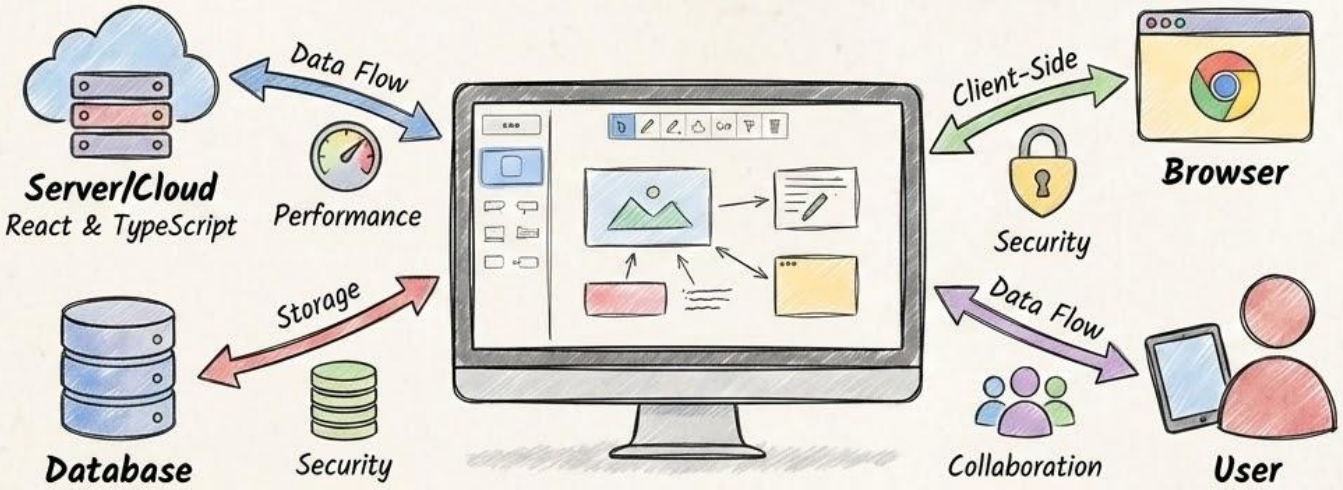
Excalidraw



Teknisk Arkitektur & Implementation

En djupdykning i modern webbutveckling

Vad är Excalidraw?

Webbaserad ritapplikation (whiteboard)

Skapar handritade diagram och skisser

Gratis och open-source

Ingen installation behövs

Möjlighet att samarbeta i realtid

+1000 automatiserade tester

Hur gammalt är projektet Excalidraw startade den 1 januari 2020 Excalidraw+, så projektet är ungefär 6 år gammalt. Det var Christopher Chedeau, en mjukvara ingenjör på Facebook, som initialt började arbeta på verktyget web.dev.

Aktivitetsnivå Projektet är mycket aktivt. Huvudprojektet uppdaterades senast den 4 januari 2026 och är skrivet i TypeScript med en MIT-licens GitHub. GitHub visar att projektet har över 110 000 stjärnor och 11 500 forking.

Bidragsgivare Efter ett år hade projektet över 100 bidragsgivare Excalidraw+, och numret har växt sedan dess. En källa från cirka 2021 nämner 149 bidragsgivare Excalidraw+. Det är svårt att få ett exakt aktuellt antal, men det är tydligt ett aktivt projekt med många bidragsgivare.

Kodens omfattning Huvudprojektet består av cirka 113 686 rader TypeScript-kod GitHub enligt GitHub. Projektet är organiserat som ett monorepo med flera npm-paket.

Stora företag/organisationer som använder det Stora användare inkluderar Google Cloud, Meta, CodeSandbox, Obsidian Excalidraw, Replit, Slite, Notion, HackerRank och många andra GitHub.

Kravspecifikation Jag hittade inget formellt kravdokument, men projektet har dokumentation för utvecklare och en betongsdokumentation GitHub för integration. Det finns också en roadmap som starkt rekommenderas att gå igenom Excalidraw för potentiella bidragsgivare.

Storlek på projektet Total files : 1708, Total code lines : 741261, total comment lines : 8939, total blank lines : 31980

Sammanfattningsvis är Excalidraw ett väletablerat, aktivt utvecklat projekt med bred industristöd och en växande community av bidragsgivare.

Projektöversikt & Status

Hur gammalt är projektet

- Start: 1 jan 2020 (~6 år gammalt)
- Skapare: Christopher Chedeau (Meta)
- Ursprungligen för web.dev.

Aktivitetsnivå

- Mycket aktivt (Uppdat. 4 jan 2026)
- TypeScript, MIT-licens
- 110k+ Stjärnor, 11.5k Forking

Bidragsgivare

- Över 100 efter år 1
- Växande antal (Källa 2021: ~149)
- Högt deltagande

Kodens omfattning

- ~113 686 rader TypeScript (GitHub)
- Monorepo-struktur, flera npm-paket.

Stora företag/ organisationer som använder det

- Google Cloud, Meta, CodeSandbox, Obsidian, Replit, Slite, Notion, HackerRank, m.fl.

Kravspecifikation

- Ingen formell spec.
- Dev-dok. & Roadmap finns (GitHub)
- Rekommenderas för bidragsgivare.

Storlek på projektet

- Total files: 1708
- Total code lines: 741261
- Total comment lines: 8939
- Total blank lines: 31980

Sammanfattningsvis: Vältablerat, aktivt projekt med brett industristöd och växande community.

SLIDE 3: Tech Stack - Vad används?

React (19.0.10)
UI Framework

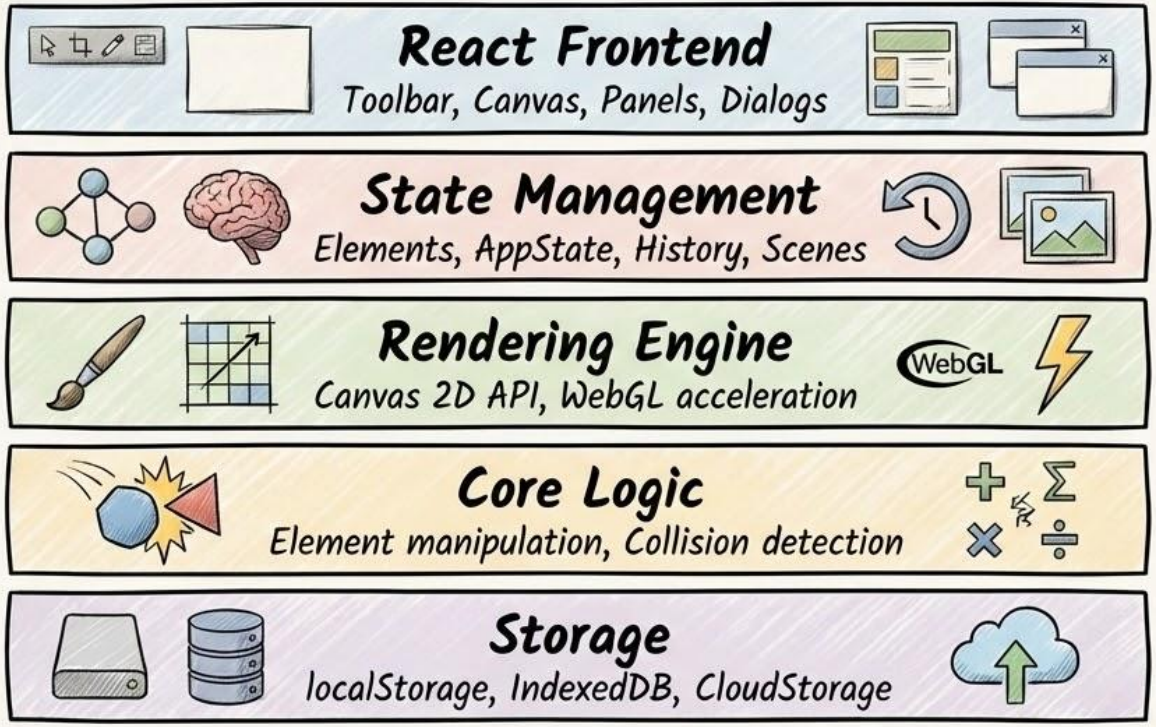
TypeScript (5.9.3)
Type Safety

Vite (5.0.12)
Build Tool

Canvas 2D
Rendering

Vitest (3.0.6)
Testing

SLIDE 4: Arkitektur - Lager

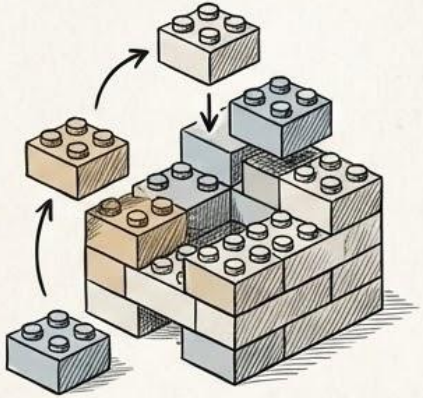


"**Varför lager?** Separation of concerns - varje lager gör en sak bra"

SLIDE 5: Varför React?

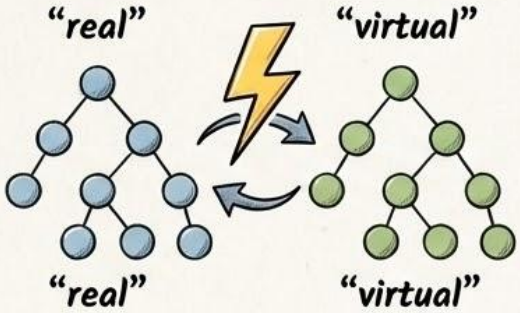
🧩 Komponentbaserad

Återanvändbar kod, enkel att organisera



⚡ Virtual DOM

Effektiv rendering av UI-ändringar

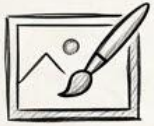
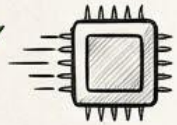


🌐 Community

Stor community, många bibliotek

SLIDE 6: Canvas vs SVG

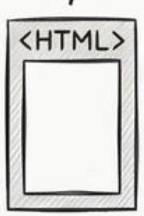
Canvas (Vald):

- ✓  Högt performance
- ✓  Perfekt för ritprogram
- ✓  GPU acceleration
- ✓  Handritad känsla

SVG (Ej vald):

- ✗  Långsammare med många element
- ✗  Svårt att implementera effekter
- ✓  Bättre för vektor-grafik
- ✓  Enklare manipulation

SLIDE 7: Canvas Rendering Pipeline



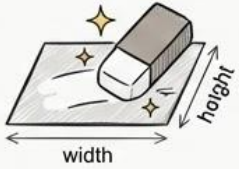

```
// 1. Hämta context
const ctx = canvas.getContext('2d');

// 2. Rensa
ctx.clearRect(0, 0, width, height);

// 3. Applicera transformationer
ctx.transform(...);

// 4. Rita alla element
elements.forEach(el => {
  ctx.fillRect(el.x, el.y, el.width, el.height);
});
```

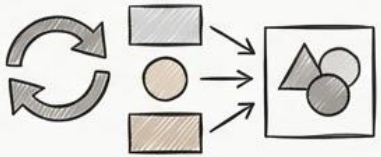
1. Hämta Context



2. Rensa

3. Applicera Transformationer

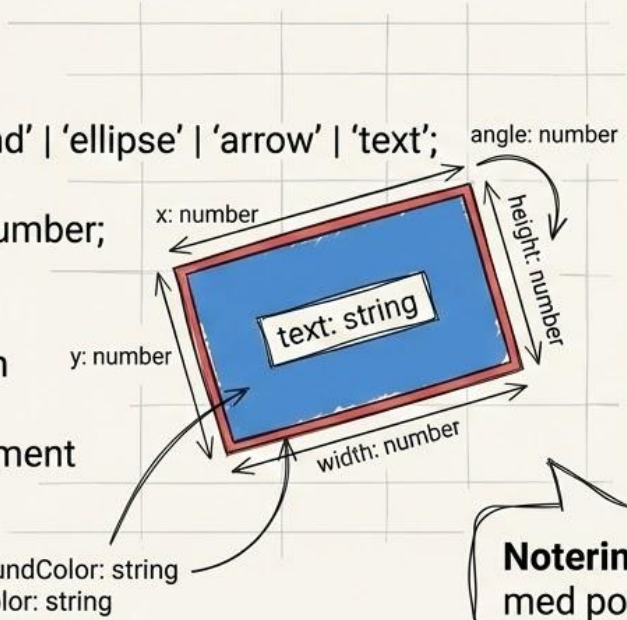
4. Rita Alla Element



Notering: Två canvas-lager: Interactive (musinteraktion) + Static (innehåll)

SLIDE 8: Element Datastruktur

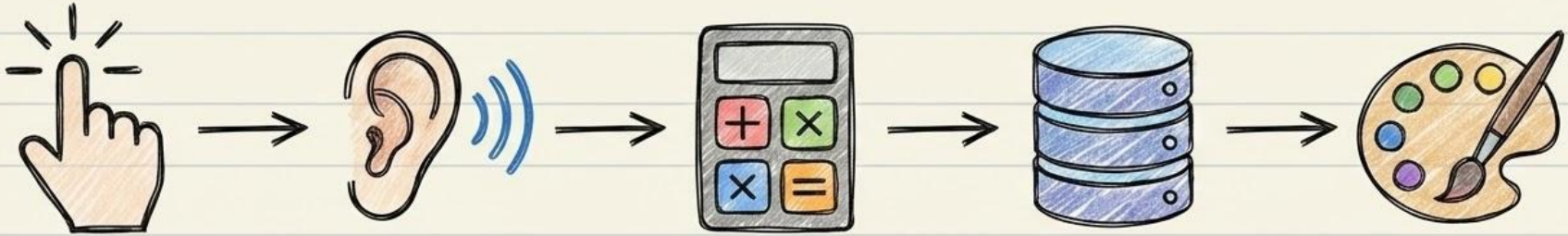
```
interface Element {
  id: string;
  type: ‘rectangle’ | ‘diamond’ | ‘ellipse’ | ‘arrow’ | ‘text’;
  x: number;  y: number;
  width: number;  height: number;
  backgroundColor: string;
  strokeColor: string;
  angle: number;  // rotation
  locked: boolean; 🔒
  text: string; // för text-element
}
```




```
interface Element {
  id: string;
  type: 'rectangle' | 'diamond' | ...;
  x: number; y: number;
  width: number; height: number;
  backgroundColor: string;
  strokeColor: string;
  angle: number; // rotation
  locked: boolean;
  text: string;// för text-element
}
```

Notering: Varje form är ett objekt med position, stil, och egenskaper

SLIDE 9: Interaction Event Loop



User Action
Klick, drag, tangent

Event Listener
onPointerDown, onPointerMove, onPointerUp

Calculate
Vad ändrade sig?

Update State
Element, AppState

Re-render
Rita på canvas

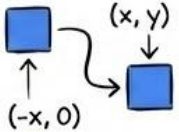
SLIDE 10: Exempel - Drag an Element

Pointer Down

```
onPointerDown = (event) => {
  state.draggedElement =
getElementAtPosition(event.x,
event.y);
}
```

Pointer Move

```
onPointerMove = (event) => {
  if (state.draggedElement) {
    state.draggedElement.x =
    state.draggedElement.y =
event.y - offset.y;
    render();
  }
}
```



Pointer Up

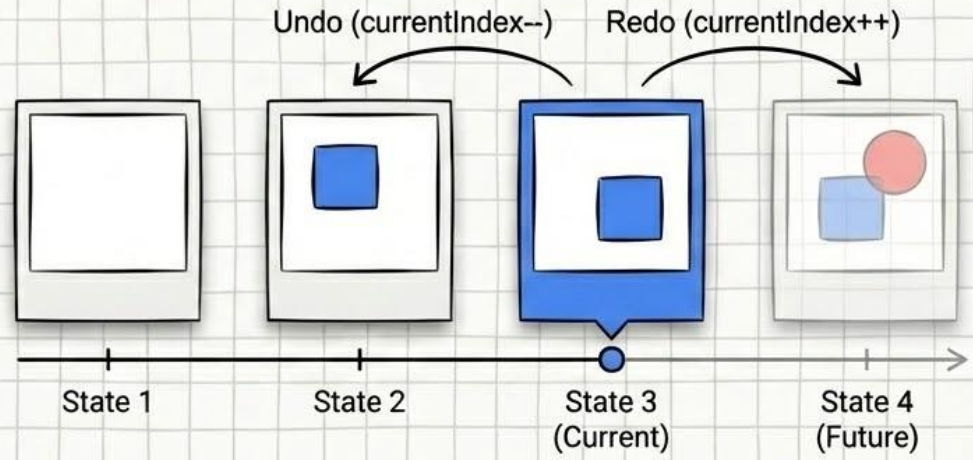
```
onPointerUp = () => {
  addToHistory(state); // För undo
}
```

SLIDE 11: Undo/Redo System

```
class History {
  entries = [];
  currentIndex = -1;

  addEntry(elements, appState) {
    this.entries = this.entries.slice(0,
this.currentIndex + 1);
    this.entries.push({ elements,
appState });
    this.currentIndex++;
  }

  undo() { this.currentIndex--; }
  redo() { this.currentIndex++; }
}
```



Notering: Spara snapshot av varje tillstånd → enkelt undo/redo

SLIDE 12: Storage & Persistence

localStorage

- 5-10MB.
- Snabb.
- Synkron.
- Lokalt.

IndexedDB

- 50MB+.
- Asynkron.
- Ingen blockering.

Cloud

- Obegränsad.
- Samarbete.
- Synk mellan enheter.

Notering: Excalidraw använder: localStorage för små diagrams, IndexedDB för större

SLIDE 13: Testing - 1079

Tests!

Test-resultat:

964 tests passar

68 snapshot mismatches (ej kritiskt)

46 skipped

Vitest

Test runner (snabb, moderna)

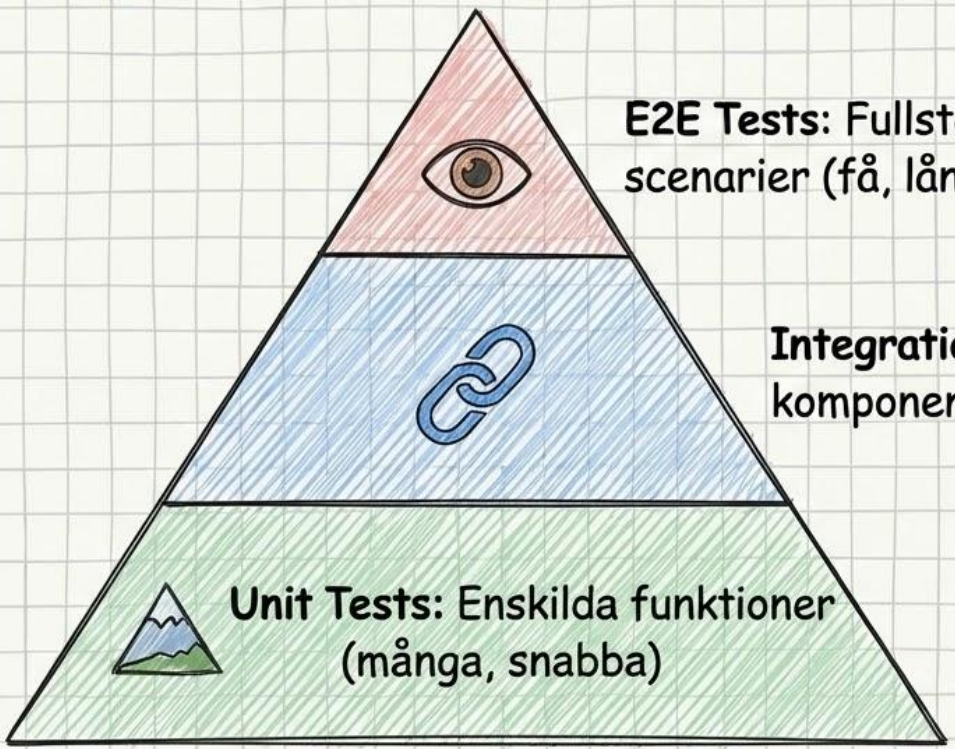
@testing-library

Test som användare gör

vitest-canvas-mock

Mock Canvas API för tests

SLIDE 14: Test Pyramid

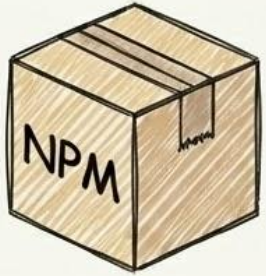


Notering: Excalidraw fokuserar på: Integration + Unit (viktigast för UI)

SLIDE 15: Deployment - 3 Alternativ

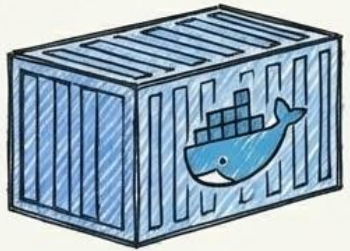
1 Web App

- Deploy till GitHub Pages, Netlify



2 NPM Package

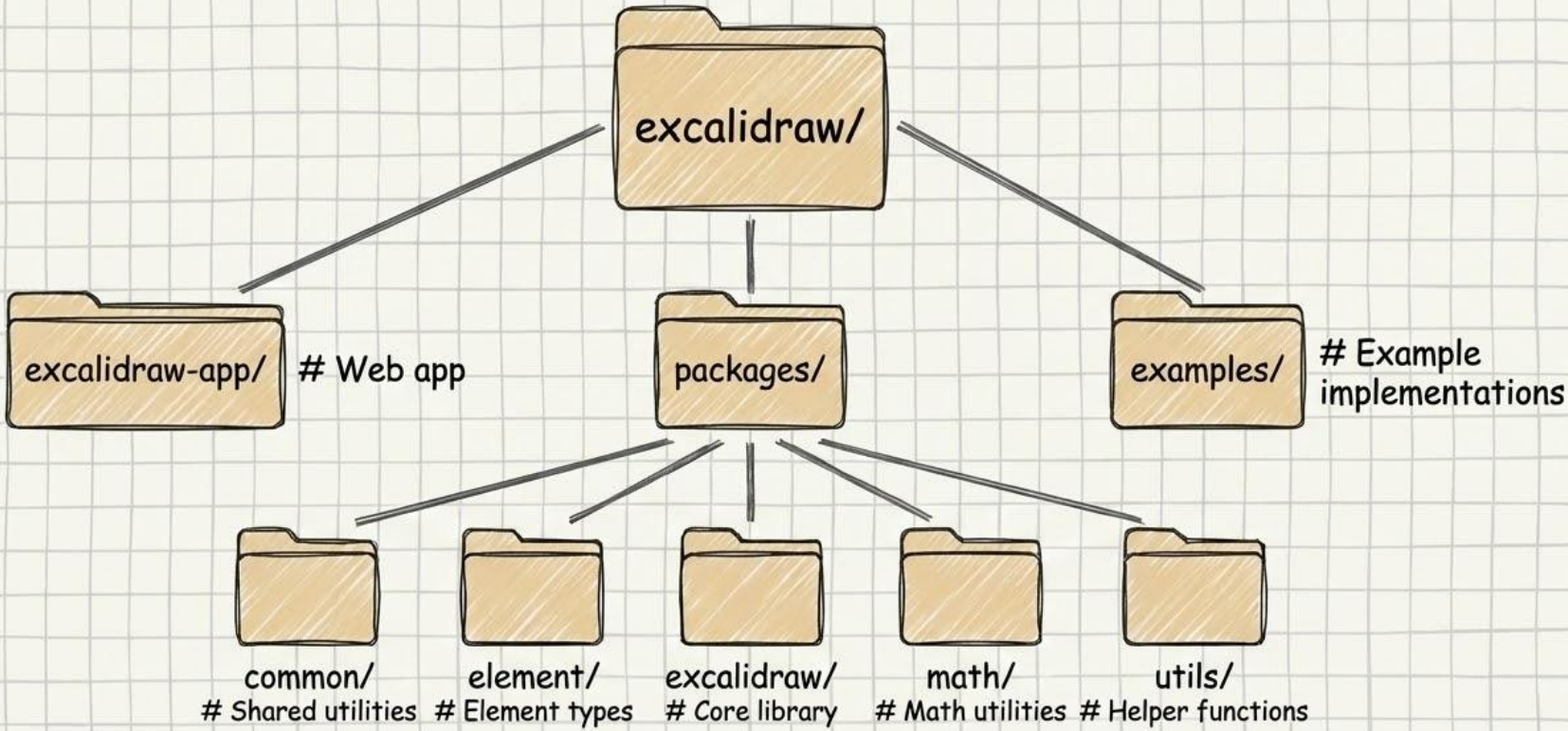
- Andra appar kan importera det



3 Docker

- Container, server-deployment

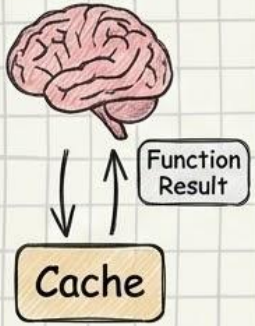
SLIDE 16: Monorepo - Projektstruktur



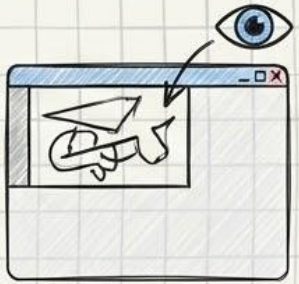
Notering: Fördel: Shared code, coordinerad versionering

SLIDE 17: Performance Optimizations

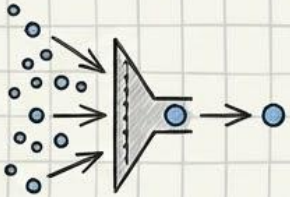
Memoization: Cache funktions-resultat



Selective Rendering: Rita bara synliga element

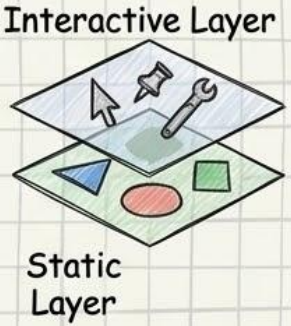


Event Debouncing: Begränsa event-frekvens



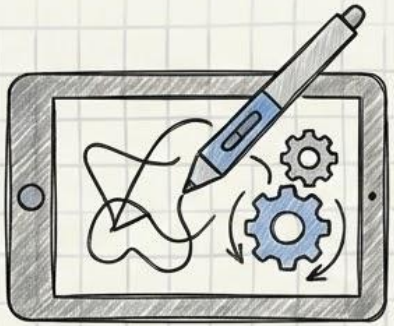
Virtual Scrolling: Rendera bara synliga i listor

Two Canvas Layers: Separate interactive och static



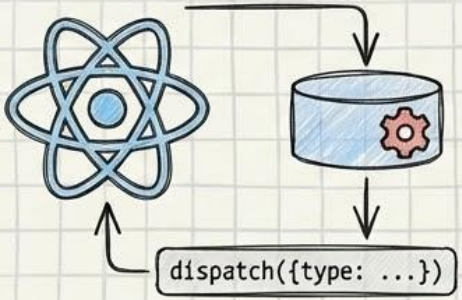
SLIDE 18: Architecture Decisions & Tradeoffs

Canvas: Performance



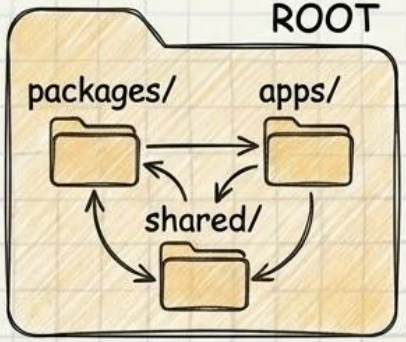
Tradeoff: Svårare att implementera vissa features

useReducer: Modern React



Tradeoff: Inte optimerad för mycket stora states

Monorepo: Shared code



Tradeoff: Mer komplext setup

SLIDE 19: Development Workflow

Setup & Install

Installera

```
yarn install
```

Development Loop

Starta dev-server

```
yarn start
```

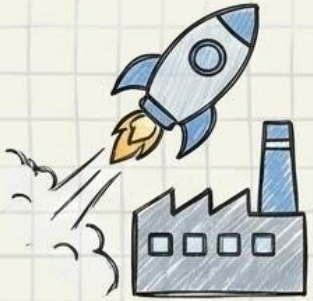
Köra tester

```
yarn test:app
```

Code quality

```
yarn test:code # ESLint
yarn test:typecheck
yarn test:all  # Allt
```

Production Build



Build för production

```
yarn build
```

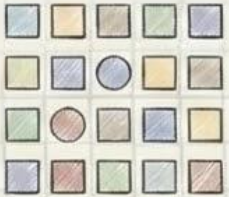
Scaling - 10,000+ Element

Problem:

- Långsam rendering
- Memory leak
- Laggy interaktion

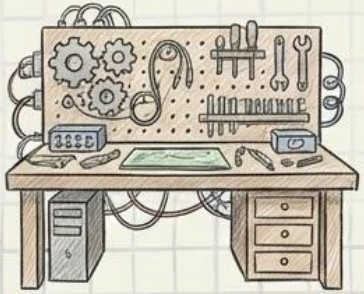
10,000+ ELEMENT

Lösningar:

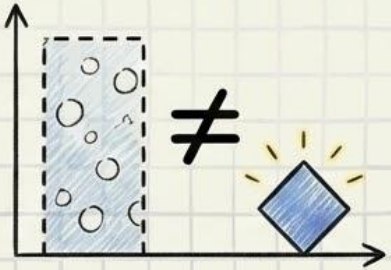


- Spatial indexing
- Culling (rita bara synliga)
- Throttling
- Worker threads

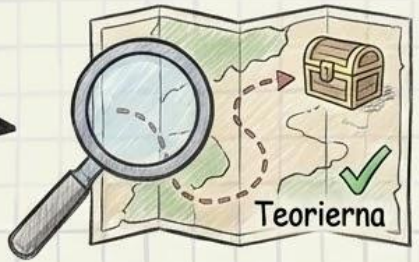
SLIDE 21: Key Learnings



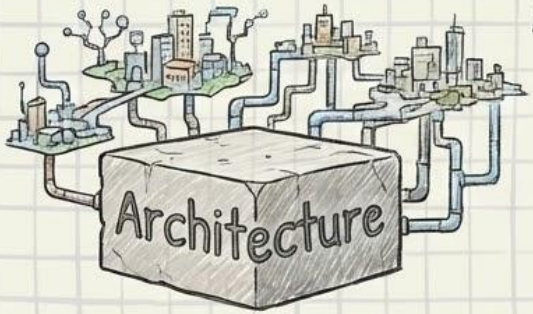
Test-setup är kritiskt



Coverage ≠ Quality



Exploratory testing bekräftar teorierna

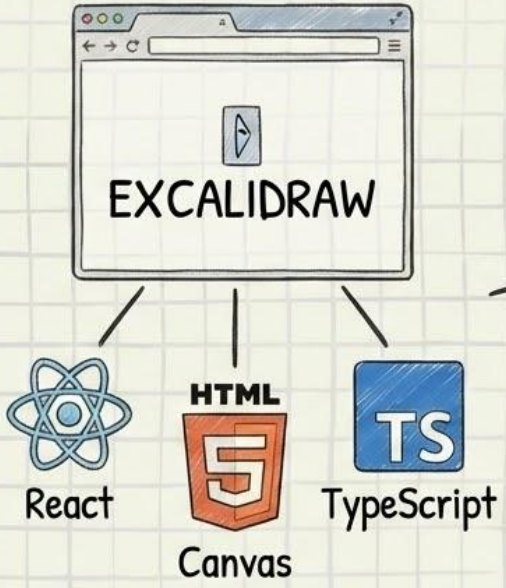


Architecture decisions påverkar allt

Performance kräver konstant focus

Open-source = högt kodkvalitet

Sammanfattning



Byggt på moderna web-teknologier

Fokus på performance, testning, och user experience

Ett utmärkt exempel