**DokuSign - eIDAS Compliant E-Signature Tool**

**Overview**

DokuSign is a web-based electronic signature application that complies with eIDAS (Electronic Identification, Authentication and Trust Services) regulations. This tool allows users to upload documents, add signatures, dates, custom text fields, and images, and generate legally compliant electronically signed documents.

**Key Features**

* PDF and image document upload and rendering
* Signature drawing and capture
* Date and custom field addition
* eIDAS compliance with Simple Electronic Signatures (SES)
* GDPR-compliant data handling
* Document integrity verification with SHA-256 hashing
* OTP-based identity verification
* Timestamping of signatures
* Audit logging for compliance

**Installation**

1. Create the following directory structure:

dokusign/

├── index.html

├── css/

│ └── styles.css

└── js/

├── app.js

├── file-handling.js

├── signature-pad.js

├── fields.js

├── document-rendering.js

├── eidas-compliance.js

└── gdpr-compliance.js

1. Copy the provided fixed code files into their corresponding locations.
2. Make sure to include the required external libraries:
   * PDF.js (for PDF rendering)
   * jsPDF (for PDF generation)

**Bugfixes and Improvements**

The following issues were fixed in the provided code:

1. **JavaScript Syntax Errors**:
   * Fixed incomplete function in app.js
   * Corrected missing function closures
   * Fixed event handler references
2. **CSS Styling Issues**:
   * Added proper styles for field containers
   * Improved responsive design
   * Added visual feedback for dragging and resizing
   * Fixed layout issues in different steps
3. **Functionality Issues**:
   * Fixed signature pad to properly capture and render signatures
   * Corrected document rendering logic
   * Improved field placement and positioning
   * Enhanced the user experience with better validation
4. **eIDAS Compliance**:
   * Ensured proper document hashing
   * Implemented OTP verification
   * Added timestamping
   * Included signature validation
5. **Performance Optimizations**:
   * Added loading screens
   * Optimized canvas rendering
   * Improved document processing

**Usage Instructions**

**For End Users**

1. **Upload Document**:
   * Upload a PDF or image document
   * Enter your name and email
   * Consent to data processing
   * Click "Continue to Step 2"
2. **Add Signature**:
   * Draw your signature in the provided canvas
   * Click "Save Signature"
   * Click "Continue to Step 3"
3. **Add Date**:
   * Select a date (optional)
   * Click "Continue to Step 4"
4. **Add Text Fields**:
   * Add custom text fields (optional)
   * Click "Continue to Step 5"
5. **Add Image**:
   * Upload an image (optional)
   * Click "Continue to Step 6"
6. **Add Recipients**:
   * Add recipient names and emails (optional)
   * Click "Continue to Step 7"
7. **Place Fields**:
   * Drag fields onto the document or right-click to place them
   * Confirm placement by clicking the checkmark
   * Complete identity verification
   * Accept the legal disclaimer
   * Click "Continue to Review"
8. **Review & Download**:
   * Review the final document
   * Click "Download Document" to save the signed PDF

**For Developers**

The application is structured in modules for easier maintenance:

1. **app.js**: Main application logic and coordination between modules
2. **file-handling.js**: Handles document uploads and rendering
3. **signature-pad.js**: Manages signature drawing and saving
4. **fields.js**: Handles field creation, placement, and management
5. **document-rendering.js**: Renders the final document with fields
6. **eidas-compliance.js**: Implements eIDAS compliance requirements
7. **gdpr-compliance.js**: Handles data privacy requirements

**Customization**

You can customize the application by:

1. **Changing the visual appearance**:
   * Modify the styles.css file
   * Update colors, fonts, and layout
2. **Adding new field types**:
   * Extend the fields.js module
   * Implement new field types in document-rendering.js
3. **Enhancing eIDAS compliance**:
   * Modify the eidas-compliance.js to add additional security measures
   * Implement Advanced Electronic Signatures (AES) or Qualified Electronic Signatures (QES)
4. **Adding server-side integration**:
   * Implement actual email sending for OTPs
   * Add secure storage for signed documents
   * Create a real backend for audit logging

**Troubleshooting**

If you encounter issues:

1. **PDF not rendering**:
   * Ensure PDF.js is properly loaded
   * Check browser console for errors
   * Verify that the PDF file is valid
2. **Signature not saving**:
   * Check if canvas drawing is working
   * Verify that the signature pad is properly initialized
3. **Fields not appearing on the document**:
   * Check if field placement is working
   * Verify that fields are properly created and positioned
4. **Download not working**:
   * Ensure jsPDF is properly loaded
   * Check if the final document rendering is complete

**Security Considerations**

This implementation includes several security features:

1. **Document Integrity**:
   * SHA-256 hashing to verify document integrity
   * Timestamping to prevent backdating
2. **Identity Verification**:
   * Email-based OTP verification
   * Audit logging of actions
3. **Data Privacy**:
   * GDPR-compliant data handling
   * Data deletion request functionality

For production use, consider implementing:

1. Server-side verification of signatures
2. Secure storage of signed documents
3. Enhanced encryption of personal data
4. Integration with trusted timestamp services
5. Server-side audit logging

**Conclusion**

This implementation provides a solid foundation for an eIDAS-compliant e-signature tool. With proper server-side integration, it can be used for legally binding electronic signatures in the European Union.