DHBW Stuttgart
Duale Hochschule
Baden-Württemberg

Software Engineering: Vorgehensmodelle

Dr. Eugenie Giesbrecht

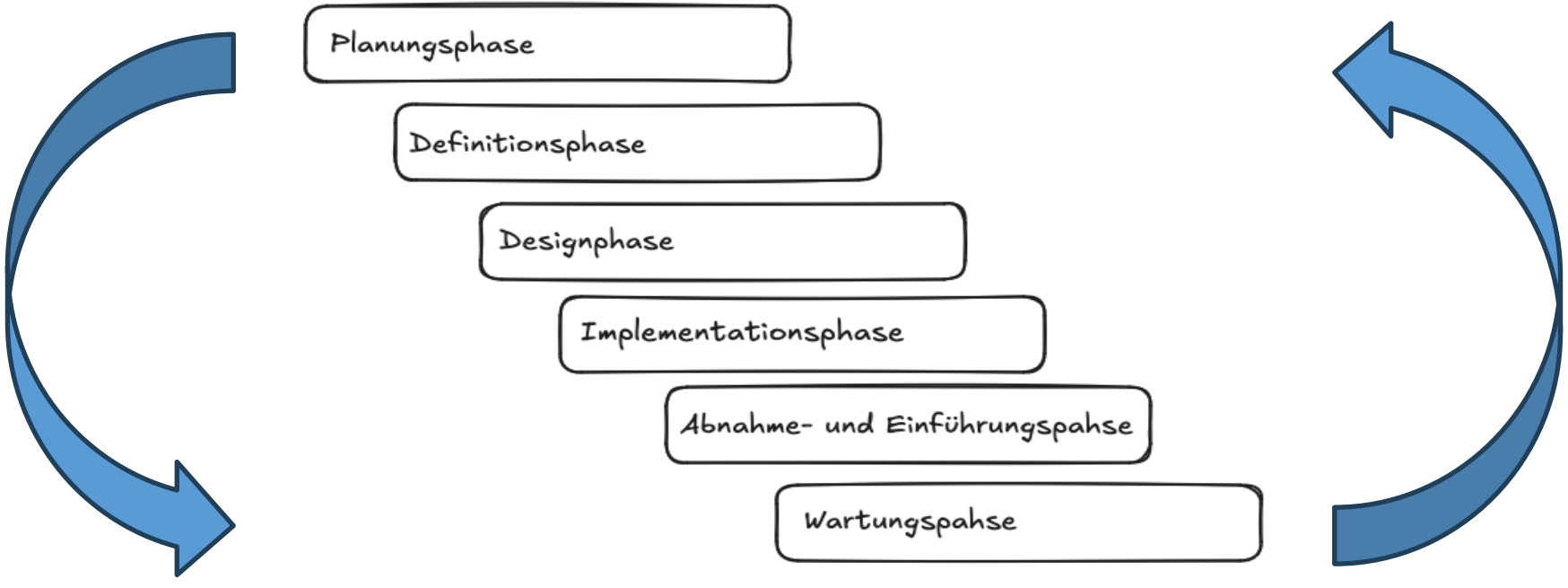
Was ist ein Vorgehensmodell?

- Abstrakte Darstellung für das Vorgehen während des Prozesses des SE
- Beschreibt die Lebenszyklen von SW
- Legt Aktivitäten fest
- Gibt eine Reihenfolge der Aktivitäten vor
- Jede Phase hat
  - Ziele
  - Aktivitäten und Rollenzuordnungen
  - Dokumentation
  - Methoden, Richtlinien, Konventionen, Werkzeuge

Warum braucht man ein Vorgehensmodell?

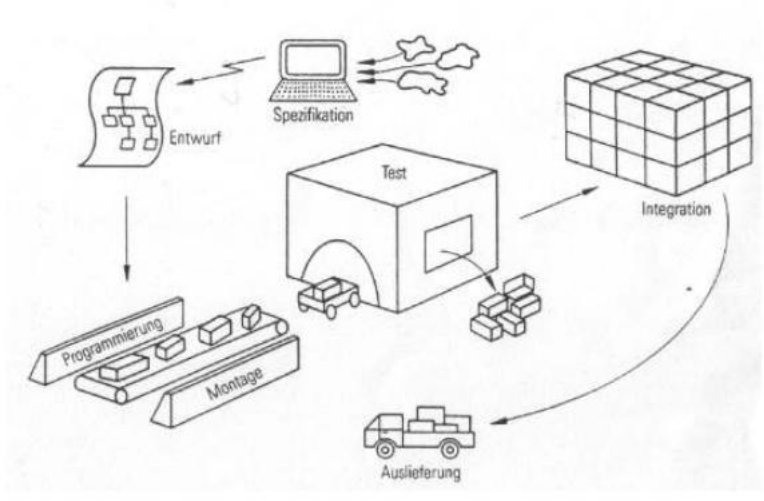
- Der Leitfaden für die Systementwicklung
- Anleitung für projektbegleitende Dokumentation
- Verbesserte Planbarkeit
- Möglichkeit der Zertifizierung (ISO… , IEC …)
- Höhere Personenunabhängigkeit
- frühzeitige Fehlererkennung durch festgeschriebene Testaktivitäten

Unterschiedliche Arten von Vorgehensmodellen



Software-Factory 1

- Software-Entwicklung = Fließbandproduktion
- Kleine, wiederverwendbare Module => kleine parallel arbeitende Teams
- Einsatz von Automatisierungstools
- Klar definierte Rollen (Programmierer, Tester, usw.)
- einheitliche Prozesse und Qualitätskontrollmaßnahmen
- Projektplanung



Software-Factory als Fließbandfabrik [vgl. Wie90]

Software-Factory 2

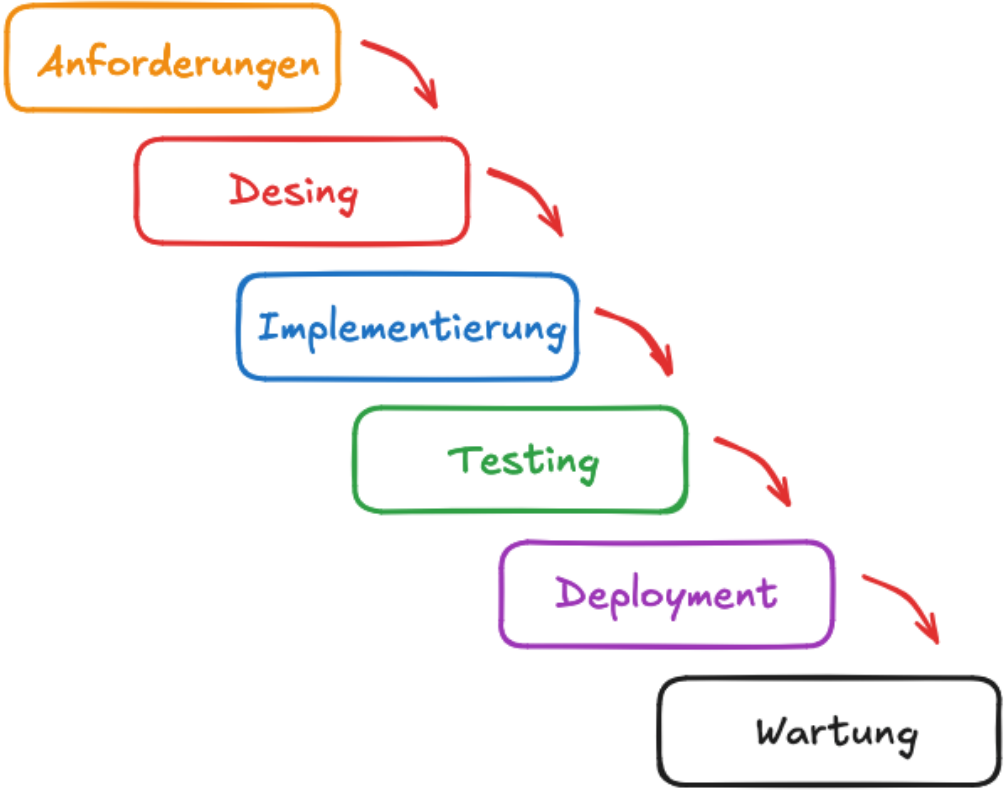
- **System Development Corporation (SDC)**
  - Ein berühmtes Projekt, an dem SDC beteiligt war, ist das SAGE-Projekt für die US-Luftwaffe, ein Frühwarnsystem zur Überwachung des Luftraums

- **Hitachi**
  - "Debugging Night Shift"
  - Export von Prozessen
  - Automatisiertes U-Bahn-System

Wassefallmodell

- das älteste und bekannteste Vorgehensmodell
- jede Phase des Software-Lebenszyklus wird strikt nacheinander abgearbeitet
- aus jeder Phase gehen definierte Dokumente hervor, die begutachtet und abgenommen werden

Wassefallmodell: 6 Hauptphasen



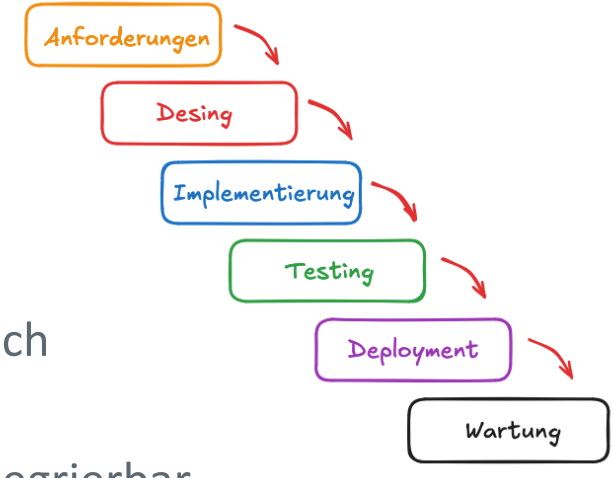
Wassefallmodell: Vor- & Nachteile

Vorteile

- Intuitiver, linearer Prozess: einfach zu verstehen, klarer Ablauf
- Nicht unterbrechbarer Prozess
- Top-down-Vorgang
- Gute Planbarkeit

Nachteile

- Starre Aufteilung in die Phasen
- Keine Rückkopplung auf frühere Phasen möglich
- Verpflichtungen zu frühen Zeitpunkten
- Neu dazu kommende Anforderungen nicht integrierbar



Wassefallmodell: Projekte, bei denen das Wasserfallmodell gut funktionieren kann

- Einsatz in eher kleinen Projekten mit eher wenigen Teilnehmern
- Infrastruktur- und Hardware-Entwicklung (z.B. Entwicklung eines neuen Betriebssystems oder eines Mikrocontrollers)
- Medizinische Software (z.B. Entwicklung von Software für MRT-Scanner, EKG-Geräte oder andere diagnostische und therapeutische Geräte)
- Verteidigung und Luftfahrt (z.B. Entwicklung von Steuerungssystemen für Raketen oder Flugzeuge)
- usw

Quiz:

- **Was ist das Hauptmerkmal des Wasserfallmodells?**
- **Welche der folgenden Aussagen ist typisch für das Wasserfallmodell?**

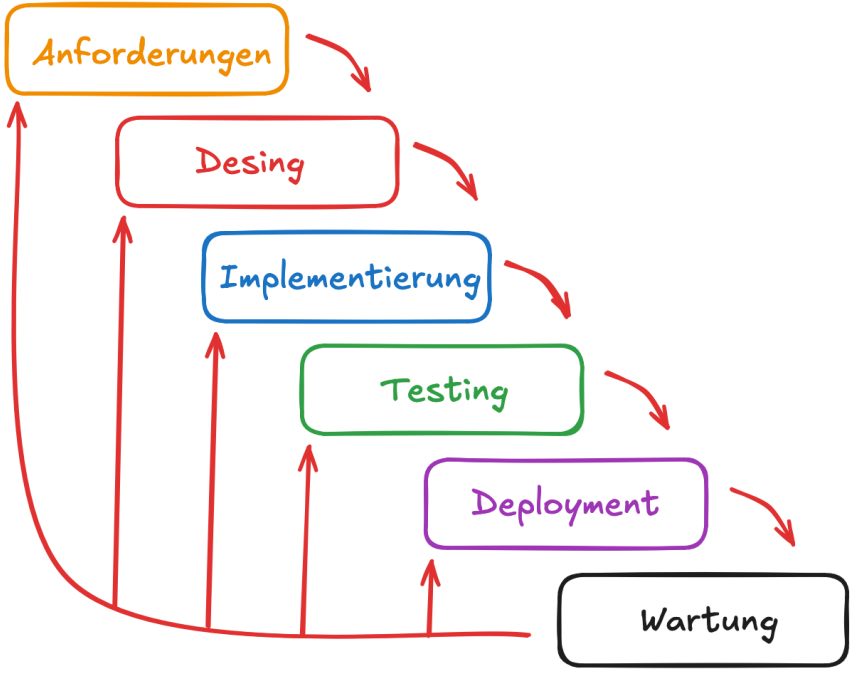
a) Der Entwicklungsprozess ist flexibel und anpassbar.

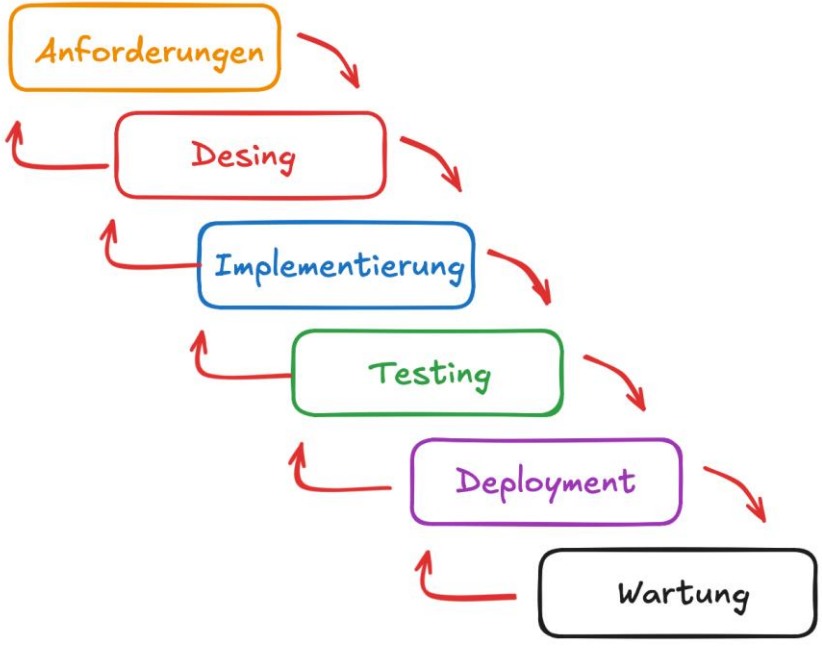
b) Jede Phase muss vollständig abgeschlossen sein, bevor die nächste Phase beginnt.

c) Fehler werden erst in der späteren Phase erkannt und behoben.

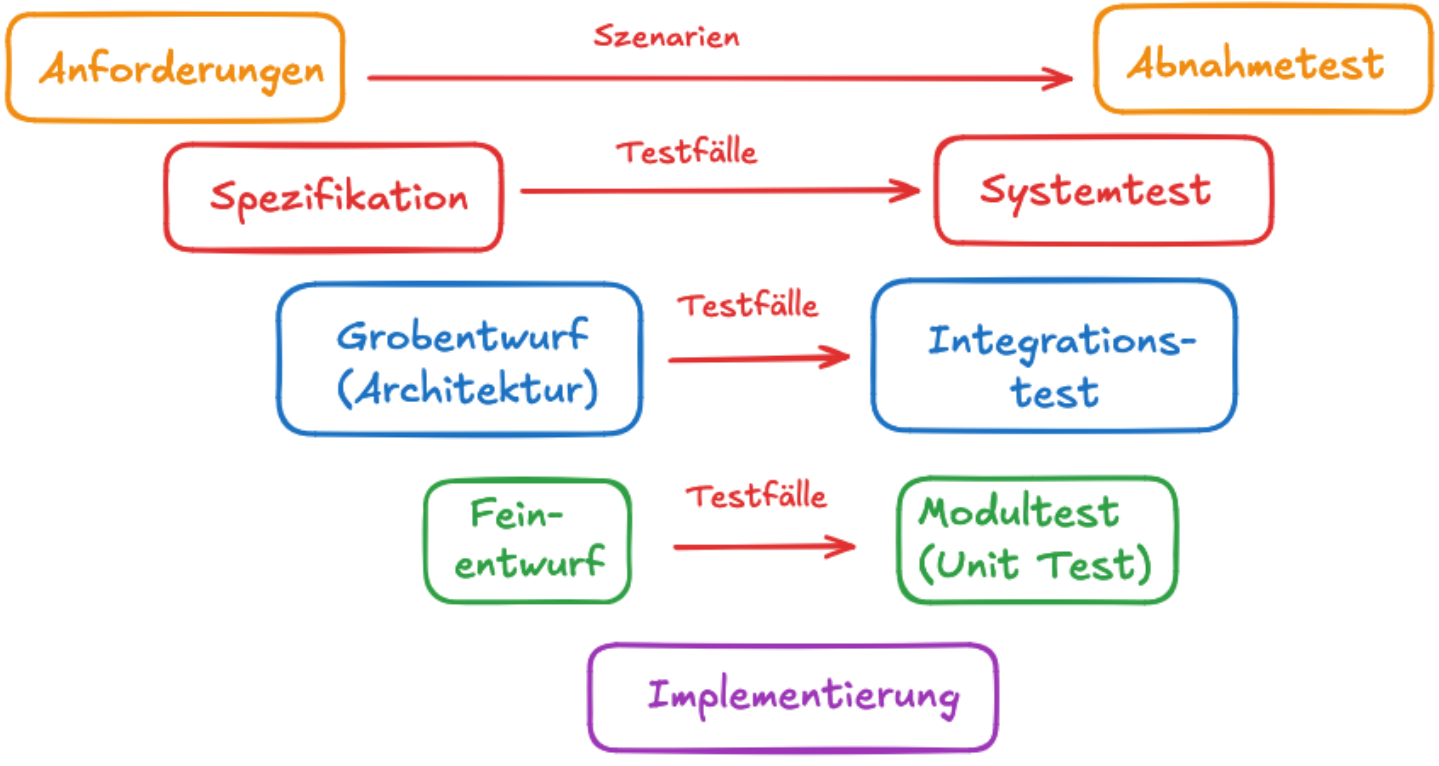
- **Was versteht man unter einer „Software Factory“?**

Verbesserte Wasserfallmodelle

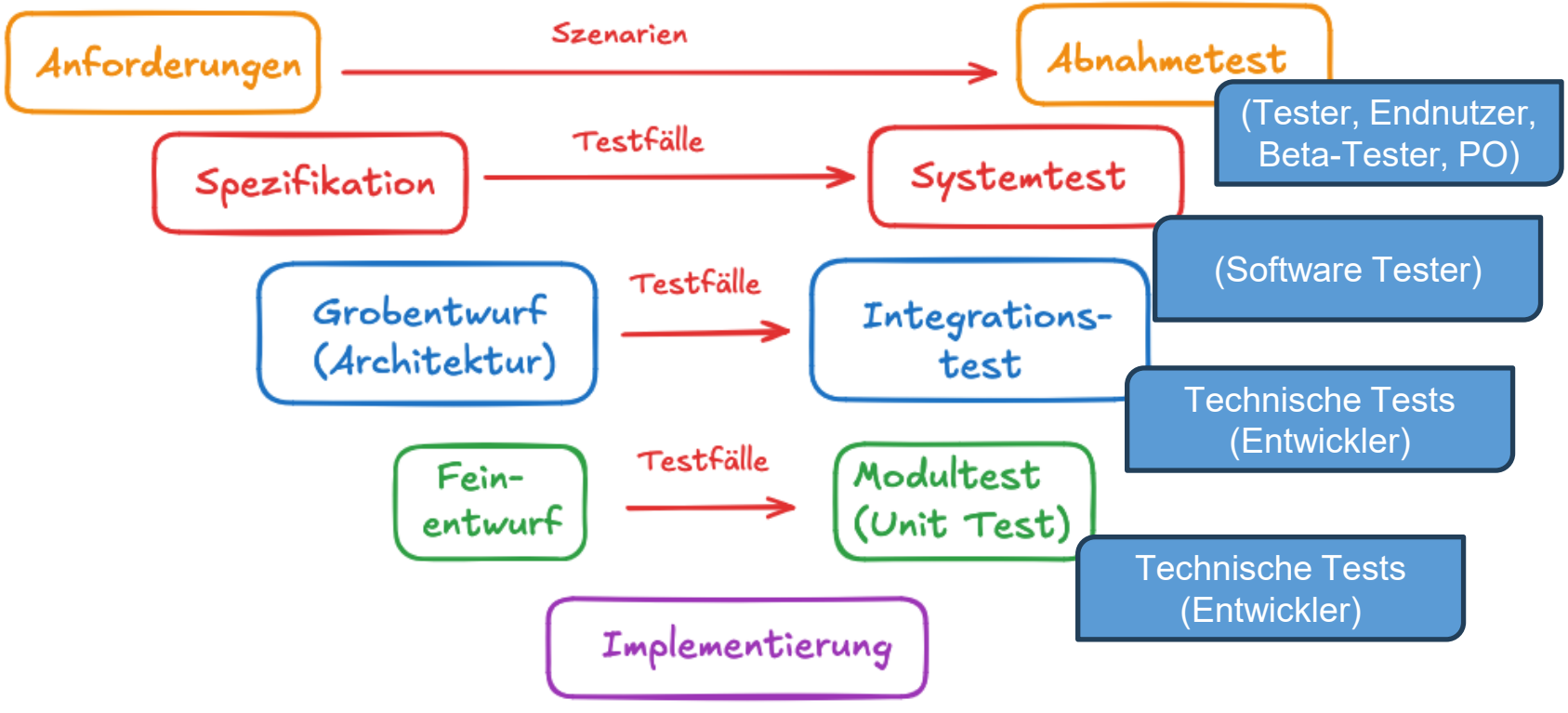




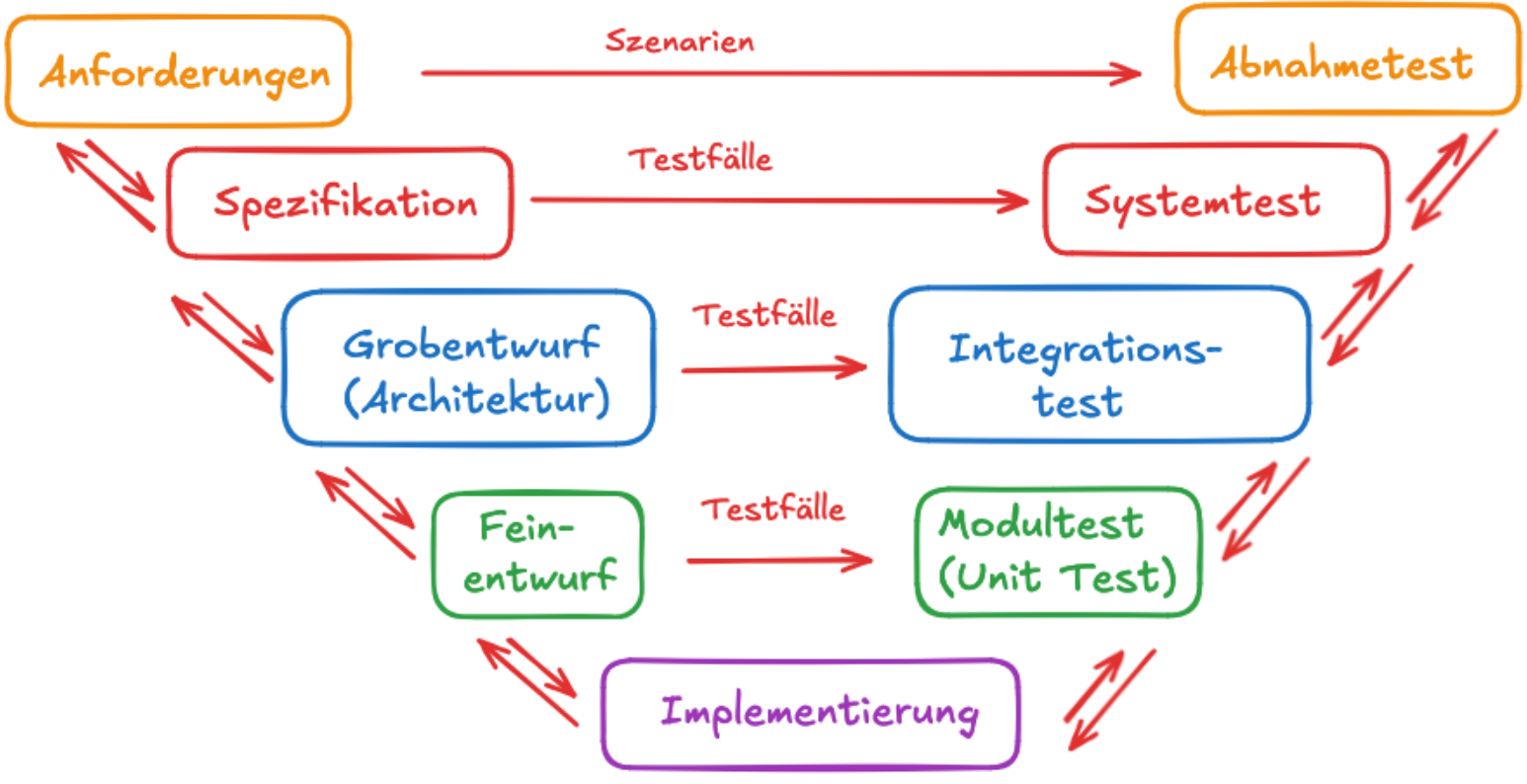
Das V-Modell 1



Das V-Modell 2



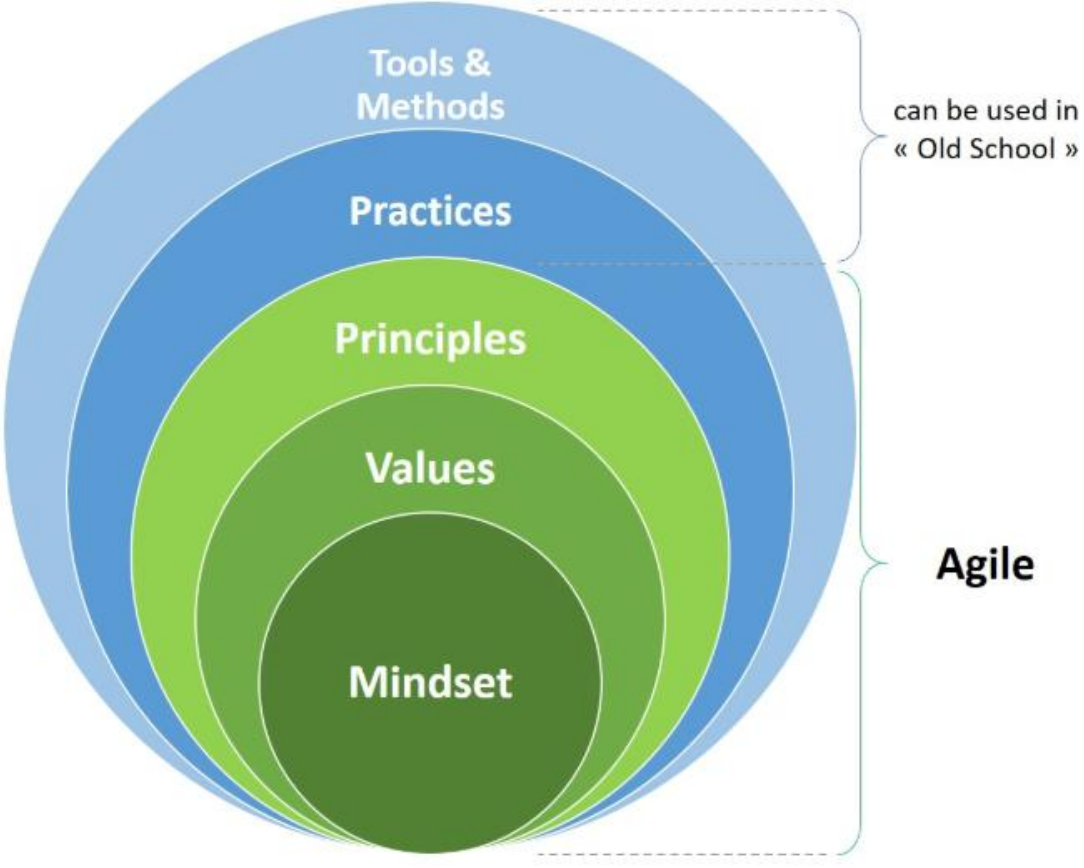
Das V-Modell 3



Quiz:

- Warum ist das V-Modell besser als Wasserfall?
- Warum gilt das V-Modell als sequentielles Modell?

Agile

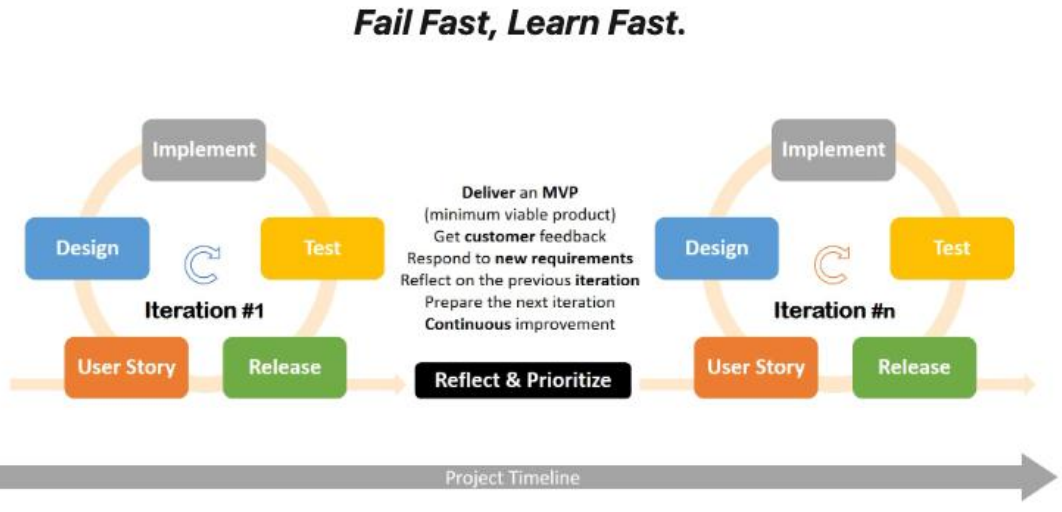

Tools &
Methods
Practices
Principles
Values
Mindset
can be used in
« Old School »
Agile

Haupteigenschaften von Agile

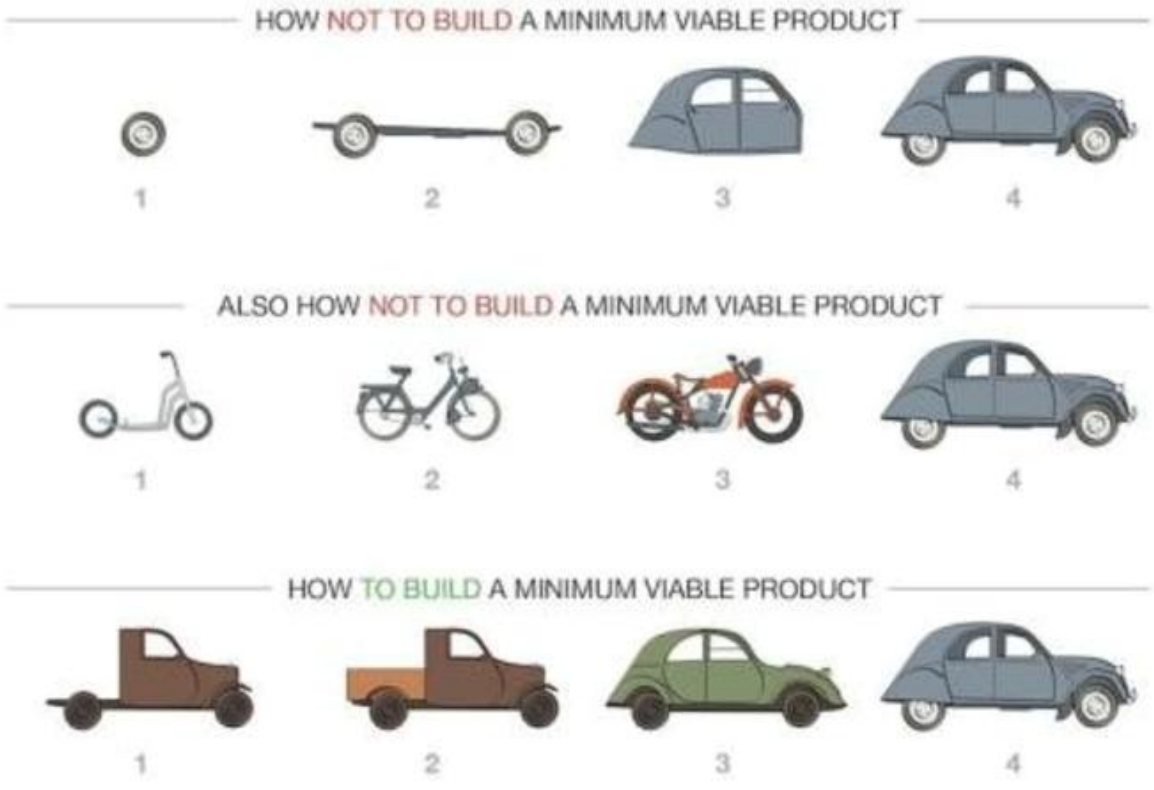
- Eine iterative und inkrementelle Entwicklung
- Eine auf den Menschen ausgerichtete Denkweise („A human-centric mindset“)

Iterative und Inkrementelle Entwicklung

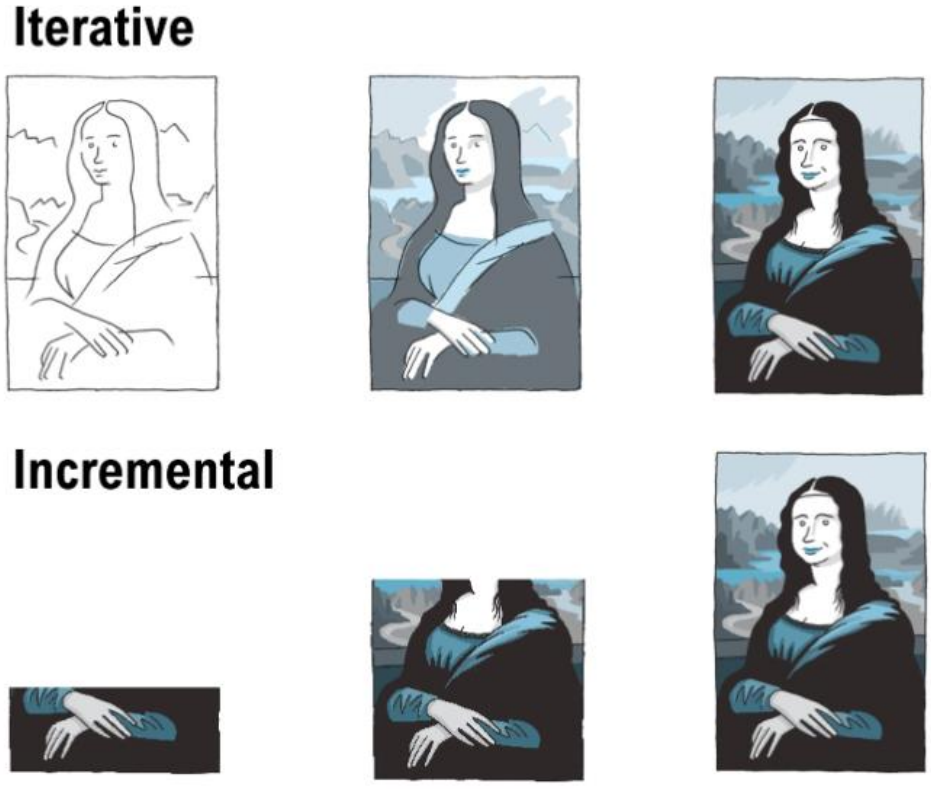
- Iterative, sehr kurze V-Zyklen
- Feedback vom Kunden bei jeder Lieferung
- Minimum Viable Product (MVP)



Iterativ versus Inkrementell



Iterativ versus Inkrementell



Agile: Human-centric Mindset

- Der Mensch (der Kunde und der Entwickler) sind im Mittelpunkt der Entwicklung
- Kundenzufriedenheit ist das oberste Ziel der Produkt-/Dienstleistungsentwicklung

Missverständnisse über Agilität

- Keine Dokumentation notwendig
- Keine Planung erforderlich
- Kein Management, keine Disziplin
- Agil = Scrum
- Agil funktioniert nur in der Softwareentwicklung
- Agil funktioniert nur mit jungen Mitarbeitern
- Agilität dient nur dazu, Geld zu sparen

TODO-App

Funktionale Anforderungen

xXxXxXxXXXXx Xxxx	Das System muss Aufgaben persistent speichern (z. B. lokal/DB).	Backend	MUSS
xxxXxxxxXXxx	Die App soll es dem Nutzer ermöglichen, innerhalb von maximal 5 Sekunden eine neue Aufgabe mit Titel anzulegen.	Performance	MUSS
xxXXxxxXxxxxx	Todo löschen	Funktional	MUSS
xxXXxXxxxxxxx	Todo bearbeiten	Funktional	MUSS
xxXxxxx	Todo als erledigt markieren	Funktional	MUSS
xXxxxXXxX	Das System muss Aufgaben in einer Liste anzeigen.	Frontend	MUSS
xxxXxxxxXXxx	Der Nutzer soll Aufgaben bis zu fünf Kategorien zuordnen können, die er selbst erstellen und löschen kann.	Frontend	Soll
xXxxXXXx	Das System soll Aufgaben nach Status (offen/erledigt) filtern können.	Frontend	SOLL
xxxXxxxxXXxx	Jede Aufgabe soll ein Fälligkeitsdatum enthalten können, das über einen Kalenderpicker in unter 3 Klicks ausgewählt werden kann.	Frontend	Kann

Übung: Überprüfung und Integration in Streamlit

- **Überprüfen Sie Ihr Design**: Stellen Sie sicher, dass mindestens fünf der vorgegebenen Anforderungen in Ihrem Design umgesetzt sind.
- **Falls erforderlich**: Ergänzen Sie die fehlenden Anforderungen in Ihrem Design.
- **Integration**: Übertragen Sie das vollständige Design in Ihre Streamlit-App, zunächst nur das Design!

Responsive Web Design (RWD)

Einführung in Responsive Web Design

- Ein responsives Design passt sich flexibel dem Ausgabemedium an.
- Es bietet damit unabhängig von der Auflösung des Ausgabemediums eine gleichbleibend gute User Experience und ein einheitliches Layout.

- **Warum wichtig?**
  - Viele Gerätetypen
  - Bessere User Experience
  - SEO-Vorteile

Grundprinzipien von RWD

- Die technische Basis bilden dabei HTML5 und CSS3.
- Grundelemente eines RWD:
  - Flexibles Layout
  - Media Queries
  - Flexbox

HTML5 und CSS3

- **HTML5:** Stellt den Inhalt und die Struktur einer Webseite bereit (Text, Bilder, Videos)
- **CSS3:** Bestimmt das Aussehen der Webseite (Farben, Layout, Animationen)

(1) Flexibles Layout: Verwendung relativer statt absoluter Größenangaben

- Flexible Schriftgrößen: em
  - Ein *em* gibt die Größe in Relation zur Schriftgröße eines Elements an.
    2em = doppelte Schriftgröße, 0.5em = halbe Schriftgröße, ...
  - Basis-Schriftgröße beträgt in der Regel 16px
  - Umrechnung: Zielgröße in px / Element-Schriftgröße = Zielgröße in em
- Flexible Breiten und Abstände: %
  - Gibt die prozentuale Länge bezogen auf die Gesamtlänge an.
    50% = halbe Größe, ...
  - Basisgröße ist durch den Viewport gegeben
- *Achtung:* Relative Angaben beziehen sich
  IMMER auf den Parent-Container:

```python
.container {
    width: 100%; /* Container nimmt 100% der Bildschirmbreite ein */
    padding: 5%; /* Der Abstand des Containers wird relativ zur Bildschirmgröße festgelegt */
}

.box {
    width: 50%; /* Box nimmt 50% der verfügbaren Breite ein */
    margin: 10px;
}
```

Von absoluten Pixelwerten hin zu relativen Angaben

```css
body {
  margin: 1.25rem;
  font-size: 1.5rem;
}

input[type="text"] {
  margin-top: 2.5rem;
  padding-top: 1.25rem;
  padding-bottom: 1.25rem;
  width: 100%;
  font-size: 2rem;
}
```

```css
button {
  margin-top: 1.25rem;
  padding: 1.25rem;
  width: 100%;
  font-size: 2rem;
}

#greeting {
  margin-top: 2.5rem;
  font-size: 3rem;
  font-weight: bold;
  text-align: center;
}
```

(2) CSS3 Media Queries

- Media Queries bieten die Möglichkeit, Eigenschaften des Ausgabemediums abzufragen und darauf zu reagieren.
- Wenn die Bedingungen einer Media Query erfüllt sind, wird das zugehörige CSS angewandt, andernfalls nicht.
- Beispiel: `@media (min-width: 512px) { ... }`
- Der Einsatz von Media Queries ist insbesondere dann sinnvoll, wenn Webseiten sowohl für Desktop-Browser, als auch für Tablets und Smartphones entwickelt werden.
- Aber auch bei reiner Mobile-Nutzung ergeben sich sinnvolle Anwendungsfälle, etwa das Anpassen des Layouts bei Drehung des Gerätes (Hochformat / Querformat).

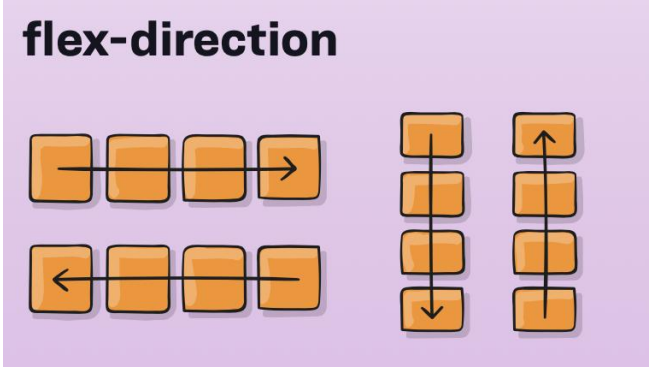
```css
/* Standard-Layout für größere Bildschirme */
body {
    font-size: 16px;
}

/* Für Bildschirme mit einer Breite von maximal 768px (Tablets und Smartphones) */
@media (max-width: 768px) {
    body {
        font-size: 14px; /* Kleinere Schriftgröße für mobile Geräte */
    }
    .container {
        padding: 10px; /* Mehr Abstand für kleinere Bildschirme */
    }
}

```

Media Queries: Testbare Gerätefeatures

Geräteeigenschaft	Bedeutung	min/max Variante
`width`	Die Breite des Viewport	✅
`height`	Die Höhe des Viewport	✅
`orientation`	Entweder Hochformat (`portrait`) oder Querformat (`landscape`).	❌
`aspect-ratio`	Das Seitenverhältnis des Viewport, z.B. 16/9	✅
`color`	Die Anzahl der Bits pro Farbkomponente, z.B. 8 für 8 Bit. Schwarz-weiß-Geräte haben einen Wert von 0.	✅
`color-index`	Die Anzahl der Farben in der Farbtabelle, z.B. 256.	✅
`monochrome`	Die Anzahl der Bits pro Pixel für ein Monochrome-Gerät.	✅
`resolution`	Die Pixeldichte, z.B. 300dpi.	✅

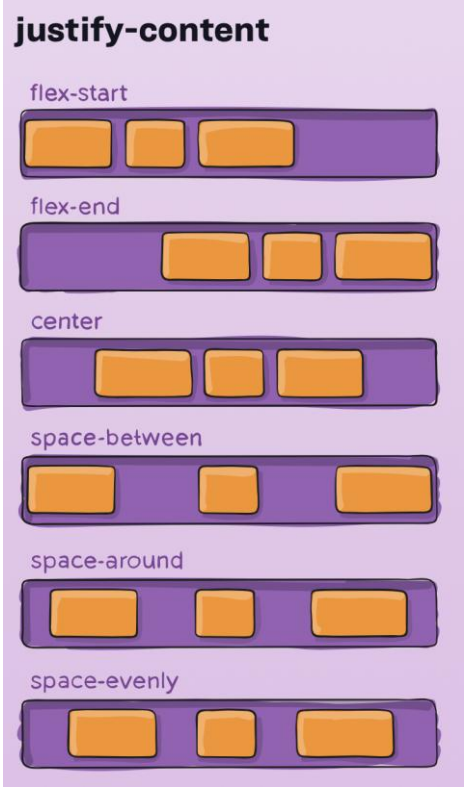
(3) Flexbox

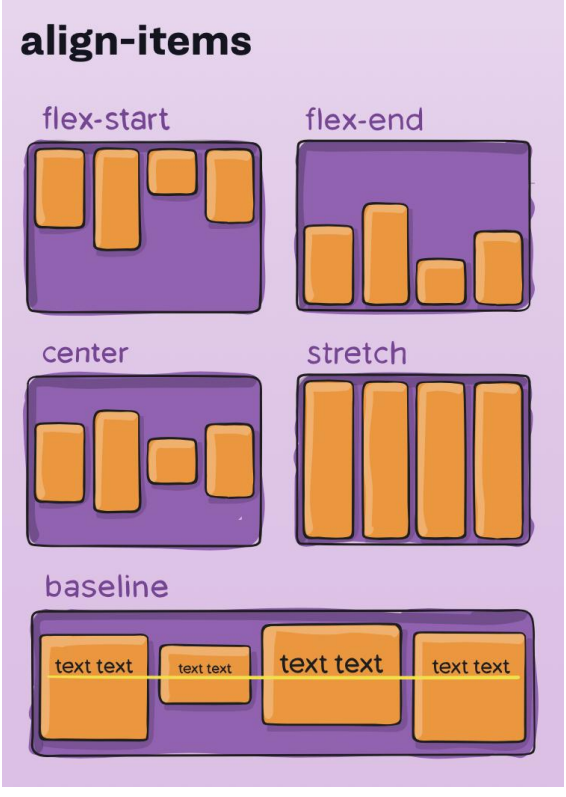
- CSS Layout System durch Definition von `display: flex;`
- Definition von Zeilen und Spalten für einen Container `flex-direction: row | column`
- Die untergeordneten Elemente können entlang der main- und cross-axis verteilt werden `justify-content` und `align-items`



(3) Flexbox

Vollständiger Guide mit vielen Beispielen: https://css-tricks.com/snippets/css/a-guide-to-flexbox/





Beispiel für Flexbox

```css
.container {
    display: flex;
    flex-wrap: wrap; /* Ermöglicht den Umbruch der Elemente, wenn der Platz nicht ausreicht */
    justify-content: space-between; /* Verteilt die Elemente gleichmäßig */
}

.box {
    flex: 1 1 30%; /* Jedes Box-Element nimmt mindestens 30% der Breite ein und wächst bei Bedarf */
    margin: 10px;
}

```

Verwendung von CSS in Streamlit

1. Einfaches CSS

```
/* Beispiel für ein responsives Layout */
@media only screen and (max-width: 768px) {
    .block-container {
        padding: 10px;
        font-size: 12px;
    }
}

@media only screen and (min-width: 769px) {
    .block-container {
        padding: 30px;
        font-size: 16px;
    }
}
```

2. CSS in Streamlit integrieren

```
st.markdown(
    """
    <style>
.block-container {
     @media only screen and (max-width: 768px) { … }
}
    </style>
    """,
    unsafe_allow_html=True
)

st.title('Responsive Web-Design mit Streamlit')
st.write("Dies ist eine einfache Beispielanwendung.")
```

3. Verwendung von Streamlit-Komponenten für Layout-Anpassungen

```
# Layout mit Spalten (responsive)
col1, col2 = st.columns([2, 1])

with col1:
    st.header("Hauptinhalt")
    st.write("Dieser Bereich ist für den Hauptinhalt.")

with col2: …
```

4. Responsive Widgets und Interaktivität

```
col1, col2 = st.columns([2, 1])

# Button in der ersten Spalte
with col1:
    if st.button('Klicken Sie mich'):
        st.write("Button wurde geklickt!")

# Auswahlfeld in der zweiten Spalte
with col2:
    option = st.selectbox(
        'Wählen Sie eine Option',
        ('Option 1', 'Option 2', 'Option 3')
    )
    st.write(f'Sie haben {option} gewählt.')
```

5. Verwendung von st.image() für adaptive Bilder

```
# Responsive Bildgröße
st.image('https://via.placeholder.com/150', use_column_width=True)
```

Übung: Responsive TODO-App in Streamlit

- Aufgabe: Implementieren Sie ein responsives Design für die TODO-App in Streamlit, das auf verschiedenen Geräten (Desktop, Smartphone) gut aussieht und benutzerfreundlich ist