Code Explanation: app.js
Generated: 2026-02-06 01:35
Block 1 (lines 1-1)
L1: ?const desktop = document.getElementById('desktop');
Explanation: JavaScript logic block.
Block 2 (lines 2-2)
L2: const windows = Array.from(document.querySelectorAll('.window'));
Explanation: JavaScript logic block.
Block 3 (lines 3-3)
L3: const dockButtons = Array.from(document.querySelectorAll('.dock-icon'));
Explanation: JavaScript logic block.
Block 4 (lines 4-4)
L4: const desktopIcons = Array.from(document.querySelectorAll('.desktop-icon'));
Explanation: JavaScript logic block.
Block 5 (lines 5-5)
L5: const menuToggle = document.getElementById('menu-toggle');
Explanation: Top menu dropdown logic (open/close, escape, outside click).
Block 6 (lines 6-6)
L6: const menuDropdown = document.getElementById('menu-dropdown');
Explanation: Top menu dropdown logic (open/close, escape, outside click).
Block 7 (lines 7-7)
L7: const menuRoot = document.querySelector('.menu-root');
Explanation: JavaScript logic block.
Block 8 (lines 8-8)
L8: const menuOptions = Array.from(document.querySelectorAll('.menu-option'));
Explanation: Menu actions that open About, Settings, or Portfolio windows.
Block 9 (lines 9-9)
L9: const appearanceOptions = Array.from(document.querySelectorAll('.appearance-option'));
Explanation: JavaScript logic block.
Block 10 (lines 10-10)
L10: const backgroundOptions = Array.from(document.querySelectorAll('.bg-option'));
Explanation: Desktop background switching from Settings tiles.
Block 11 (lines 12-12)
L12: let zIndexCounter = 10;
Explanation: JavaScript logic block.
Block 12 (lines 14-16)
L14: if (desktop) {
L15:   desktop.dataset.bg = desktop.dataset.bg || 'beach';
L16: }
Explanation: JavaScript logic block.
Block 13 (lines 18-21)
L18: const bringToFront = (win) => {
L19:   zIndexCounter += 1;
L20:   win.style.zIndex = zIndexCounter;
L21: };
Explanation: Function to raise a window above others (z-index management).
Block 14 (lines 23-29)
L23: const setDockActive = (id, isActive) => {
L24:   dockButtons.forEach((button) => {
L25:   if (button.dataset.window === id) {
L26:   button.classList.toggle('active', isActive);
L27:   }
L28:   });
L29: };
Explanation: JavaScript logic block.
Block 15 (lines 31-41)
L31: const openWindow = (id) => {
L32:   const win = document.getElementById(id);
L33:   if (!win) {
L34:   return;

```
  }
  win.classList.add('is-open');
  win.classList.remove('is-minimized');
  win.setAttribute('aria-hidden', 'false');
  bringToFront(win);
  setDockActive(id, true);
};
```
Explanation: Function to raise a window above others (z-index management).
Block 16 (lines 43-47)
```
const closeWindow = (win) => {
  win.classList.remove('is-open', 'is-minimized', 'is-maximized');
  win.setAttribute('aria-hidden', 'true');
  setDockActive(win.id, false);
};
```
Explanation: JavaScript logic block.
Block 17 (lines 49-52)
```
const minimizeWindow = (win) => {
  win.classList.add('is-minimized');
  setDockActive(win.id, true);
};
```
Explanation: JavaScript logic block.
Block 18 (lines 54-56)
```
const toggleMaximize = (win) => {
  win.classList.toggle('is-maximized');
};
```
Explanation: JavaScript logic block.
Block 19 (lines 58-80)
```
windows.forEach((win) => {
  win.addEventListener('mousedown', () => bringToFront(win));
  const titlebar = win.querySelector('.window-titlebar');
  if (titlebar) {
    titlebar.addEventListener('mousedown', (event) => startDrag(event, win));
  }
  win.querySelectorAll('[data-action]').forEach((button) => {
    button.addEventListener('click', (event) => {
      event.stopPropagation();
      const action = button.dataset.action;
      if (action === 'close') {
        closeWindow(win);
      }
      if (action === 'minimize') {
        minimizeWindow(win);
      }
      if (action === 'maximize') {
        toggleMaximize(win);
      }
    });
  });
});
```
Explanation: Function to raise a window above others (z-index management).
Block 20 (lines 82-84)
```
desktopIcons.forEach((icon) => {
  icon.addEventListener('dblclick', () => openWindow(icon.dataset.window));
});
```
Explanation: Double-click desktop icons to open corresponding windows.
Block 21 (lines 86-101)
```
dockButtons.forEach((button) => {
  button.addEventListener('click', () => {
    const id = button.dataset.window;
    const win = document.getElementById(id);
```

```
L90:     if (!win) {
L91:       return;
L92:     }
L93:     const isOpen = win.classList.contains('is-open');
L94:     const isMinimized = win.classList.contains('is-minimized');
L95:     if (isOpen && !isMinimized) {
L96:       minimizeWindow(win);
L97:       return;
L98:     }
L99:     openWindow(id);
L100:   });
L101: });
```

Explanation: Dock click behavior to open or minimize windows.

Block 22 (lines 103-109)

```
L103: const closeMenu = () => {
L104:   if (!menuDropdown || !menuToggle) {
L105:     return;
L106:   }
L107:   menuDropdown.classList.remove('is-open');
L108:   menuToggle.setAttribute('aria-expanded', 'false');
L109: };
```

Explanation: Top menu dropdown logic (open/close, escape, outside click).

Block 23 (lines 111-129)

```
L111: if (menuToggle && menuDropdown) {
L112:   menuToggle.addEventListener('click', (event) => {
L113:     event.stopPropagation();
L114:     const isOpen = menuDropdown.classList.toggle('is-open');
L115:     menuToggle.setAttribute('aria-expanded', String(isOpen));
L116:   });
L118:   document.addEventListener('click', (event) => {
L119:     if (menuRoot && !menuRoot.contains(event.target)) {
L120:       closeMenu();
L121:     }
L122:   });
L124:   document.addEventListener('keydown', (event) => {
L125:     if (event.key === 'Escape') {
L126:       closeMenu();
L127:     }
L128:   });
L129: }
```

Explanation: Top menu dropdown logic (open/close, escape, outside click).

Block 24 (lines 131-145)

```
L131: menuOptions.forEach((option) => {
L132:   option.addEventListener('click', () => {
L133:     const action = option.dataset.menuAction;
L134:     if (action === 'settings') {
L135:       openWindow('window-settings');
L136:     }
L137:     if (action === 'about') {
L138:       openWindow('window-about');
L139:     }
L140:     if (action === 'portfolio') {
L141:       openWindow('window-computer');
L142:     }
L143:     closeMenu();
L144:   });
L145: });
```

Explanation: Menu actions that open About, Settings, or Portfolio windows.

Block 25 (lines 147-147)

```
L147: let dragState = null;
```

Explanation: JavaScript logic block.
Block 26 (lines 149-169)
```
L149: const startDrag = (event, win) => {
L150:   if (event.button !== 0) {
L151:     return;
L152:   }
L153:   if (win.classList.contains('is-maximized')) {
L154:     return;
L155:   }
L156:   if (event.target.closest('.window-controls')) {
L157:     return;
L158:   }
L159:   event.preventDefault();
L160:   const deskRect = desktop.getBoundingClientRect();
L161:   const winRect = win.getBoundingClientRect();
L162:   dragState = {
L163:     win,
L164:     deskRect,
L165:     offsetX: event.clientX - winRect.left,
L166:     offsetY: event.clientY - winRect.top
L167:   };
L168:   bringToFront(win);
L169: };
```
Explanation: Function to raise a window above others (z-index management).
Block 27 (lines 171-184)
```
L171: const dragMove = (event) => {
L172:   if (!dragState) {
L173:     return;
L174:   }
L175:   const { win, deskRect, offsetX, offsetY } = dragState;
L176:   const maxX = deskRect.width - win.offsetWidth - 10;
L177:   const maxY = deskRect.height - win.offsetHeight - 10;
L178:   let nextX = event.clientX - deskRect.left - offsetX;
L179:   let nextY = event.clientY - deskRect.top - offsetY;
L180:   nextX = Math.max(10, Math.min(nextX, maxX));
L181:   nextY = Math.max(60, Math.min(nextY, maxY));
L182:   win.style.left = `${nextX}px`;
L183:   win.style.top = `${nextY}px`;
L184: };
```
Explanation: Window drag-and-drop behavior with boundary constraints.
Block 28 (lines 186-188)
```
L186: const stopDrag = () => {
L187:   dragState = null;
L188: };
```
Explanation: JavaScript logic block.
Block 29 (lines 190-190)
```
L190: window.addEventListener('mousemove', dragMove);
```
Explanation: Window drag-and-drop behavior with boundary constraints.
Block 30 (lines 191-191)
```
L191: window.addEventListener('mouseup', stopDrag);
```
Explanation: JavaScript logic block.
Block 31 (lines 193-193)
```
L193: const clockEl = document.getElementById('clock');
```
Explanation: JavaScript logic block.
Block 32 (lines 195-205)
```
L195: const updateClock = () => {
L196:   const now = new Date();
L197:   const weekday = now.toLocaleString('en-US', { weekday: 'short' }).toUpperCase();
L198:   const month = now.toLocaleString('en-US', { month: 'short' }).toUpperCase();
L199:   const day = now.getDate();
```

```
L200:   let hours = now.getHours();
L201:   const minutes = now.getMinutes().toString().padStart(2, '0');
L202:   const ampm = hours >= 12 ? 'PM' : 'AM';
L203:   hours = hours % 12 || 12;
L204:   clockEl.textContent = `${weekday} ${month} ${day} ${hours}:${minutes} ${ampm}`;
L205: };
```

Explanation: Live clock formatting and update interval.

Block 33 (lines 207-207)

```
L207: updateClock();
```

Explanation: Live clock formatting and update interval.

Block 34 (lines 208-208)

```
L208: setInterval(updateClock, 60000);
```

Explanation: Live clock formatting and update interval.

Block 35 (lines 210-210)

```
L210: const computerBreadcrumb = document.getElementById('computer-breadcrumb');
```

Explanation: JavaScript logic block.

Block 36 (lines 211-211)

```
L211: const computerList = document.getElementById('computer-list');
```

Explanation: JavaScript logic block.

Block 37 (lines 212-212)

```
L212: const computerPreview = document.getElementById('computer-preview');
```

Explanation: JavaScript logic block.

Block 38 (lines 213-213)

```
L213: const computerBack = document.getElementById('computer-back');
```

Explanation: JavaScript logic block.

Block 39 (lines 214-214)

```
L214: const sidebarButtons = Array.from(document.querySelectorAll('.sidebar-item'));
```

Explanation: JavaScript logic block.

Block 40 (lines 216-336)

```
L216: const computerViews = {
L217:   portfolio: {
L218:     label: 'Portfolio',
L219:     items: [
L220:       { title: 'Projects', openView: 'projects' },
L221:       { title: 'Lab', openView: 'lab' },
L222:       { title: 'Resources', openView: 'resources' },
L223:       { title: 'About', openView: 'about' },
L224:       { title: 'Contact', openView: 'contact' },
L225:       { title: 'Work Experience', openView: 'experience' }
L226:     ]
L227:   },
L228:   projects: {
L229:     label: 'Projects',
L230:     items: [
L231:       {
L232:         title: 'Sibarita',
L233:         subtitle: 'Brand identity system',
L234:         description: 'Full brand system with packaging, tone of voice, and digital
      rollout. Retro meets modern with playful color and bold typography.',
L235:         tags: ['Branding', 'Packaging', 'Digital']
L236:       },
L237:       {
L238:         title: 'Neon Dunes',
L239:         subtitle: 'UX refresh',
L240:         description: 'UX polish for a travel platform focused on clarity, search, and
      moments of delight across mobile and desktop.',
L241:         tags: ['UX', 'UI', 'Research']
L242:       },
L243:       {
L244:         title: 'Bytewave',
```

```
        subtitle: 'Web experience',
        description: 'Landing page design and animation system for a Web3 music label with a synthwave aesthetic.',
        tags: ['Web', 'Motion', 'Brand']
      },
      {
        title: 'Studio Vela',
        subtitle: 'Portfolio site',
        description: 'Custom portfolio with modular case studies, lightweight CMS, and playful transitions.',
        tags: ['Web', 'System', 'CMS']
      },
      {
        title: 'Horizon',
        subtitle: 'Product design',
        description: 'Reimagined onboarding and dashboards for a productivity app used by remote teams.',
        tags: ['Product', 'UI', 'Strategy']
      }
    ]
  },
  lab: {
    label: 'Lab',
    items: [
      {
        title: 'Pixel Map',
        subtitle: 'Experimental prototype',
        description: 'A playful map interface that reacts to sound and scroll to reveal hidden layers.',
        tags: ['Prototype', 'Interaction']
      },
      {
        title: 'Arcade Nav',
        subtitle: 'Menu concept',
        description: 'Navigation UI inspired by classic arcade cabinets with neon states and chime sounds.',
        tags: ['Concept', 'UI']
      }
    ]
  },
  resources: {
    label: 'Resources',
    items: [
      {
        title: 'Design System',
        subtitle: 'Components',
        description: 'Tokens, grids, iconography, and type ramp for fast, consistent UI work.',
        tags: ['System', 'UI']
      },
      {
        title: 'Case Study Kit',
        subtitle: 'Templates',
        description: 'Case study layouts, storytelling prompts, and motion templates for portfolio updates.',
        tags: ['Docs', 'Templates']
      }
    ]
  },
  about: {
```

```
L298:     label: 'About',
L299:     items: [
L300:       {
L301:         title: 'Profile',
L302:         subtitle: 'Creative director',
L303:         description: 'I design bold digital experiences, combining nostalgic references
      with modern usability.',
L304:         tags: ['Bio', 'Vision']
L305:       }
L306:     ]
L307:   },
L308:   contact: {
L309:     label: 'Contact',
L310:     items: [
L311:       {
L312:         title: 'Email',
L313:         subtitle: 'hello@example.com',
L314:         description: 'Swap this with your real email, social links, and location
      details.',
L315:         tags: ['Email', 'Social']
L316:       }
L317:     ]
L318:   },
L319:   experience: {
L320:     label: 'Work Experience',
L321:     items: [
L322:       {
L323:         title: 'Lead Designer',
L324:         subtitle: 'Sunset Lab (2021 - Now)',
L325:         description: 'Led a small team delivering identity systems and playful product
      experiences.',
L326:         tags: ['Leadership', 'Brand']
L327:       },
L328:       {
L329:         title: 'UX Designer',
L330:         subtitle: 'Aether Co (2017 - 2021)',
L331:         description: 'Owned the design system and collaborated across product,
      engineering, and research.',
L332:         tags: ['Product', 'Systems']
L333:       }
L334:     ]
L335:   }
L336: };
```

Explanation: Data model for My Computer sections and file preview content.

Block 41 (lines 338-338)

```
L338: let viewStack = ['portfolio'];
```

Explanation: JavaScript logic block.

Block 42 (lines 340-377)

```
L340: const renderComputerView = () => {
L341:   const viewId = viewStack[viewStack.length - 1];
L342:   const view = computerViews[viewId];
L343:   if (!view) {
L344:     return;
L345:   }
L347:   computerBreadcrumb.textContent = viewStack
L348:     .map((id) => computerViews[id].label.toUpperCase())
L349:     .join(' / ');
L351:   computerBack.disabled = viewStack.length <= 1;
L353:   computerList.innerHTML = '';
L354:   computerPreview.innerHTML = '<div class="empty-state">SELECT A FILE TO PREVIEW</div>';
```

```
  view.items.forEach((item) => {
    const button = document.createElement('button');
    button.type = 'button';
    button.className = 'file-item';
    button.innerHTML = `<svg class="icon small"><use href="#icon-folder"></use></svg><span>${item.title}</span>`;
    button.addEventListener('click', () => {
      const currentItems = Array.from(computerList.querySelectorAll('.file-item'));
      currentItems.forEach((node) => node.classList.remove('active'));
      button.classList.add('active');
      if (item.openView) {
        viewStack.push(item.openView);
        renderComputerView();
        return;
      }
      showComputerPreview(viewId, item);
    });
    computerList.appendChild(button);
  });
};
```

Explanation: Data model for My Computer sections and file preview content.

Block 43 (lines 379-396)

```
const showComputerPreview = (viewId, item) => {
  const tags = item.tags
    ? `<div class="tags">${item.tags
        .map((tag) => `<span class="tag">${tag.toUpperCase()}</span>`)
        .join('')}</div>`
    : '';
  computerPreview.innerHTML = `
    <h3 class="pixel-title">${item.title.toUpperCase()}</h3>
    <p class="muted">${item.subtitle || ''}</p>
    <p>${item.description || ''}</p>
    ${tags}
  `;
  computerBreadcrumb.textContent = `${viewStack
    .map((id) => computerViews[id].label.toUpperCase())
    .join(' / ')} / ${item.title.toUpperCase()}`;
};
```

Explanation: Data model for My Computer sections and file preview content.

Block 44 (lines 398-409)

```
sidebarButtons.forEach((button) => {
  button.addEventListener('click', () => {
    const target = button.dataset.section;
    if (!computerViews[target]) {
      return;
    }
    sidebarButtons.forEach((node) => node.classList.remove('active'));
    button.classList.add('active');
    viewStack = [target];
    renderComputerView();
  });
});
```

Explanation: Data model for My Computer sections and file preview content.

Block 45 (lines 411-416)

```
computerBack.addEventListener('click', () => {
  if (viewStack.length > 1) {
    viewStack.pop();
    renderComputerView();
  }
});
```

Explanation: Renders file lists, breadcrumbs, and preview content in My Computer.
Block 46 (lines 418-418)
L418: renderComputerView();
Explanation: Renders file lists, breadcrumbs, and preview content in My Computer.
Block 47 (lines 420-425)
L420: const applyTheme = (theme) => {
L421:   if (!document.body) {
L422:   return;
L423:   }
L424:   document.body.dataset.theme = theme;
L425: };
Explanation: JavaScript logic block.
Block 48 (lines 427-427)
L427: const initialThemeButton = appearanceOptions.find((button) =>
    button.classList.contains('active'));
Explanation: JavaScript logic block.
Block 49 (lines 428-428)
L428: applyTheme((initialThemeButton && initialThemeButton.dataset.theme) || 'light');
Explanation: JavaScript logic block.
Block 50 (lines 430-436)
L430: appearanceOptions.forEach((button) => {
L431:   button.addEventListener('click', () => {
L432:   appearanceOptions.forEach((item) => item.classList.remove('active'));
L433:   button.classList.add('active');
L434:   applyTheme(button.dataset.theme || 'light');
L435:   });
L436: });
Explanation: Theme toggling (light/dark) from Settings.
Block 51 (lines 438-446)
L438: backgroundOptions.forEach((button) => {
L439:   button.addEventListener('click', () => {
L440:   backgroundOptions.forEach((item) => item.classList.remove('active'));
L441:   button.classList.add('active');
L442:   if (desktop) {
L443:   desktop.dataset.bg = button.dataset.bg || 'beach';
L444:   }
L445:   });
L446: });
Explanation: Desktop background switching from Settings tiles.
Block 52 (lines 448-448)
L448: const paintCanvas = document.getElementById('paint-canvas');
Explanation: Paint app drawing logic, including resize and clearing.
Block 53 (lines 449-449)
L449: const paintColor = document.getElementById('paint-color');
Explanation: JavaScript logic block.
Block 54 (lines 450-450)
L450: const paintSize = document.getElementById('paint-size');
Explanation: JavaScript logic block.
Block 55 (lines 451-451)
L451: const paintClear = document.getElementById('paint-clear');
Explanation: JavaScript logic block.
Block 56 (lines 453-503)
L453: if (paintCanvas) {
L454:   const ctx = paintCanvas.getContext('2d');
L455:   let drawing = false;
L457:   const resizeCanvas = () => {
L458:   const rect = paintCanvas.getBoundingClientRect();
L459:   const scale = window.devicePixelRatio || 1;
L460:   paintCanvas.width = rect.width * scale;
L461:   paintCanvas.height = rect.height * scale;

```
L462:     ctx.setTransform(1, 0, 0, 1, 0, 0);
L463:     ctx.scale(scale, scale);
L464:     ctx.fillStyle = '#ffffff';
L465:     ctx.fillRect(0, 0, rect.width, rect.height);
L466:   };
L468:   resizeCanvas();
L469:   window.addEventListener('resize', resizeCanvas);
L471:   const startDraw = (event) => {
L472:     drawing = true;
L473:     ctx.beginPath();
L474:     ctx.moveTo(event.offsetX, event.offsetY);
L475:   };
L477:   const draw = (event) => {
L478:     if (!drawing) {
L479:       return;
L480:     }
L481:     ctx.lineTo(event.offsetX, event.offsetY);
L482:     ctx.strokeStyle = paintColor.value;
L483:     ctx.lineWidth = Number(paintSize.value);
L484:     ctx.lineCap = 'round';
L485:     ctx.lineJoin = 'round';
L486:     ctx.stroke();
L487:   };
L489:   const endDraw = () => {
L490:     drawing = false;
L491:     ctx.closePath();
L492:   };
L494:   paintCanvas.addEventListener('mousedown', startDraw);
L495:   paintCanvas.addEventListener('mousemove', draw);
L496:   paintCanvas.addEventListener('mouseup', endDraw);
L497:   paintCanvas.addEventListener('mouseleave', endDraw);
L499:   paintClear.addEventListener('click', () => {
L500:     ctx.clearRect(0, 0, paintCanvas.width, paintCanvas.height);
L501:     resizeCanvas();
L502:   });
L503: }
```

Explanation: Paint app drawing logic, including resize and clearing.

Block 57 (lines 505-505)

```
L505: const musicTitle = document.getElementById('music-title');
```

Explanation: Music player behavior: station selection, play/pause, volume display.

Block 58 (lines 506-506)

```
L506: const musicSubtitle = document.getElementById('music-subtitle');
```

Explanation: JavaScript logic block.

Block 59 (lines 507-507)

```
L507: const musicPlay = document.getElementById('music-play');
```

Explanation: JavaScript logic block.

Block 60 (lines 508-508)

```
L508: const volumeSlider = document.getElementById('music-volume');
```

Explanation: JavaScript logic block.

Block 61 (lines 509-509)

```
L509: const volumePercent = document.getElementById('music-volume-percent');
```

Explanation: JavaScript logic block.

Block 62 (lines 510-510)

```
L510: const stationButtons = Array.from(document.querySelectorAll('.station-item'));
```

Explanation: Music player behavior: station selection, play/pause, volume display.

Block 63 (lines 512-512)

```
L512: let activeStation = stationButtons.find((button) => button.classList.contains('active'))
    || stationButtons[0];
```

Explanation: Music player behavior: station selection, play/pause, volume display.

Block 64 (lines 513-513)

L513: let isPlaying = true;
Explanation: JavaScript logic block.
Block 65 (lines 515-529)
L515: const updateMusicUI = () => {
L516:   if (activeStation && musicTitle && musicSubtitle) {
L517:   musicTitle.textContent = activeStation.dataset.track || 'UNKNOWN';
L518:   musicSubtitle.textContent = activeStation.dataset.genre || 'RADIO';
L519:   }
L520:   if (musicPlay) {
L521:   musicPlay.textContent = isPlaying ? '||' : '>';
L522:   }
L523:   stationButtons.forEach((button) => {
L524:   button.classList.toggle('is-playing', isPlaying && button === activeStation);
L525:   });
L526:   if (volumeSlider && volumePercent) {
L527:   volumePercent.textContent = `${volumeSlider.value}%`;
L528:   }
L529: };
Explanation: Music player behavior: station selection, play/pause, volume display.
Block 66 (lines 531-539)
L531: stationButtons.forEach((button) => {
L532:   button.addEventListener('click', () => {
L533:   stationButtons.forEach((item) => item.classList.remove('active'));
L534:   button.classList.add('active');
L535:   activeStation = button;
L536:   isPlaying = true;
L537:   updateMusicUI();
L538:   });
L539: });
Explanation: Music player behavior: station selection, play/pause, volume display.
Block 67 (lines 541-546)
L541: if (musicPlay) {
L542:   musicPlay.addEventListener('click', () => {
L543:   isPlaying = !isPlaying;
L544:   updateMusicUI();
L545:   });
L546: }
Explanation: JavaScript logic block.
Block 68 (lines 548-550)
L548: if (volumeSlider) {
L549:   volumeSlider.addEventListener('input', updateMusicUI);
L550: }
Explanation: JavaScript logic block.
Block 69 (lines 552-552)
L552: updateMusicUI();
Explanation: JavaScript logic block.
Block 70 (lines 554-554)
L554: openWindow('window-computer');
Explanation: Initializes the UI by opening the main window.