# BITBOXER - Bitbox Preset Editor

A web-based preset editor for the 1010music Bitbox Micro and mk2 samplers.

## 📁 File Structure

```
bitboxer/
├── index.html            # Main HTML structure
├── css/
│   └── styles.css        # All styling (organized with CSS variables)
├── js/
│   ├── lib/
│   │   └── fflate.js     # ZIP/GZIP library (3rd party)
│   ├── config.js         # Constants & configuration
│   ├── data-structures.js # Data models & empty presets
│   ├── utils.js          # Helper functions & formatters
│   ├── ui-controller.js  # UI state management & interactions
│   ├── xml-handler.js    # XML import/export logic
│   ├── pad-editor.js     # Pad editing & modulation
│   ├── fx-editor.js      # FX editing & modulation
│   └── main.js           # Application initialization
└── README.md             # This file
```

## 🔗 Module Dependencies & Load Order

**Critical:** JavaScript files must be loaded in this exact order:

```
1. fflate.js          (No dependencies - ZIP library)
2. config.js          (No dependencies - defines constants)
3. data-structures.js  (Uses: config.js)
4. utils.js           (Uses: config.js)
5. ui-controller.js    (Uses: config.js, utils.js)
6. xml-handler.js      (Uses: data-structures.js, utils.js, fflate.js)
7. pad-editor.js       (Uses: config.js, data-structures.js, ui-controller.js, utils.js)
8. fx-editor.js        (Uses: config.js, data-structures.js, ui-controller.js, utils.js)
9. main.js            (Uses: ALL above modules - initializes app)
```

## 📦 Module Responsibilities

### config.js

- **Purpose:** Central configuration and constants
- **Exports:**
  - `MOD_SOURCES` - Modulation source definitions

- `MOD_DESTINATIONS` - Modulation destinations by cell mode
- `FX_MOD_DESTINATIONS` - FX modulation destinations
- `CELL_MODE_NAMES` - Human-readable mode names
- UI constants (max slots, timeouts, etc.)

### data-structures.js

- **Purpose:** Data models and templates
- **Functions:**
  - `createEmptyPadData()` - Creates blank pad template
  - `createEmptyFXData()` - Creates blank FX template
  - `createEmptyPreset()` - Creates full preset structure
- **Data Models:**
  - Pad data structure
  - FX data structure
  - Preset data structure

### utils.js

- **Purpose:** Reusable helper functions
- **Functions:**
  - `formatParamValue()` - Formats parameter values for display
  - `parseDisplayValue()` - Parses user input back to internal values
  - `generateRandomHex()` - Creates random project names
  - `validateModDestination()` - Checks modulation slot limits
  - `cloneObject()` - Deep clones data structures
  - `setStatus()` - Updates status bar messages

### ui-controller.js

- **Purpose:** UI state management
- **Functions:**
  - `openModal()` / `closeModal()` - Modal window management
  - `switchTab()` - Tab navigation in modals
  - `updateButtonStates()` - Enable/disable buttons

- `showContextMenu()` / `hideContextMenu()` - Right-click menu
- `updateConditionalVisibility()` - Show/hide based on mode
- `setupEventListeners()` - Wire up UI events

### xml-handler.js

- **Purpose:** XML parsing and generation
- **Functions:**
  - `loadPreset()` - Parses XML into preset data
  - `savePreset()` - Generates XML from preset data
  - `generatePresetXML()` - Creates XML structure
  - `parseXMLCell()` - Parses individual cell data
  - `exportPads()` - Exports pads to JSON/ZIP

### pad-editor.js

- **Purpose:** Pad parameter editing and modulation
- **Functions:**
  - `openPadEditor()` - Opens pad edit modal
  - `loadPadParams()` - Loads pad data to UI
  - `savePadParams()` - Saves UI data to pad
  - `renderModSlots()` - Renders modulation UI
  - `addModSlot()` / `removeModSlot()` - Mod slot management
  - `updateModSlotAppearance()` - Visual feedback for active/inactive mods
  - `drawEnvelope()` - Canvas ADSR visualization

### fx-editor.js

- **Purpose:** FX parameter editing and modulation
- **Functions:**
  - `openFxEditor()` - Opens FX edit modal
  - `loadFxParams()` - Loads FX data to UI
  - `saveFxParams()` - Saves UI data to FX
  - `renderFxModSlots()` - Renders FX modulation UI
  - `addFxModSlot()` / `removeFxModSlot()` - FX mod management

- `updateFxConditionalVisibility()` - Show/hide based on FX settings

### main.js

- **Purpose:** Application initialization and coordination
- **Functions:**
  - `initializeApp()` - Main entry point
  - `createPadGrid()` - Generates pad grid UI
  - `setupGlobalEventListeners()` - Sets up global events
  - `setupDragDrop()` - Drag and drop functionality
  - `handlePadClick()` / `handlePadDrag()` - Pad interactions

## 🎨 CSS Organization

The stylesheet is organized into logical sections:

1. **CSS Variables** - Theme colors, spacing, transitions
2. **Reset & Base** - Browser resets and base styles
3. **Layout** - Grid system and containers
4. **Components** - Reusable UI components
   - Buttons
   - Sliders
   - Dropdowns
   - Status bars
5. **Modals** - Modal windows and tabs
6. **Parameters** - Parameter controls and grids
7. **Modulation** - Modulation slot styling
8. **Responsive** - Mobile/tablet breakpoints
9. **Accessibility** - Focus states and reduced motion

### CSS Improvements Made:

- **CSS Variables** for easy theming and consistency
- **Better organization** with clear section comments
- **Improved accessibility** with focus states

- **Responsive design** with mobile-friendly breakpoints
- **Reduced motion** support for accessibility
- **Hover states** for better interactivity feedback
- **Consistent spacing** using spacing variables

## 🔄 Data Flow

```
User Action
  ↓
UI Controller (captures event)
  ↓
Validation (utils.js)
  ↓
Update Data Structure (data-structures.js)
  ↓
Update UI (ui-controller.js)
  ↓
Auto-save to Internal State
```

### Example: Editing a Pad Parameter

1. User adjusts slider in pad editor
2. `pad-editor.js` captures input event
3. `utils.js` formats the value
4. `data-structures.js` validates and updates pad data
5. `ui-controller.js` updates display value
6. Change is reflected in internal `presetData` object

### Example: Loading a Preset

1. User selects XML file
2. `xml-handler.js` parses XML structure
3. `data-structures.js` creates preset object
4. `ui-controller.js` updates entire UI
5. `pad-editor.js` refreshes pad grid
6. `main.js` updates project name

## 🚀 Getting Started

### Running Locally

1. Simply open `index.html` in a modern web browser
2. No build process required!
3. All dependencies are included

### Browser Compatibility

- Chrome/Edge (recommended)
- Firefox
- Safari 14+

**Note:** File drag-and-drop may behave differently across browsers.

## 🛠️ Development

### Adding a New Parameter

1. Add constant to `config.js` if needed
2. Update data structure in `data-structures.js`
3. Add UI element to `index.html`
4. Add parameter handler in `pad-editor.js` or `fx-editor.js`
5. Update XML parsing/generation in `xml-handler.js`

### Adding a New Modulation Destination

1. Add to `MOD_DESTINATIONS` in `config.js`
2. No other changes needed! System is automatic.

## 📝 Code Style

- **Comments:** Use JSDoc-style comments for functions
- **Naming:**
  - camelCase for functions and variables
  - UPPERCASE for constants
  - Descriptive names (no single letters except loops)
- **Organization:** Group related functions together
- **Error Handling:** Always use try-catch for file operations

## 🐛 Known Issues

- localStorage/sessionStorage not supported (by design for Claude.ai)
- Asset cells (multi-samples) preserved but not fully editable yet
- Some Bitbox parameters may not have UI controls yet

## 📄 License

This project is open source. The included fflate.js library is MIT licensed.

## 🤝 Contributing

Feel free to improve the code! Areas that could use enhancement:

1. Multi-sample (asset) editing UI
2. Additional parameter visualizations
3. Preset browser/library
4. Undo/redo functionality
5. Keyboard shortcuts
6. Preset templates

## 📞 Support

For issues or questions about the Bitbox hardware, visit:

- 1010music Website
- 1010music Forum

---

**Built with ❤️ for the Bitbox community**