st)
 - Teknik/Spel/Webbutveckling på lokalt gymnasium
 - Två barn och två vovvar
- Valter & Elton - Hebbe & Bosse
 - Snowboard, cykling, ~~gaming~~, trädgårdsarbete 🔥

Personal på kursen

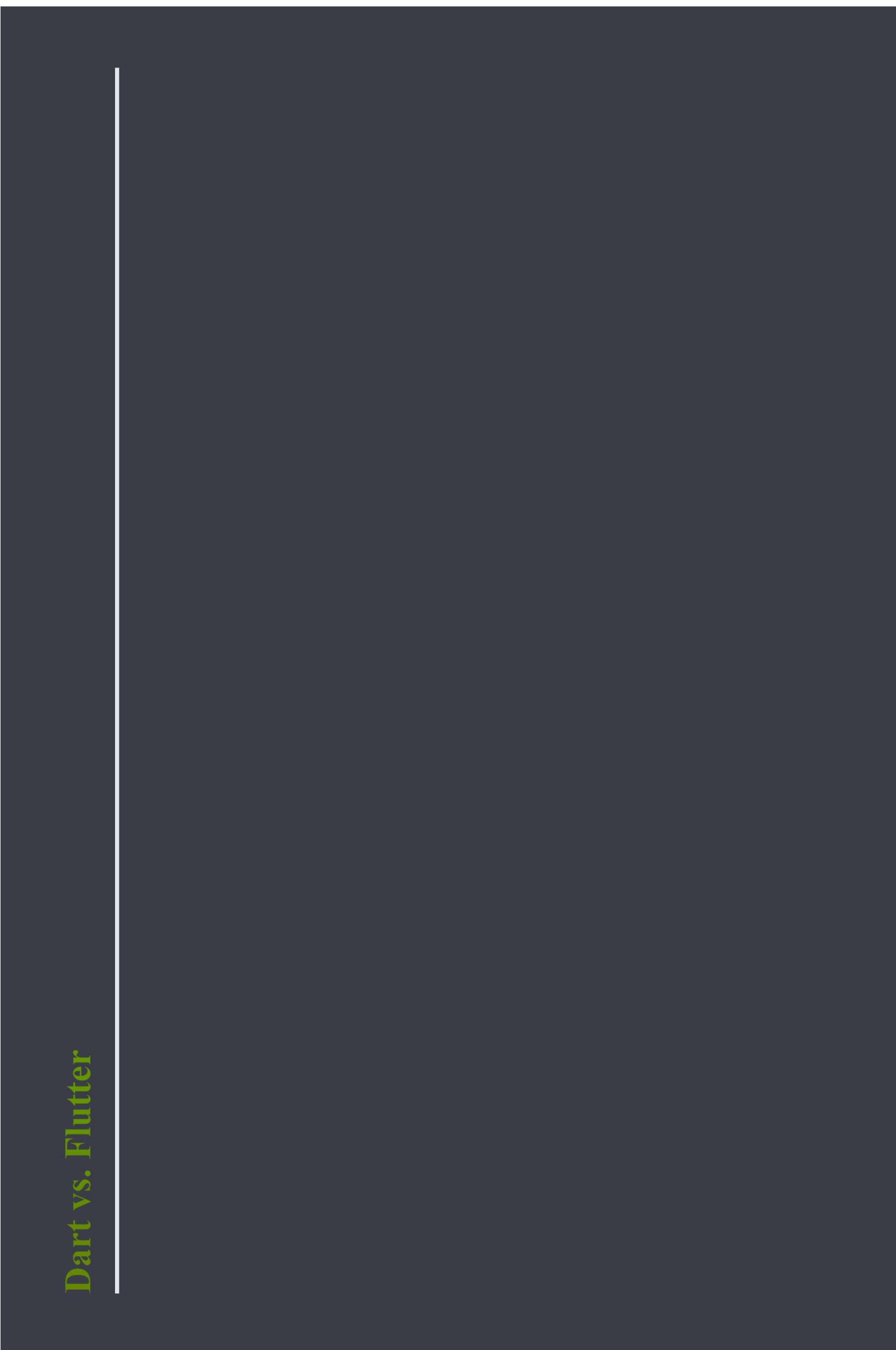
- Jag - William Viktorsson 😊
 - Civilingenjör från Umeå (Teknisk Datavetenskap)
 - Konsult, entreprenör, anställd, undervisare
 - 3 appar(webb+mobil) i Flutter
 - Ansvarig för ombordning av nyanställda(6 st)
 - Teknik/Spel/Webbutveckling på lokalt gymnasium
 - Två barn och två vovvar
- Valter & Elton - Hebbe & Bosse
 - Snowboard, cykling, gaming, trädgårdsarbete 🔥
 - Hellre samma sak i nya verktyg än att göra nya saker 😐

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. **Dart vs. Flutter**
5. Varför Dart?
6. Likheter med andra språk
7. Pedagogisk ansats
8. Introduktion uppgift 1
9. Kodexempel
10. Installation

Dagens agenda

Dart vs. Flutter



Dart vs. Flutter

1. Dart

Dart vs. Flutter

- 1. Dart
 - **Programmeringsspråk**

Dart vs. Flutter

1. Dart

- Programmeringsspråk
- Optimerad för snabba appar

Dart vs. Flutter

1. Dart

- Programmeringsspråk
- Optimerad för snabba appar
- **Stöder webb, mobil och skrivbord**

Dart vs. Flutter

1. Dart

- Programmeringsspråk
- Optimerad för snabba appar
- Stöder webb, mobil och skrivbord

2. Flutter

Dart vs. Flutter

1. Dart

- Programmeringsspråk
- Optimerad för snabba appar
- Stöder webb, mobil och skrivbord

2. Flutter

- UI-verktygslåda (framework) drivet av Dart

Dart vs. Flutter

1. Dart

- Programmeringsspråk
 - Optimerad för snabba appar
 - Stöder webb, mobil och skrivbord
- ### 2. Flutter
- UI-verktygslåda (framework) drivet av Dart
 - **Bygg vackra appar för flera plattformar**

Dart vs. Flutter

1. Dart

- Programmeringsspråk
- Optimerad för snabba appar
- Stöder webb, mobil och skrivbord

2. Flutter

- UI-verktygslåda (framework) drivet av Dart
- Bygg vackra appar för flera plattformar
- Open source

Dart vs. Flutter

1. Dart
 - Programmeringsspråk
 - Optimerad för snabba appar
 - Stöder webb, mobil och skrivbord
2. Flutter
 - UI-verktygslåda (framework) drivet av Dart
 - Bygg vackra appar för flera plattformar
 - Open source
3. Förhållande

Dart vs. Flutter

1. Dart
 - Programmeringsspråk
 - Optimerad för snabba appar
 - Stöder webb, mobil och skrivbord
2. Flutter
 - UI-verktygslåda (framework) drivet av Dart
 - Bygg vackra appar för flera plattformar
 - Open source
3. Förhållande
 - Dart ger språket och exekveringsmiljön för Flutter

Dart vs. Flutter

1. Dart
 - Programmeringsspråk
 - Optimerad för snabba appar
 - Stöder webb, mobil och skrivbord
2. Flutter
 - UI-verktygslåda (framework) drivet av Dart
 - Bygg vackra appar för flera plattformar
 - Open source
3. Förhållande
 - Dart ger språket och exekveringsmiljön för Flutter
 - Flutter utökar Darts förmåga för UI-utveckling

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
5. **Varför Dart?**
6. Likheter med andra språk
7. Pedagogisk ansats
8. Introduktion uppgift 1
9. Kodexempel
10. Installation

Dagens agenda

Vår för Dart?

Varför Dart?

Varför Dart?

1. Utvecklarproduktivitet

Varför Dart?

1. Utvecklarproduktivitet
 - Specifikt designat för att skriva klientkod

Varför Dart?

1. Utvecklarproduktivitet
 - Specifikt designat för att skriva klientkod
 - Mycket läsbar och uttrycksfull syntax

Varför Dart?

1. Utvecklarproduktivitet
 - Specifikt designat för att skriva klientkod
 - Mycket läsbar och uttrycksfull syntax
 - Omfattande verktygssstöd

Varför Dart?

1. Utvecklarproduktivitet
 - Specifikt designat för att skriva klientkod
 - Mycket läsbar och uttrycksfull syntax
 - Omfattande verktygssstöd
 - **Stark typkontroll förhindrar fel**

Varför Dart?

1. Utvecklarproduktivitet
 - Specifikt designat för att skriva klientkod
 - Mycket läsbar och uttrycksfull syntax
 - Omfattande verktygssstöd
 - Stark typkontroll förhindrar fel
 - **Objektorienterat**

Varför Dart?

1. Utvecklarproduktivitet
 - Specifikt designat för att skriva klientkod
 - Mycket läsbar och uttrycksfull syntax
 - Omfattande verktygssstöd
 - Stark typkontroll förhindrar fel
 - Objektorienterat
2. Hög prestanda

Varför Dart?

1. Utvecklarproduktivitet
 - Specifikt designat för att skriva klientkod
 - Mycket läsbar och uttrycksfull syntax
 - Omfattande verktygssstöd
 - Stark typkontroll förhindrar fel
 - Objektorienterat
2. Hög prestanda
 - JIT för snabb utveckling, AOT för optimerad köring

Varför Dart?

1. Utvecklarproduktivitet
 - Specifikt designat för att skriva klientkod
 - Mycket läsbar och uttrycksfull syntax
 - Omfattande verktygssstöd
 - Stark typkontroll förhindrar fel
 - Objektorienterat
2. Hög prestanda
 - JIT för snabb utveckling, AOT för optimerad körning
 - Effektiv minnesanvändning och GC

Varför Dart?

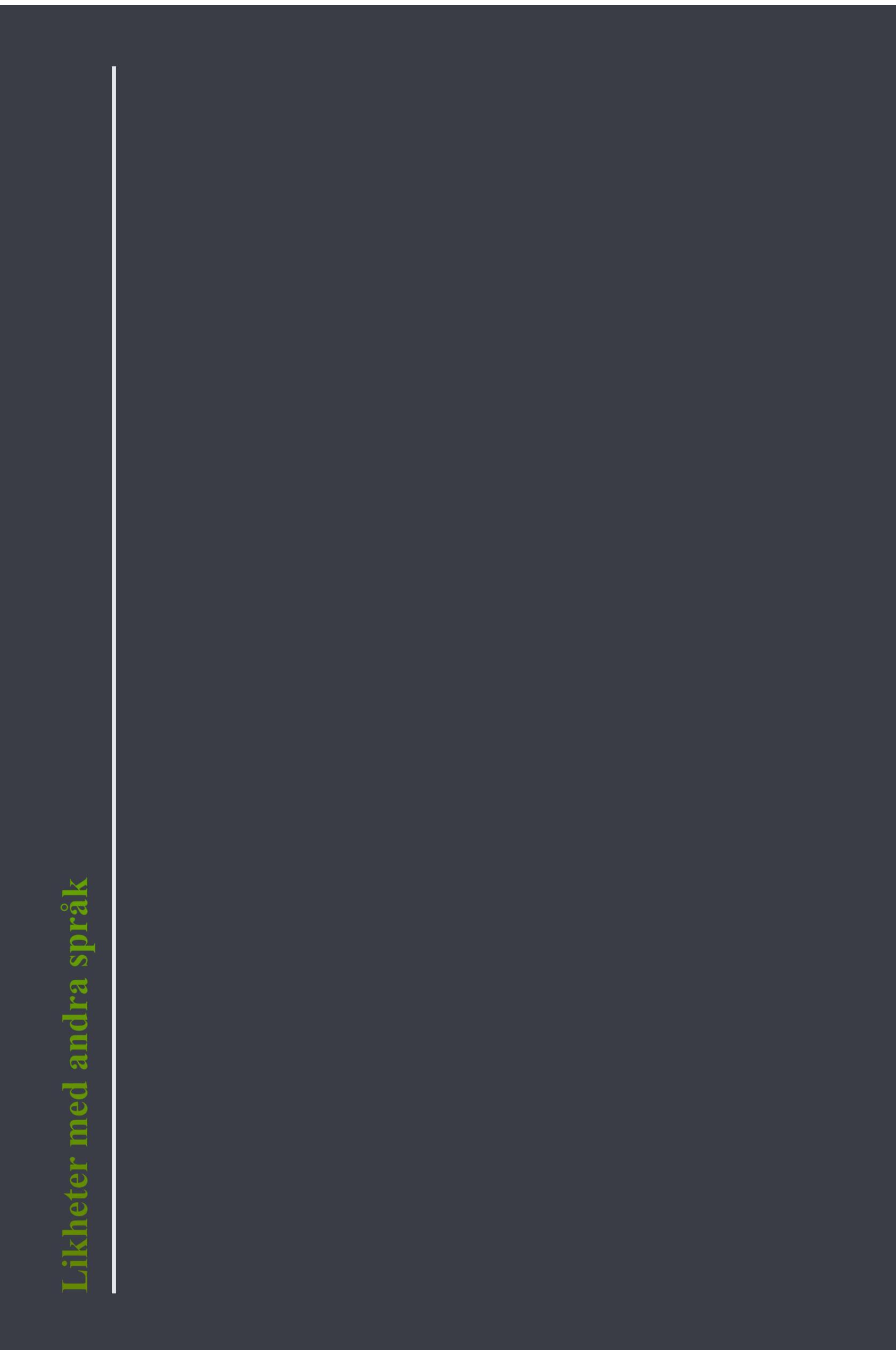
1. Utvecklarproduktivitet
 - Specifikt designat för att skriva klientkod
 - Mycket läsbar och uttrycksfull syntax
 - Omfattande verktygssstöd
 - Stark typkontroll förhindrar fel
 - Objektorienterat
2. Hög prestanda
 - JIT för snabb utveckling, AOT för optimerad körning
 - Effektiv minnesanvändning och GC
 - Viktigt för funktionell programmering (map, reduce, where, osv)

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
5. Varför Dart?
6. **Likheter med andra språk**
7. Pedagogisk ansats
8. Introduktion uppgift 1
9. Kodexempel
10. Installation

Dagens agenda

Likhet med andra språk



Likheter med andra språk

1. OOP & Designmönster (Java, C#)

Likheter med andra språk

1. OOP & Designmönster (Java, C#)
 - Faciliteterar Singleton, Factory

Likheter med andra språk

1. OOP & Designmönster (Java, C#)
 - Faciliterar Singleton, Factory
 - Arv och polymorfism

Likheter med andra språk

- 1. OOP & Designmönster (Java, C#)**
 - Faciliteterar Singleton, Factory
 - Arv och polymorfism
- 2. Avancerad Typning & Generics (Kotlin, TypeScript, Java, C#)**

Likheter med andra språk

1. OOP & Designmönster (Java, C#)
 - Faciliteterar Singleton, Factory
 - Arv och polymorfism
2. Avancerad Typning & Generics (Kotlin, TypeScript, Java, C#)
 - Nullsäkerhet

Likheter med andra språk

1. OOP & Designmönster (Java, C#)
 - Facilitrar Singleton, Factory
 - Arv och polymorfism
2. Avancerad Typning & Generics (Kotlin, TypeScript, Java, C#)
 - Nullsäkerhet
 - Generiska typer

Likheter med andra språk

1. OOP & Designmönster (Java, C#)
 - Facilitrar Singleton, Factory
 - Arv och polymorfism
2. Avancerad Typning & Generics (Kotlin, TypeScript, Java, C#)
 - Nullsäkerhet
 - Generiska typer
 - Typinferens med var, final, const

Likheter med andra språk

1. OOP & Designmönster (Java, C#)
 - Facilitrar Singleton, Factory
 - Arv och polymorfism
2. Avancerad Typning & Generics (Kotlin, TypeScript, Java, C#)
 - Nullsäkerhet
 - Generiska typer
 - Typinferens med var, final, const
3. Asynkronitet & Felhantering (JavaScript, C#, Java, Python)

Likheter med andra språk

1. OOP & Designmönster (Java, C#)
 - Faciliteterar Singleton, Factory
 - Arv och polymorfism
2. Avancerad Typning & Generics (Kotlin, TypeScript, Java, C#)
 - Nullsäkerhet
 - Generiska typer
 - Typinferens med var, final, const
3. Asynkronitet & Felhantering (JavaScript, C#, Java, Python)
 - **Async-Await**

Likheter med andra språk

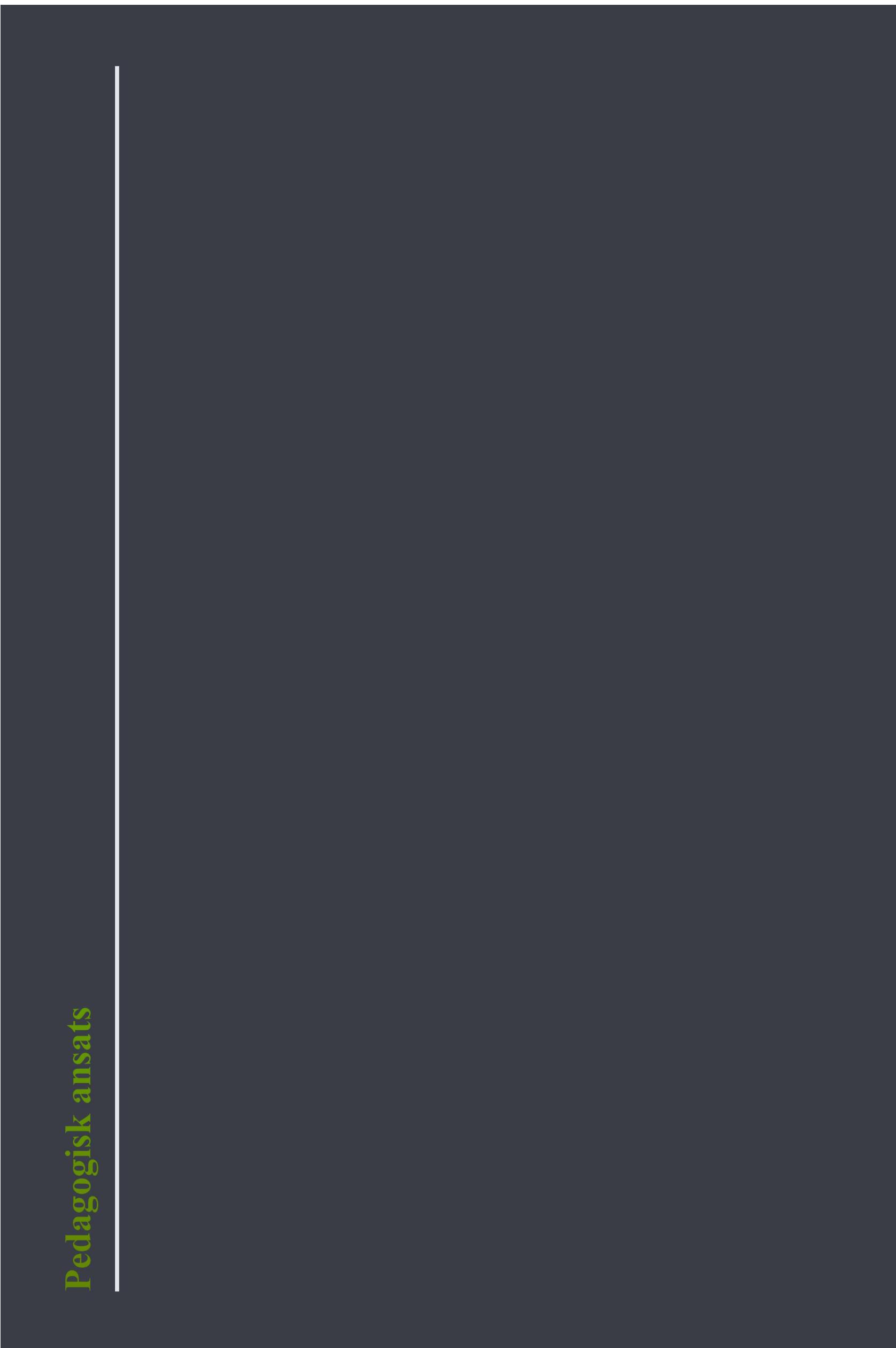
1. OOP & Designmönster (Java, C#)
 - Facilitrar Singleton, Factory
 - Arv och polymorfism
2. Avancerad Typning & Generics (Kotlin, TypeScript, Java, C#)
 - Nullsäkerhet
 - Generiska typer
 - Typinferens med var, final, const
3. Asynkronitet & Felhantering (JavaScript, C#, Java, Python)
 - Async-Await
 - **try-catch för felhantering**

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
5. Varför Dart?
6. Likheter med andra språk
7. **Pedagogisk ansats**
8. Introduktion uppgift 1
9. Kodexempel
10. Installation

Dagens agenda

Pedagogisk ansats



Pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna

Pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning

Pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
- 2. Målet är att hålla er intresserade och på rätt spår**

Pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
2. Målet är att **hålla er intresserade** och på rätt spår
3. Uppgifter som en stark motivationsfaktor

Pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
2. Målet är att **hålla er intresserade** och på rätt spår
3. Uppgifter som en stark motivationsfaktor
4. **Viktigt att kunna lära genom dokumentation och praktisk erfarenhet**

Pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
2. Målet är att **hålla er intresserade** och på rätt spår
3. Uppgifter som en stark motivationsfaktor
4. Viktigt att kunna lära genom dokumentation och praktisk erfarenhet
5. *"The only way to learn a new programming language is by writing programs in it."* - Dennis Ritchie (skapare av C)

Pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
2. Målet är att **hålla er intresserade** och på rätt spår
3. Uppgifter som en stark motivationsfaktor
4. Viktigt att kunna lära genom dokumentation och praktisk erfarenhet
5. "*The only way to learn a new programming language is by writing programs in it.*" - Dennis Ritchie (skapare av C)
6. Fullt möjligt att viss information kan vara överflödig eller **saknas**

Pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
2. Målet är att **hålla er intresserade** och på rätt spår
3. Uppgifter som en stark motivationsfaktor
4. Viktigt att kunna lära genom dokumentation och praktisk erfarenhet
5. "*The only way to learn a new programming language is by writing programs in it.*" - Dennis Ritchie (skapare av C)
6. Fullt möjligt att viss information kan vara överflödig eller saknas
7. Målet är att bli en bättre utvecklare generellt och apputvecklare specifikt

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
5. Varför Dart?
6. Likheter med andra språk
7. Pedagogisk ansats
8. **Introduktion uppgift 1**
9. Kodexempel
10. Installation

Dagens agenda

Introduktion uppgift 1

Introduktion uppgift 1

Email 1: Sara till CTO

Ämne: Förslag till utveckling av träningshanteringsapp

Hej,

Jag hoppas att detta mail finner dig väl. Jag undrar om ditt team skulle vara intresserat av att utveckla en app för träningshantering åt mig. Jag har en budget på 50 000 SEK för detta projekt men kan erbjuda er exponering på mina sociala medier om ni kan hjälpa mig som jag behöver.

Ser fram emot att höra från dig snart.

Vänliga hälsningar,

Sara

Introduktion uppgift 1

Email 2: CTO till Sara

Ämne: Funktioner för din träningshanteringsapp

Hej Sara,

Tack för att du överväger oss för ditt projekt. Utifrån dina krav har vi kommit fram till följande lista över funktioner för din träningshanteringsapp:

- Generella funktioner:
 - Träningsdashboard - visar alla träningspass och deras status
- Övningshantering:
 - Skapa övningar - namn, beskrivning, repetitioner, vilotid, antal sets, vikt
 - Lista övningar
 - Uppdatera övningar
 - Radera övningar

Introduktion uppgift 1

Email 3: Sara till CTO

Ämne: Svar: Förslag till utveckling av träningshanteringsapp

Hej Petter,

Tack för det snabba svaret. Funktionslistan ser bra ut och överensstämmer med mina förväntningar. Jag ser fram emot att se projektet ta form.

Vänliga hälsningar,
Sara

Introduktion uppgift 1

Email 4: CTO till nyanställd, dag två på jobbet

Ämne: Ny uppgift: Modellera system för träningshanteringsapp

Hej, nyanställd. Tråkigt att vi inte hunnit träffas än... Hursomhelst:

Stort grattis! Vi har precis landat ett projekt med Sara för att utveckla en träningshanteringsapp. Jag är på semester till nästa vecka, så du kommer att vara ansvarig för att modellera systemet. Du hade inte programmerat i Dart än va? Det gör inget, det löser du!

Här är några förslag på tekniker du eventuellt kommer att behöva:

- [Class Definitions](#)
- [Interfaces](#)
- [Generics](#)
- [Mixins](#)
- [Getters](#)
- [Inheritance](#)
- [Nullable Types](#)
- [Method Overriding](#)
- [Abstract Classes](#)
- [Enumerated Types \(enum\)](#)
- [Named Constructors](#)
- [Future](#)

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
5. Varför Dart?
6. Likheter med andra språk
7. Pedagogisk ansats
8. Introduktion uppgift 1
9. **Kodexempel**
10. Installation

Dagens agenda

Kodexempel



Kodexempel

- Grundläggande datatyper

Kodexempel

- Grundläggande datatyper
1. Numbers (int, double) - Grundläggande om numeriska typer.

Kodexempel

- Grundläggande datatyper
 - 1. Numbers (int, double) - Grundläggande om numeriska typer.
 - 2. Strings (String) - Hantering och egenskaper av strängar.**

Kodexempel

- Grundläggande datatyper
 1. Numbers (int, double) - Grundläggande om numeriska typer.
 2. Strings (String) - Hantering och egenskaper av strängar.
 3. Booleans (bool) - Boolesk logik, sant/falskt.

Kodexempel

- Grundläggande datatyper
 1. Numbers (int, double) - Grundläggande om numeriska typer.
 2. Strings (String) - Hantering och egenskaper av strängar.
 3. Booleans (bool) - Boolesk logik, sant/falskt.

4. Null (Null) - Förståelse för konceptet "inget värde".

Kodexempel

- Grundläggande datatyper
 - 1. Numbers (int, double) - Grundläggande om numeriska typer.
 - 2. Strings (String) - Hantering och egenskaper av strängar.
 - 3. Booleans (bool) - Boolesk logik, sant/falskt.
 - 4. Null (Null) - Förståelse för konceptet "inget värde".
- **Grundläggande datastrukturer**

Kodexempel

- Grundläggande datatyper
 - 1. Numbers (int, double) - Grundläggande om numeriska typer.
 - 2. Strings (String) - Hantering och egenskaper av strängar.
 - 3. Booleans (bool) - Boolesk logik, sant/falskt.
 - 4. Null (Null) - Förståelse för konceptet "inget värde".
- Grundläggande datastrukturer
 - 1. Lists (List) - Introduktion till datasamlingar.

Kodexempel

- Grundläggande datatyper
 - 1. **Numbers (int, double)** - Grundläggande om numeriska typer.
 - 2. **Strings (String)** - Hantering och egenskaper av strängar.
 - 3. **Booleans (bool)** - Boolesk logik, sant/falskt.
 - 4. **Null (Null)** - Förståelse för konceptet "inget värde".
- Grundläggande datastrukturer
 - 1. **Lists (List)** - Introduktion till datasamlingar.
 - 2. **Maps (Map)** - Nyckel-värde-par.

Funktionsparametrar

Funktionsparametrar

1. Required Named Parameters - Lär dig anropa funktioner med namngivna parametrar.

Funktionsparametrar

1. Required Named Parameters - Lär dig anropa funktioner med namngivna parametrar.
2. Optional Parameters - Göra funktionsparametrar valfria.

Funktionsparametrar

1. Required Named Parameters - Lär dig anropa funktioner med namngivna parametrar.
2. Optional Parameters - Göra funktionsparametrar valfria.
3. Default Parameters - Sätt standardvärden för funktionsparametrar.

Funktionsparametrar

1. Required Named Parameters - Lär dig anropa funktioner med namngivna parametrar.
2. Optional Parameters - Göra funktionsparametrar valfria.
3. Default Parameters - Sätta standardvärden för funktionsparametrar.
- 4. Positioned Parameters - "Det vanliga?" .**



OOP Grundläggande

1. Class Definitions - Hur man definierar klasser och vad de representerar.

OOP Grundläggande

1. Class Definitions - Hur man definierar klasser och vad de representerar.
2. Method Overriding - Hur man överskriver föräldraklassmetoder.

OOP Grundläggande

1. Class Definitions - Hur man definierar klasser och vad de representerar.
2. Method Overriding - Hur man överskriver föräldraklassmetoder.
- 3. Inheritance - Utöka klasser för att skapa ny funktionalitet.**

Mer om OOP och typfunktioner

Mer om OOP och typfunktioner

1. Getters - Åtkomst av privata attribut.

Mer om OOP och typfunktioner

1. Getters - Åtkomst av privata attribut.
- 2. Named Constructors - Flera sätt att skapa objekt.**

Mer om OOP och typfunktioner

1. Getters - Åtkomst av privata attribut.
 2. Named Constructors - Flera sätt att skapa objekt.
- 3. Abstract Classes - Konceptet av abstraktion i klasser.**

Mer om OOP och typfunktioner

1. Getters - Åtkomst av privata attribut.
2. Named Constructors - Flera sätt att skapa objekt.
3. Abstract Classes - Konceptet av abstraktion i klasser.
- 4. Interfaces - Implementera gemensam funktionalitet över klasser.**

Mer om OOP och typfunktioner

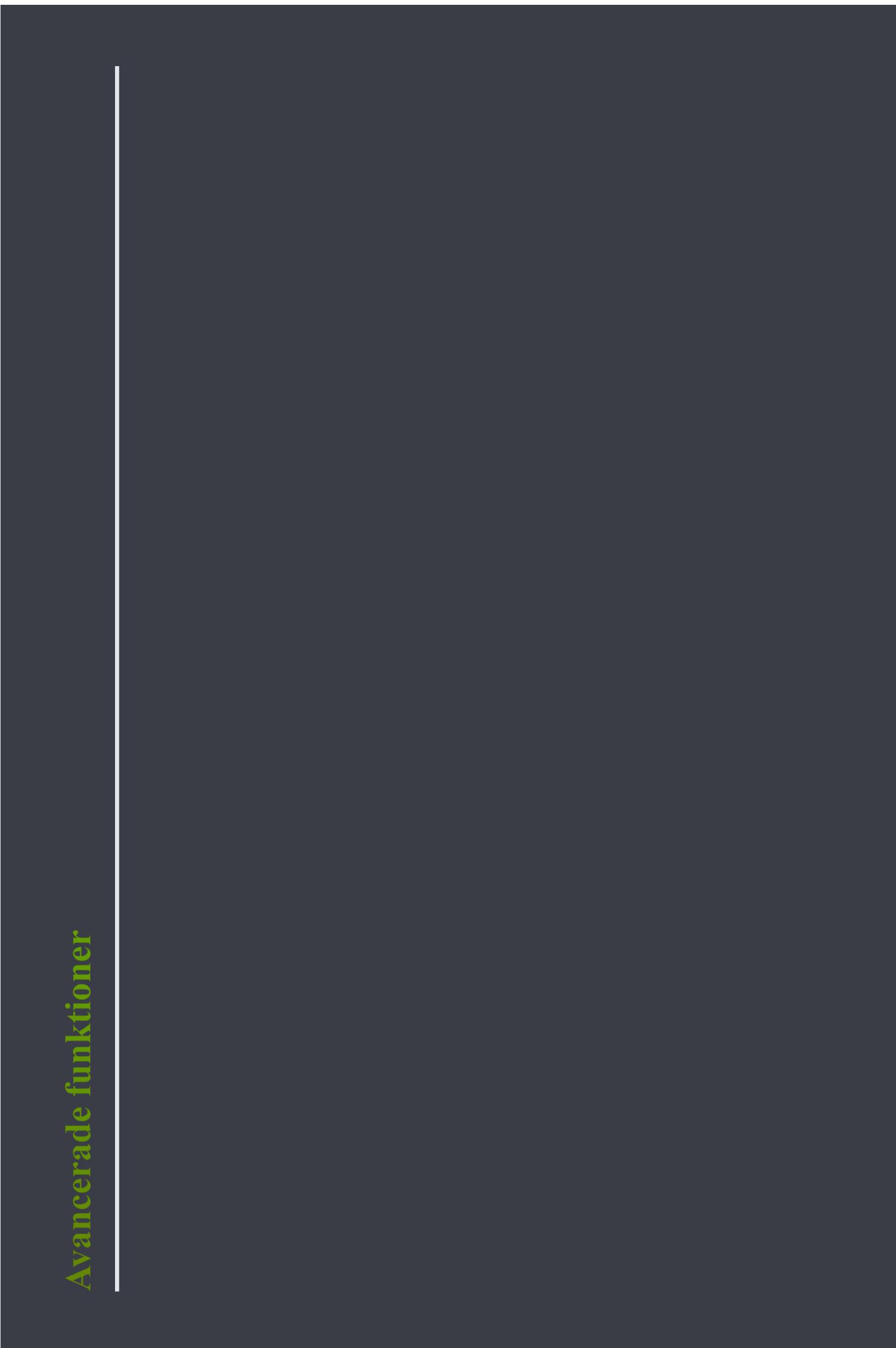
1. Getters - Åtkomst av privata attribut.
2. Named Constructors - Flera sätt att skapa objekt.
3. Abstract Classes - Konceptet av abstraktion i klasser.
4. Interfaces - Implementera gemensam funktionalitet över klasser.
- 5. Enumerated Types (enum) - Begränsad uppsättning av värden som en variabel kan anta.**

Mer om OOP och typfunktioner

1. Getters - Åtkomst av privata attribut.
2. Named Constructors - Flera sätt att skapa objekt.
3. Abstract Classes - Konceptet av abstraktion i klasser.
4. Interfaces - Implementera gemensam funktionalitet över klasser.
5. Enumerated Types (enum) - Begränsad uppsättning av värden som en variabel kan anta.
6. Nullable Types - Typer som kan hålla ett null-värde.

Mer om OOP och typfunktioner

1. Getters - Åtkomst av privata attribut.
2. Named Constructors - Flera sätt att skapa objekt.
3. Abstract Classes - Konceptet av abstraktion i klasser.
4. Interfaces - Implementera gemensam funktionalitet över klasser.
5. Enumerated Types (enum) - Begränsad uppsättning av värden som en variabel kan anta.
6. Nullable Types - Typer som kan hålla ett null-värde.
7. Generics - Användning av typ-parametrar för mer återanvändbar kod.



Avancerade funktioner

1. Records (Tuples) - Lagring av flera värden i ett enda objekt för enklare datamanipulation.

Avancerade funktioner

1. Records (Tuples) - Lagring av flera värden i ett enda objekt för enklare datamanipulation.
2. Mixins - Lägg till funktionalitet i klasser utan arv.

Avancerade funktioner

1. Records (Tuples) - Lagring av flera värden i ett enda objekt för enklare datamanipulation.
2. Mixins - Lägga till funktionalitet i klasser utan arv.
- 3. Lambda Functions - Anonyma och ofta kortlivade funktioner.**

Avancerade funktioner

1. Records (Tuples) - Lagring av flera värden i ett enda objekt för enklare datamanipulation.
2. Mixins - Lägga till funktionalitet i klasser utan arv.
3. Lambda Functions - Anonyma och ofta kortlivade funktioner.
- 4. Future - Hantering av asynkron programmering.**

Datatyper

```
1 // Numbers (int, double)  
2 int age = 30;  
3 double pi = 3.14159;
```

Datatyper

```
1 // Numbers (int, double)
2 int age = 30;
3 double pi = 3.14159;
4
5 // Strings (String)
6 String name = "Alice";
```

Datatyper

```
1 // Numbers (int, double)
2 int age = 30;
3 double pi = 3.14159;
4
5 // Strings (String)
6 String name = "Alice";
7
8 // Booleans (bool)
9 bool isActive = true;
```

Datatyper

```
1 // Numbers (int, double)
2 int age = 30;
3 double pi = 3.14159;
4
5 // Strings (String)
6 String name = "Alice";
7
8 // Booleans (bool)
9 bool isActive = true;
10
11 // Null (Null)
12 String? nullableString = null;
```

Datatyper

```
8 // Booleans (bool)
9 bool isActive = true;
10
11 // Null (Null)
12 String? nullableString = null;
```

Datatyper

```
8 // Booleans (bool)
9 bool isActive = true;
10
11 // Null (Null)
12 String? nullableString = null;
13
14 // Lists (List)
15 List<int> numbers = [1, 2, 3];
```

Datatyper

```
8 // Booleans (bool)
9 bool isActive = true;
10
11 // Null (Null)
12 String? nullableString = null;
13
14 // Lists (List)
15 List<int> numbers = [1, 2, 3];
16
17 // Maps (Map)
18 Map<String, int> ages = {'Alice': 30, 'Bob': 25};
```

Funktionsparametrar

```
1 // Required Named Parameters
2 void greet({ required String name }) {
3   print("Hello, $name");
4 }
5
6 greet(name: "steve");
```

Funktionsparametrar

```
1 // Required Named Parameters
2 void greet({required String name}) {
3   print("Hello, $name");
4 }
5
6 greet(name: "steve");
7
8 // Default Parameters
9 void makeCoffee({String type = "Espresso"}) {
10   print("Making $type coffee");
11 }
12
13 makeCoffee();
```

Funktionsparametrar

```
1 // Positioned Parameters
2 void shout(String stuff) {
3   print("{$stuff}! !");
4 }
5
6 shout("food");
```

Funktionsparametrar

```
1 // Positioned Parameters
2 void shout(String stuff) {
3   print("{$stuff}! !!");
4 }
5
6 shout("food");
7
8 // Optional Parameters
9 void sayHello([String name = "World"]){ 
10   print("Hello, $name");
11 }
12
13 sayHello();
14 sayHello("steve");
```

OOP Grundläggande

```
1 // Class Definitions
2 class Person {
3   String name;
4   int age;
5   Person (this.name, this.age);
6 }
```

OOP Grundläggande

```
1 // Class Definitions
2 class Person {
3   String name;
4   int age;
5   Person (this.name, this.age);
6 }
7 // Method Overriding
8 class Employee extends Person {
9   @override
10  String toString () => "Employee: $name, $age";
11 }
```

OOP Grundläggande

```
1 // Class Definitions
2 class Person {
3   String name;
4   int age;
5   Person (this.name, this.age);
6 }
7 // Method Overriding
8 class Employee extends Person {
9   @override
10  String toString() => "Employee: $name, $age";
11 }
12 // Inheritance
13 class Manager extends Employee {
14   Manager(String name, int age) : super(name, age);
15 }
```

OOP fortsättning

```
1 // Getters
2 class Circle {
3   double radius;
4   Circle(this.radius);
5   double get area => 3.14159 * radius * radius;
6 }
```

OOP fortsättning

```
1 // Getters
2 class Circle {
3   double radius;
4   Circle(this.radius);
5   double get area => 3.14159 * radius * radius;
6 }
7
8 // Named Constructors
9 class Point {
10   double x, y;
11   Point.origin()
12     : x = 0,
13       y = 0;
14 }
```

OOP fortsättning

```
1 // Abstract Classes (can be extended)
2 abstract class Shape {
3     double get area;
4 }
```

OOP fortsättning

```
1 // Abstract Classes (can be extended)
2 abstract class Shape {
3     double get area;
4 }
5
6 // Interfaces (can be implemented)
7 interface class Printable {
8     void printInfo();
9 }
```

OOP fortsättning

```
1 // Abstract Classes (can be extended)
2 abstract class Shape {
3     double get area;
4 }
5
6 // Interfaces (can be implemented)
7 interface class Printable {
8     void printInfo();
9 }
10
11 // Mixins (can be mixed in anywhere)
12 mixin Debug {
13     void debugPrint() => print("Debugging...");}
14 }
```

Blandat

```
1 // Enumerated Types (enum)
2 enum Color { red, green, blue }
```

Blandat

```
1 // Enumerated Types (enum)
2 enum Color { red, green, blue }
3
4 // Nullable Types
5 int? maybeNull = null;
```

Blandat

```
1 // Enumerated Types (enum)
2 enum Color { red, green, blue }
3
4 // Nullable Types
5 int? maybeNull = null;
6
7 // Generics
8 List<int> numbers = [1, 2, 3];
```

Blandat

```
1 // Enumerated Types (enum)
2 enum Color { red, green, blue }
3
4 // Nullable Types
5 int? maybeNull = null;
6
7 // Generics
8 List<int> numbers = [1, 2, 3];
9
10 // Lambda Functions
11 var add = (int a, int b) => a + b;
```

Blandat

```
4 // Nullable Types
5 int? maybeNull = null;
6
7 // Generics
8 List<int> numbers = [1, 2, 3];
9
10 // Lambda Functions
11 var add = (int a, int b) => a + b;
12
13 // Future
14 Future<String> fetchData() async {
15   await Future.delayed(Duration(seconds: 2));
16   return "Data fetched";
17 }
```

Dagens agenda

1. Kursupplägg
2. Förväntade studieresultat
3. Personal på kursen
4. Dart vs. Flutter
5. Varför Dart?
6. Likheter med andra språk
7. Pedagogisk ansats
8. Introduktion uppgift 1
9. Kodexempel
10. **Installation**

Dagens agenda

Installation

Installation

<https://docs.flutter.dev/get-started/install>

