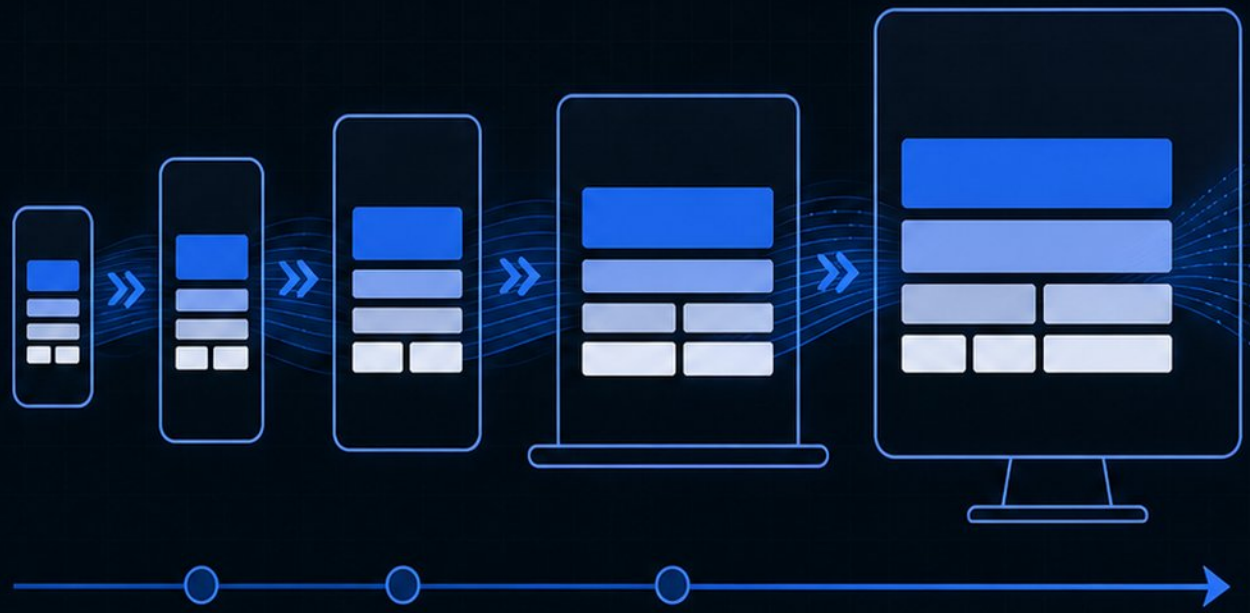
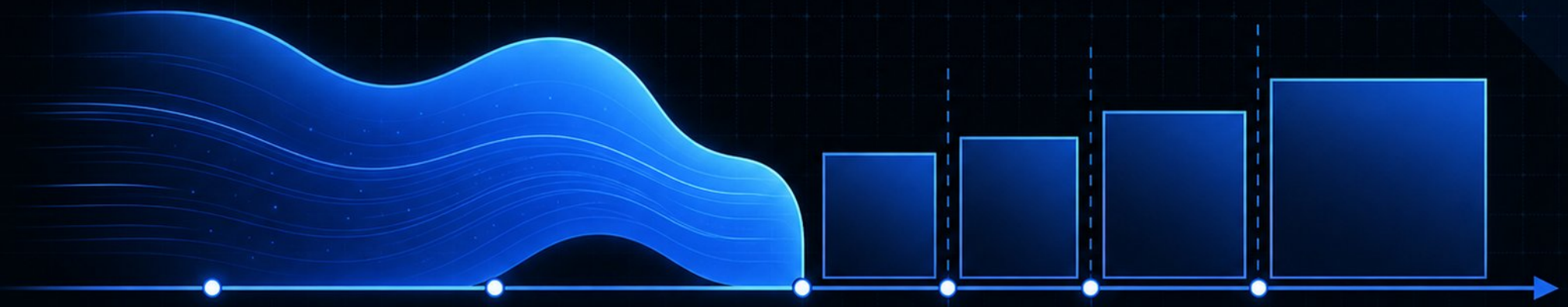
Introduction to Designing Responsive Layouts

- **Responsive design:** one codebase that fluidly adapts layout, typography, and media to any screen size
- **Adapt** a single content structure across breakpoints, not separate mobile, tablet, and desktop versions
- **Course scope:** practical fluid grids, flexible media, container queries, and actionable workflows for designers
- **Designer pain points:** context-breaking components, endless breakpoint patches, and testing bottlenecks



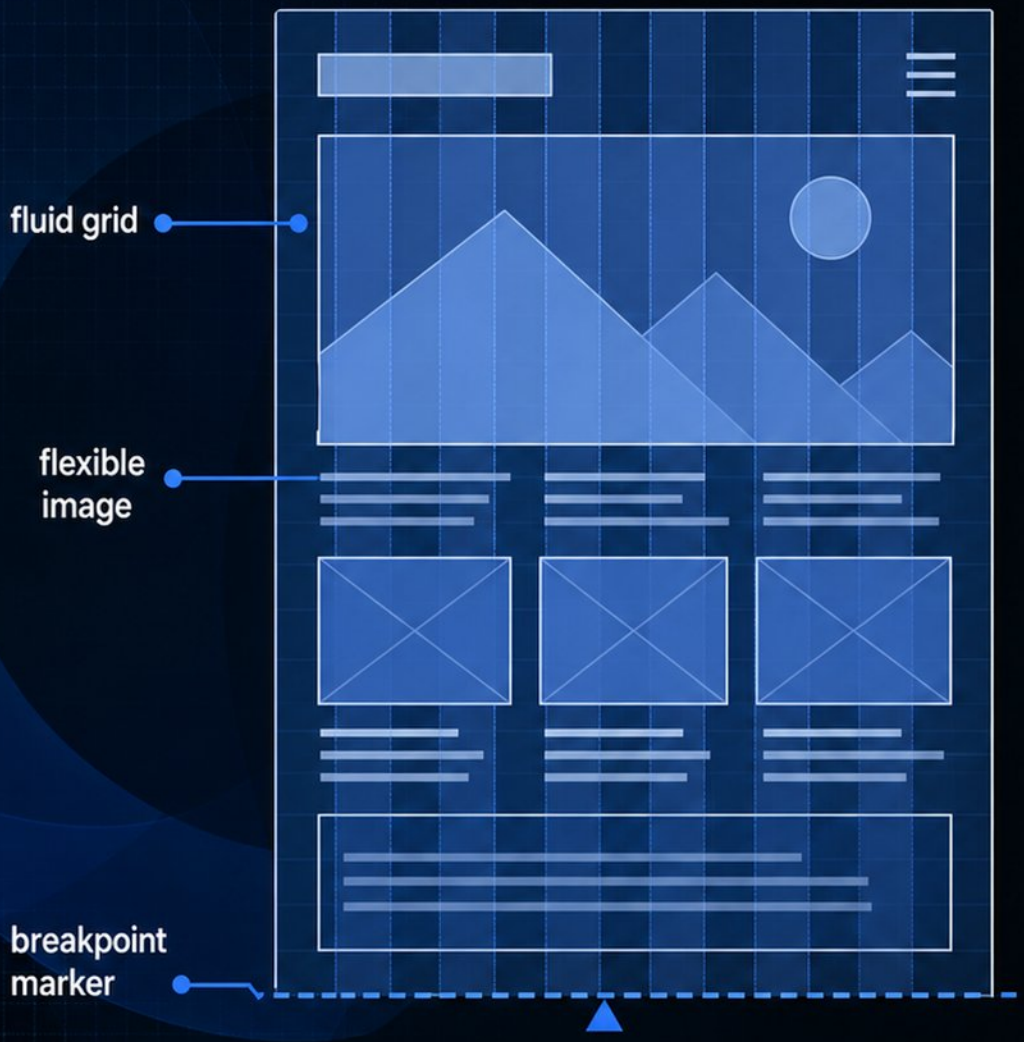
Responsive vs. Adaptive: Understanding the Distinction



- - Fluid, proportional scaling in one codebase and one URL
- - From 2010 proposal to 2026 Grid, clamp(), container queries

- - Fixed layouts at set breakpoints, each with a version
- - Higher maintenance and SEO costs for device-specific builds

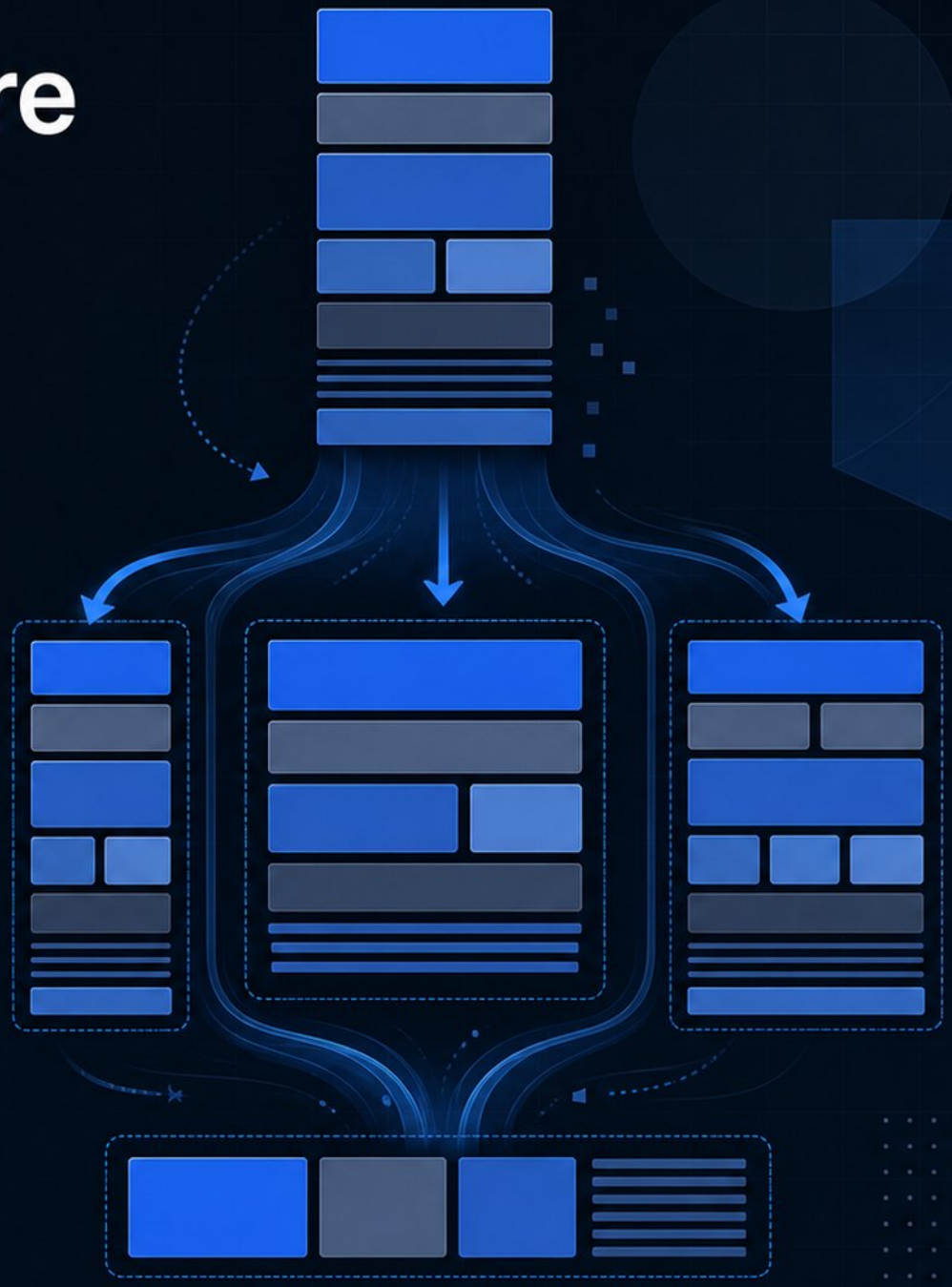
Core Principles of Responsive Design



- **Fluid grids:** use relative units like %, fr, rem, and vw instead of fixed pixels
- **Flexible images:** `max-width: 100%` baseline, with `srcset` and `sizes` for performance
- **Media queries:** apply conditional styles only where the layout needs change
- **Content-first:** let content determine breakpoints, not device dimensions
- **Mobile-first:** write base styles for mobile, then add complexity for larger screens

Content Structure and Hierarchy

- Model content structure, not individual pages
- Maintain meaning hierarchy across all breakpoints
- Use progressive disclosure: accordions, tabs, 'show more'
- Reorder elements with CSS Grid order or template areas
- Choreograph content flow without altering semantics



Layout Patterns and Component Behavior

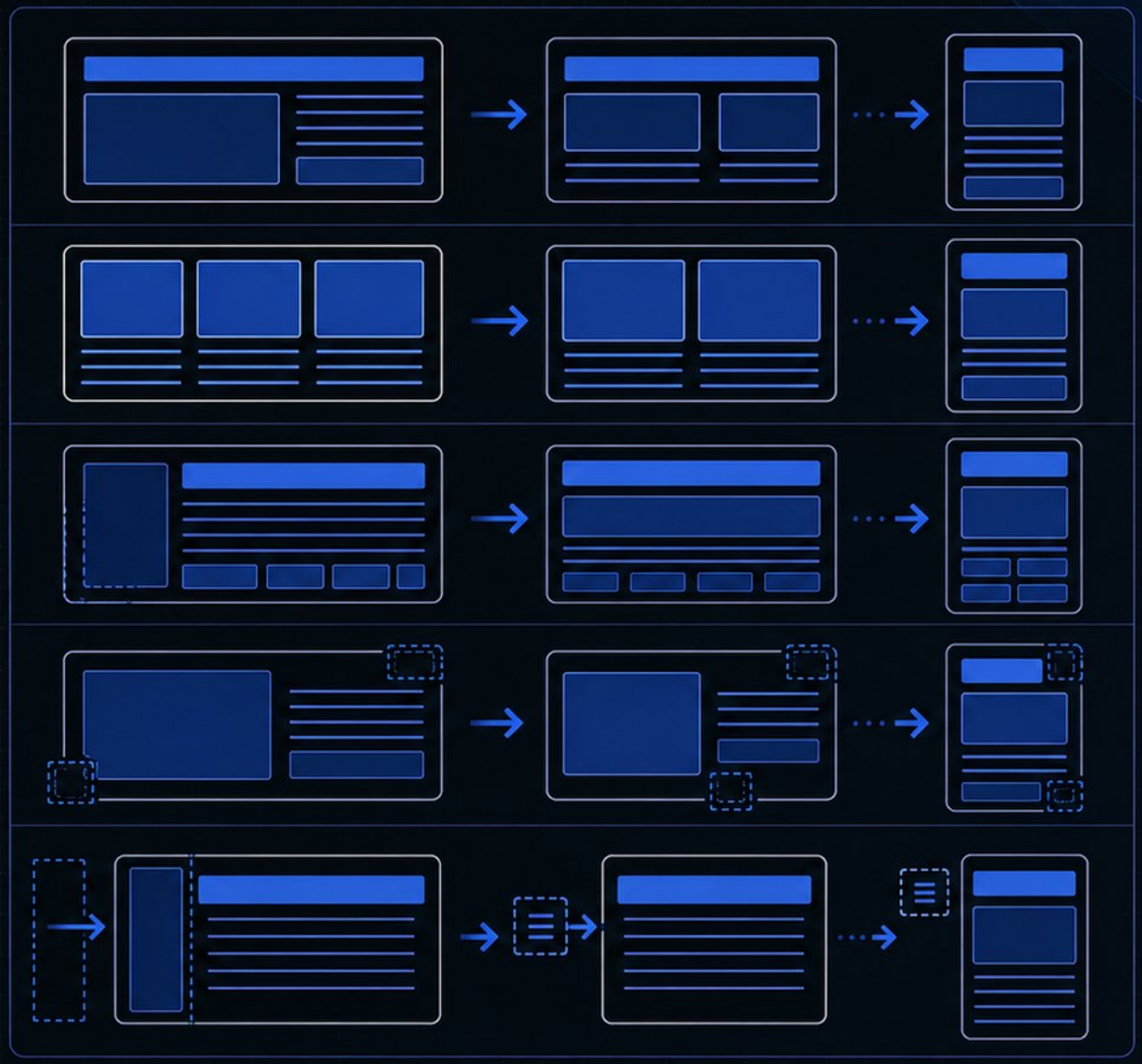
Mostly Fluid, Column Drop,
Layout Shifter, Tiny Tweaks, Off Canvas

Match pattern to context:
marketing, dashboards, or long-form articles

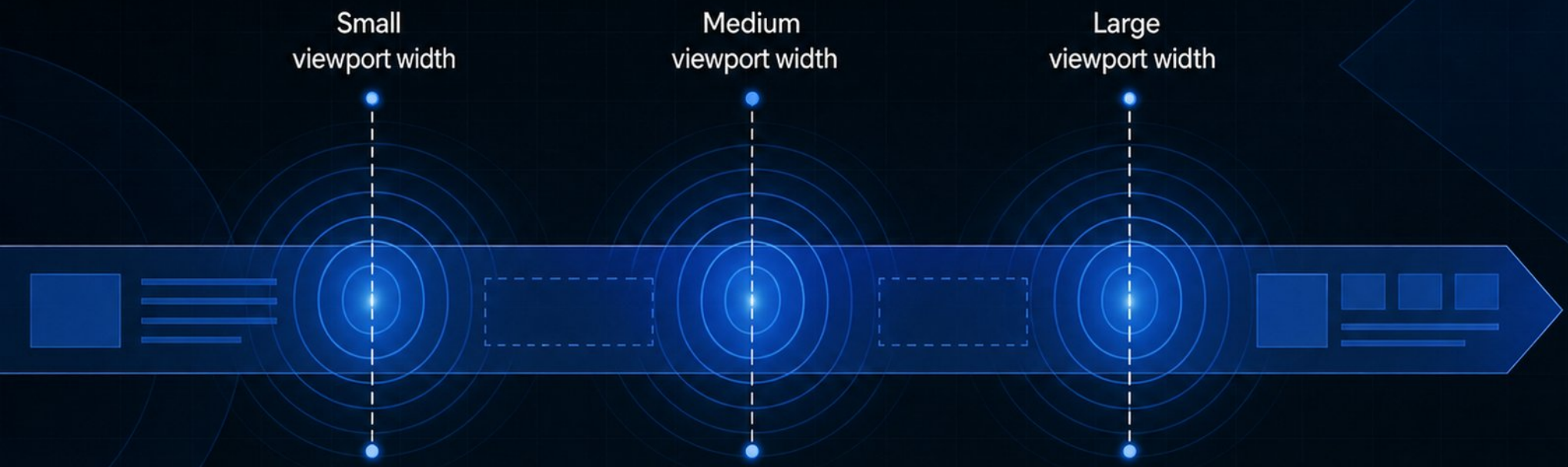
Define per-breakpoint rules:
reflow, resize, hide, or restructure components

Container-driven responsiveness:
adapt to parent width, not viewport
(2026 standard)

See it in action: Stripe, GitHub,
and Smashing Magazine



Breakpoints and Design Decisions



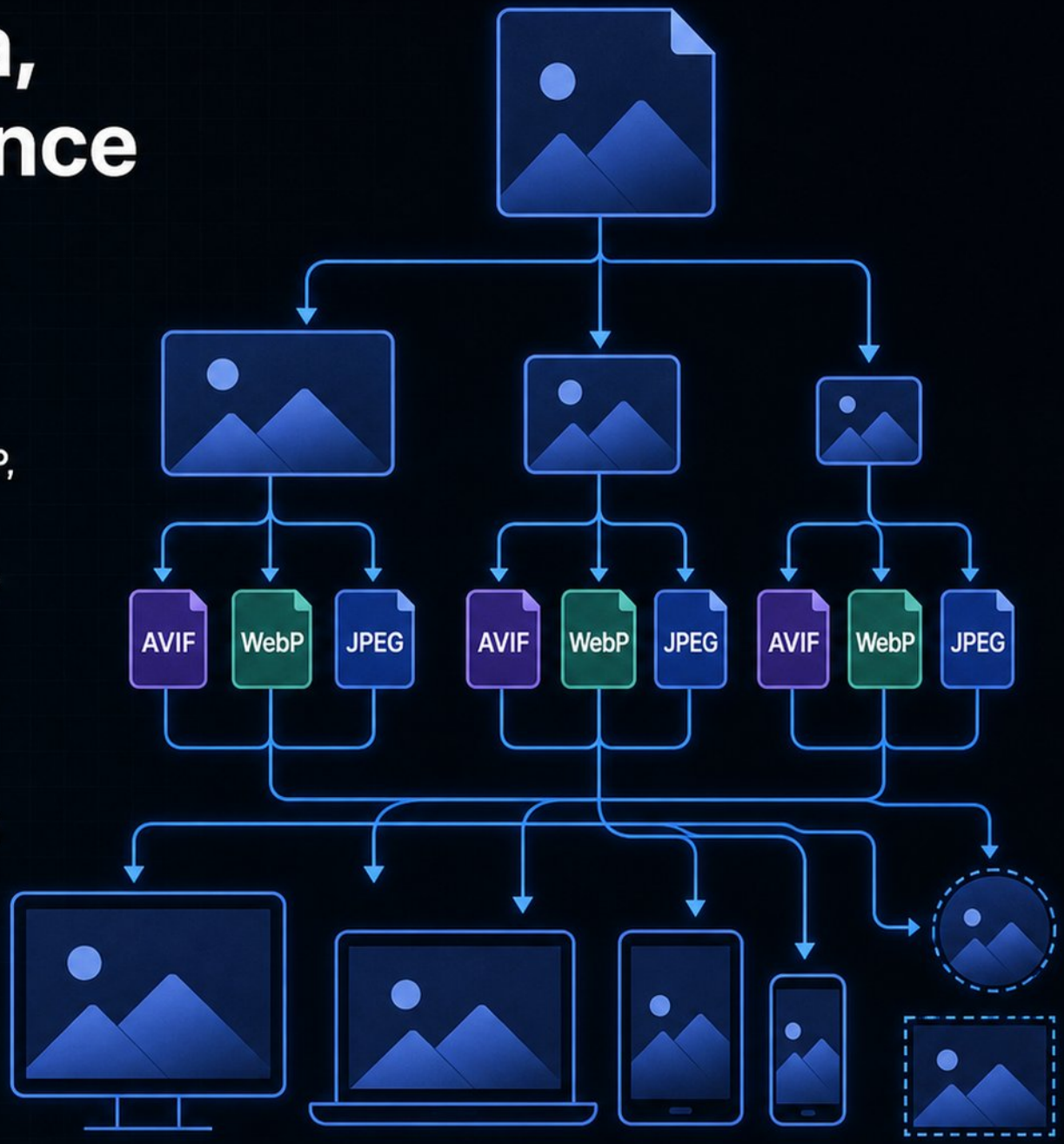
- Choose breakpoints where content breaks, not for device sizes
- Major breakpoints restructure pages; micro tweaks refine components
- Use @media for page layout; @container for component adaptation
- Container queries are fully supported across all modern browsers

Typography and Readability Across Sizes

- Fluid typography with clamp(): smooth scaling without media queries
- Optimal line length: 45–75 characters per line
- Headings: larger, bolder on desktop; tighter leading on mobile
- Fluid spacing: padding, margins, and gaps that scale with viewport

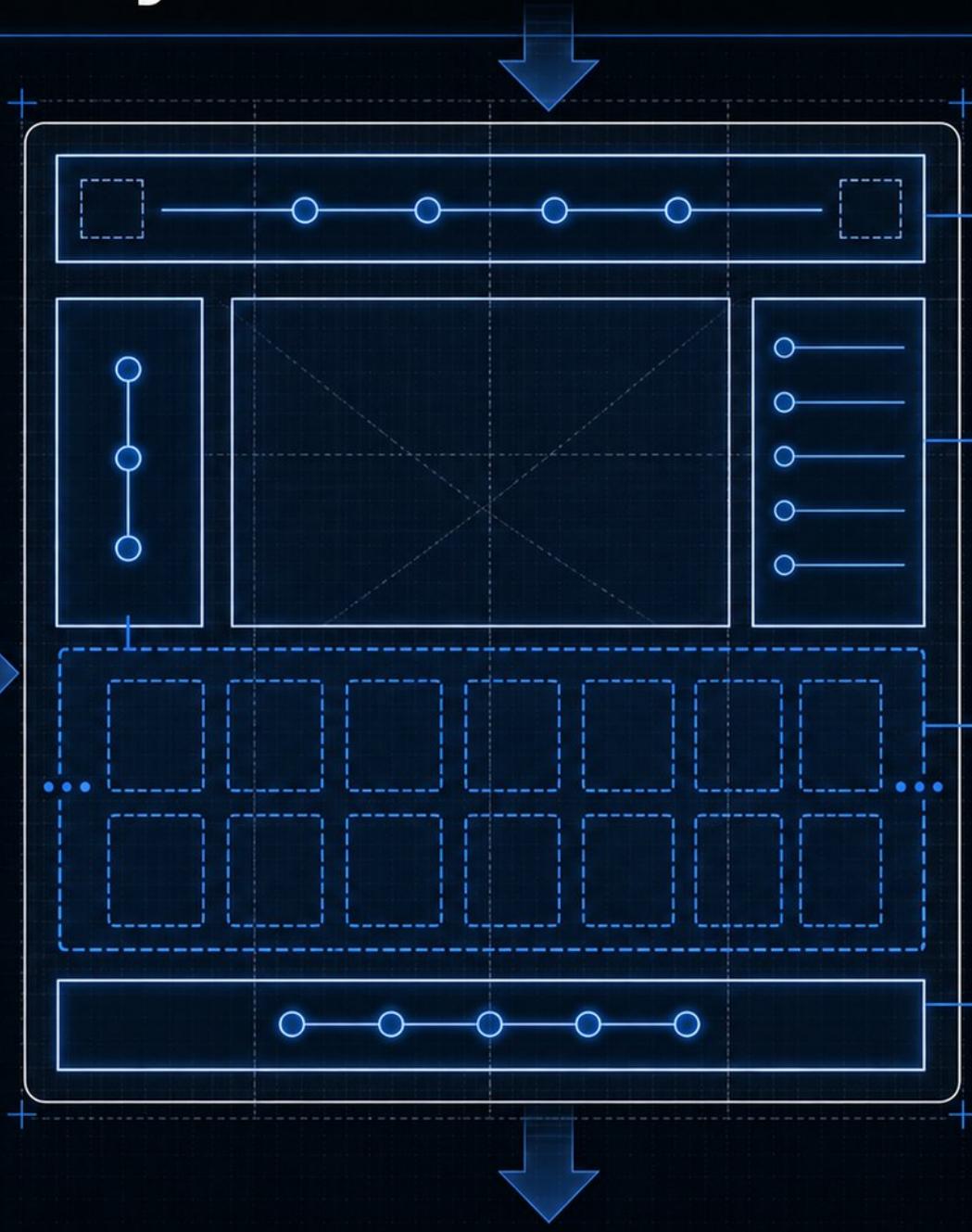
Images, Media, and Performance

- Use srcset and sizes for resolution switching; picture element for art direction
- SVG for icons and logos; AVIF, WebP, JPEG for photos
- AVIF primary (96%+ support), WebP fallback, JPEG baseline
- Correctly sized images reduce page weight by 40–60%
- Lazy load off-screen images; preload LCP image with fetchpriority



Modern CSS Layout: Grid and Flexbox

- CSS Grid: 2D page-level layouts, template areas, auto-fit/minmax for media-query-free grids
- Flexbox: 1D component alignment, wrapping navigation, flexible card interiors
- Combine Grid (outer shell) + Flexbox (internal alignment) for robust layouts
- auto-fit/minmax pattern: responsive columns with zero media queries
- min(100%, Xpx) trick: prevents grid overflow in narrow containers

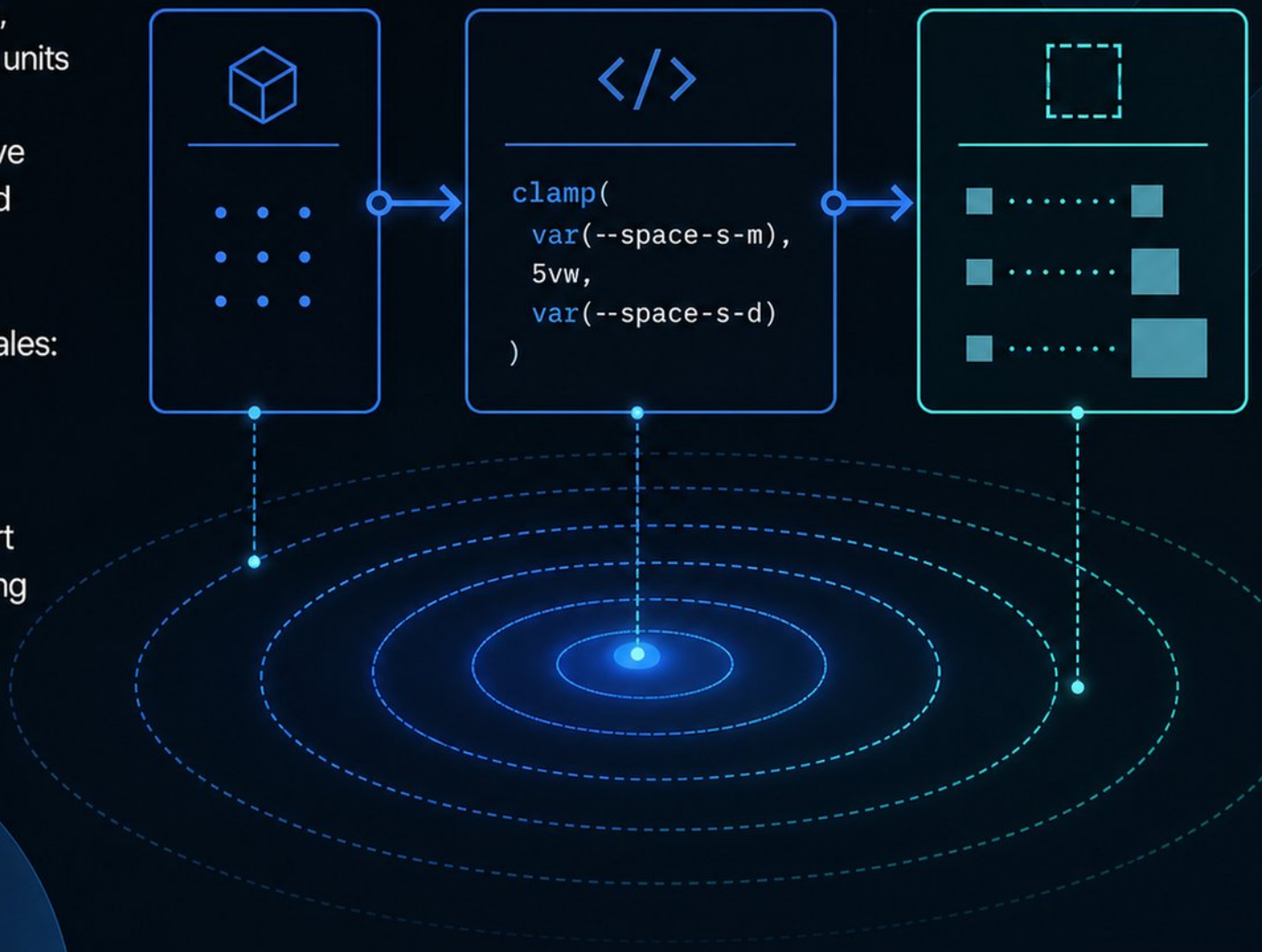


Container Queries: Component-Level Responsiveness

- Shift from viewport-based to parent-based responsive design
- Wrap with container-type, style with @container rules
- Use container query units: cqw, cqi, cqb, cqmin, cqmax
- Style queries respond to container custom properties
- Page shell uses media queries; components use container queries

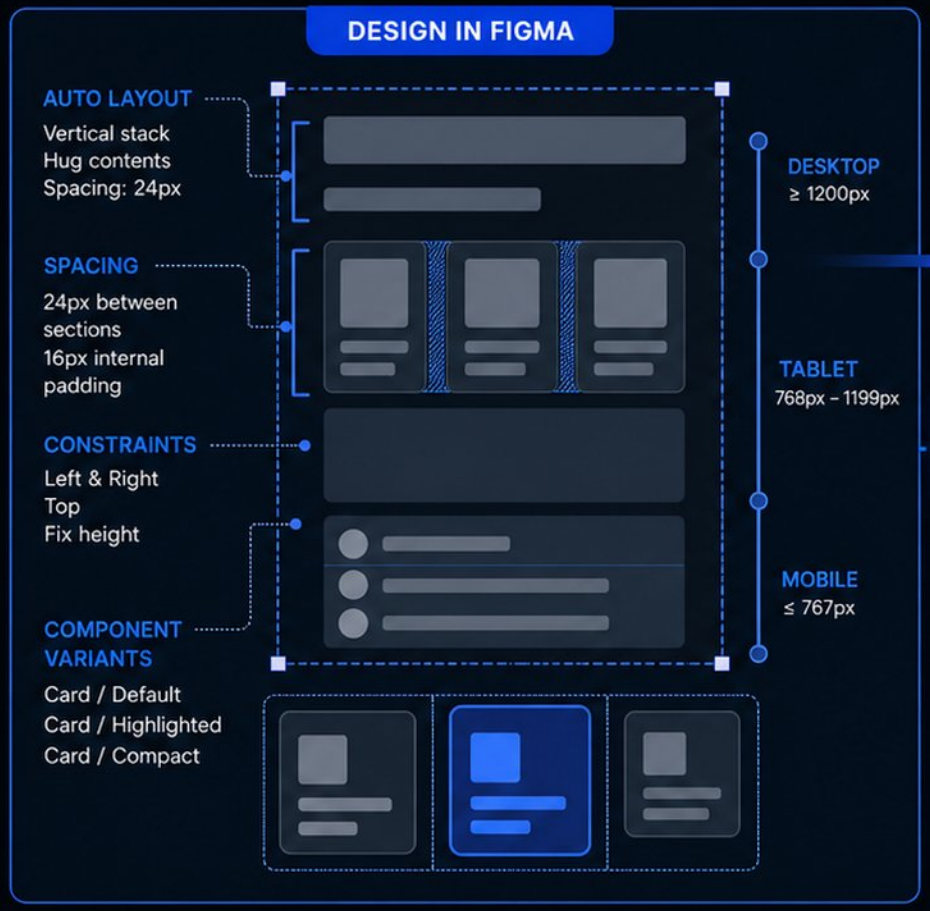
Responsive Spacing and Design Tokens

- Spacing that scales: clamp(), viewport units, and container units
- Design tokens for responsive behavior: type, spacing, and breakpoint scales
- Token-based responsive scales: one token, mobile base + desktop step-up
- Theming and tokens: support light/dark modes and branding at any size



Design Handoff and Developer Collaboration

DESIGN IN FIGMA



DESIGN HANDOFF

CODE (CSS)

```css
/* Design tokens from Figma */
:root {
  --color-bg: #0F1116;
  --color-surface: #161B22;
  --color-text: #E6ECF1;
  --space-xxs: 4px;
  --space-xs:  8px;
  --space-sm:  16px;
  --space-md:  24px;
  --space-lg:  32px;
  --radius-md:  12px;
  --radius-lg:  16px;
}

/* Base styles */
.stack {
  display: flex;
  flex-direction: column;
  gap: var(--space-md);
  align-items: stretch;
}

.grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: var(--space-md);
}

/* Responsive rules */
@media (max-width: 1199px) {
  .grid { grid-template-columns: repeat(2, 1fr); }
}
@media (max-width: 767px) {
  .grid { grid-template-columns: 1fr; }
}
```

TOKENS
Figma tokens mapped to CSS custom properties

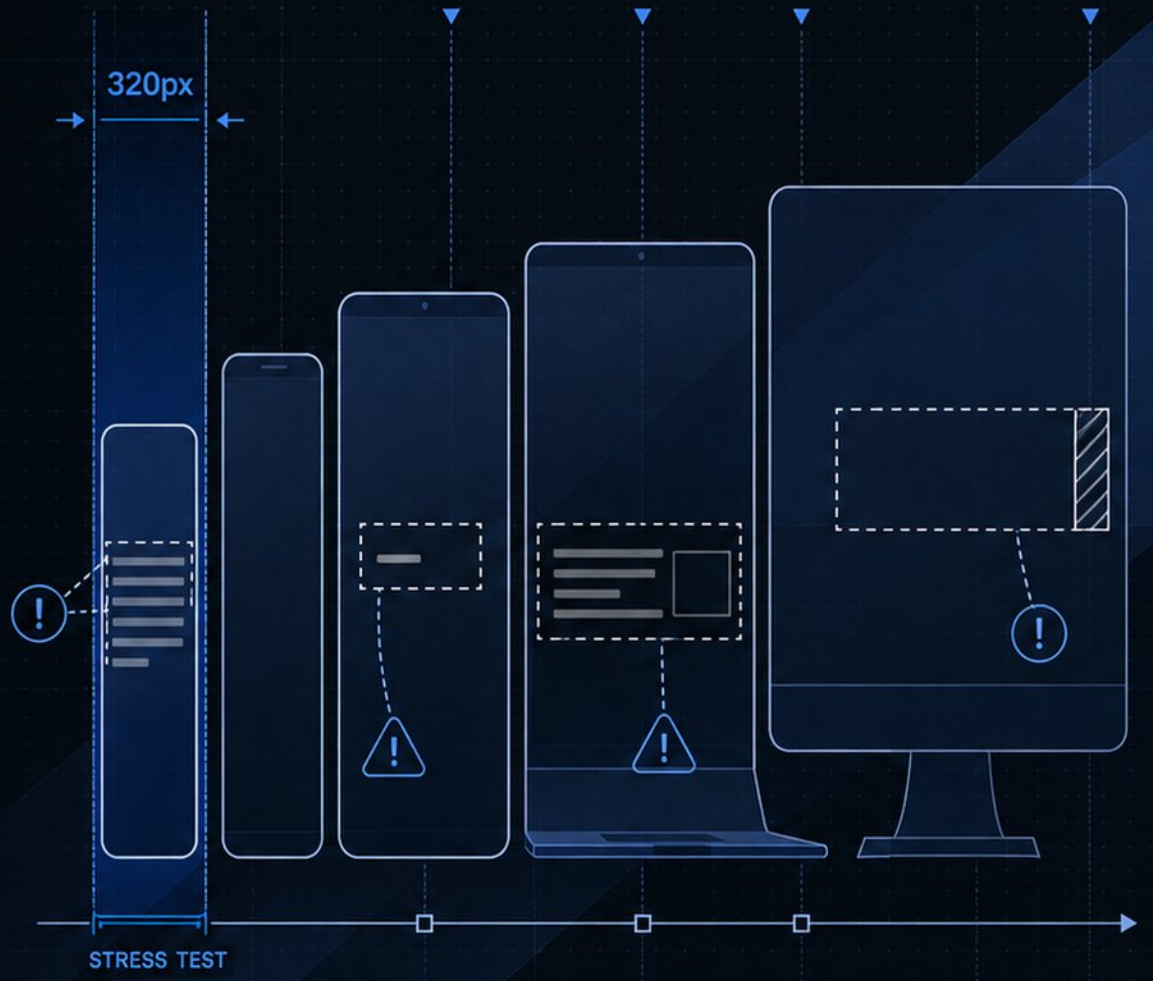
BASE STYLES
Layout, spacing, and components use tokens

RESPONSIVE RULES
Breakpoints match Figma documentation

- Document component reflow at each breakpoint, not just static mockups
- Annotate column counts, grid behavior, spacing, and component variants
- Use Figma auto layout, constraints, and annotation plugins to reduce ambiguity
- Address friction: missing breakpoint docs, ambiguous spacing, unstated rules
- Sync responsive rules from Figma tokens to CSS custom properties to production

Testing and Iterating Responsive Layouts

- Test on real devices, DevTools responsive mode, and services like BrowserStack
- Watch for overflow, orphaned text, broken grids, and unreadable line lengths
- Always test at 320px to catch early breakage
- Refine: start mobile-first, add breakpoints where content breaks, test intermediate widths
- Core Web Vitals (LCP, CLS, INP) depend on responsive image and layout choices



Responsive Design Systems and Scaling

- Embed responsive rules into reusable components that behave consistently everywhere
- Define spacing, type, and breakpoint tokens that all components inherit
- Create container-aware cards adapting to sidebar, grid, and hero contexts
- Govern with documentation, automated tests, and clear design handoff workflows
- Prepare for AI-assisted responsive design and adaptive, context-aware UX

Putting It All Together: A Responsive Design Workflow

- Start with content structure — model the content, not the device
- Apply fluid grids and flexible media using Grid and Flexbox
- Add fluid typography and spacing with clamp() and container units
- Introduce breakpoints only where the layout needs a different structure
- Use container queries for component-level adaptability in any slot

Key Takeaways and Next Steps

PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/f9f95656/public