Component Documentation

File Upload

A comprehensive file upload component that provides multiple layout variants, drag-and-drop functionality, file validation, and preview capabilities. Built with accessibility in mind and designed for various file upload scenarios including single file uploads, multiple file selections, and enterprise file management interfaces.

How to use

import { AavaFileUploadComponent } from "@aava/play-core" ;

Note : The FileUpload component is standalone and includes all necessary dependencies for icon, button, tag, and common modules.

Basic Usage

Simple file upload with default layout and basic functionality.

```
<aava-file-upload
  [allowedFormats]="['pdf', 'doc', 'docx', 'txt', 'jpg', 'png']"
  (selectedList)="onFileChange($event)"
>
</aava-file-upload>

---

selectedFiles: File[] = [];

  onFileChange(files: File[]): void {
    console.log('Files changed:', files);
    this.selectedFiles = files;
  }
```

Layout Variants

```
<div>
  <h1>Default Layout</h1>
  <aava-file-upload
    layout="default"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png','txt']"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
  <p>Single file with left-side positioning</p>
  <aava-file-upload
    singleFileSelectionPosition="left"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Icon Layout</h1>
  <aava-file-upload
    layout="icon"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>List Layout</h1>
  <aava-file-upload
    layout="list"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Tags Layout</h1>
  <aava-file-upload
    layout="tags"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Table Layout</h1>
  <aava-file-upload
    (selectedList)="onFilesChanged($event, 'advanced')"
    (deletedList)="onDeletedFiles($event)"
    [allowedFormats]="allowedFormats"
    layout="icon"
    [maxFileSize]="maxFileSize"
    uploaderId="advanced"
```

```
    deleteIconPosition="right"
    deleteIconName="trash"
    deleteButtonSize="sm"
    previewLayout="table"
    [previewData]="previewData"
  >
  </aava-file-upload>
</div>

---

previewData = [
    {
      fileName: '1.png',
      fileSize: '5kb',
      fileType: 'png',
    },
  ];
    public maxFileSize = 3 * 1024 * 1024; //3MB
    public allowedFormats: string[] = [
    'jpeg',
    'jpg',
    'png',
    'svg',
    'doc',
    'docx',
    'xlsx',
    'txt',
    'pdf',
    ];

    onFileChange(files: File[]): void {
      console.log('Files changed:', files);
    }

    uploadedFiles: Record<string, File[]> = {};
    upDisabled = false;


    onFilesChanged(files: File[], uploaderId: string) {
      this.uploadedFiles[uploaderId] = [...files];
      this.upDisabled = true;

      console.log(
        `Files list changed for ${uploaderId}:`,
        files.map((f) => f.name)
      );
    }

    onDeletedFiles(files: UploadFile[]) {
      console.log('Deleted files:', files);
    }
```

The component supports multiple layout options to fit different UI contexts and use cases:
default – Standard button and file display layout.

- list – Displays files in a stacked list with remove options.
- tags – Shows files as removable tags, suitable for compact interfaces.

• icon – Drag-and-drop area with rich visual preview and side-by-side file listing.
• Each layout offers unique ways to visualize uploaded files and actions.

Size Variants

```
<div>
  <h1>Extra Small (xs)</h1>
  <aava-file-upload
    componentTitle="XS Upload"
    size="xs"
    uploadButtonSize="xs"
    deleteButtonSize="xs"
    [allowedFormats]="['pdf', 'doc', 'jpg']"
    [maxFiles]="2"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Small (sm)</h1>
  <aava-file-upload
    componentTitle="Small Upload"
    size="sm"
    uploadButtonSize="sm"
    deleteButtonSize="sm"
    [allowedFormats]="['pdf', 'doc', 'jpg']"
    [maxFiles]="2"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Medium (md)</h1>
  <aava-file-upload
    componentTitle="Medium Upload"
    size="md"
    uploadButtonSize="md"
    deleteButtonSize="md"
    [allowedFormats]="['pdf', 'doc', 'jpg']"
    [maxFiles]="2"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Large (lg)</h1>
  <aava-file-upload
    componentTitle="Large Upload"
    size="lg"
    uploadButtonSize="lg"
    deleteButtonSize="lg"
    [allowedFormats]="['pdf', 'doc', 'jpg']"
    [maxFiles]="2"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Extra Large (xl)</h1>
  <aava-file-upload
    componentTitle="XL Upload"
```

```
    size="xl"
    uploadButtonSize="xl"
    deleteButtonSize="xl"
    [allowedFormats]="['pdf', 'doc', 'jpg']"
    [maxFiles]="2"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Tags Layout with xs button size</h1>
  <aava-file-upload
    layout="tags"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    uploadButtonSize="xs"
    deleteButtonSize="xs"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

---

onFileChange(files: File[]): void {
    console.log('Files changed:', files);
  }
```

Available Sizes

File Upload buttons support multiple size options to align with your design scale:

- xs (Extra Small) – For dense layouts.
- sm (Small) – Compact option.
- md (Medium) – Default size for general usage.
- lg (Large) – For prominent actions or primary uploads.
- xl (Extra Large) – When emphasizing upload as a major user task.

File Validation

```
<div>
  <h1>Format Validation</h1>
  <p>Only PDF and DOC files allowed</p>
  <aava-file-upload
    componentTitle="PDF/DOC Only"
    [allowedFormats]="['pdf', 'doc']"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Size Validation</h1>
  <p>Maximum 1MB per file</p>
  <aava-file-upload
    componentTitle="Max 1MB Files"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFileSize]="1048576"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Max Files Validation</h1>
  <p>Maximum 2 files allowed</p>
  <aava-file-upload
    componentTitle="Max 2 Files"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    layout="tags"
    [maxFiles]="2"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Duplicate Detection</h1>
  <p>Shows error for duplicate files</p>
  <aava-file-upload
    componentTitle="No Duplicates"
    layout="tags"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [showDuplicateError]="true"
    duplicateErrorText="This file already exists!"
    [maxFiles]="5"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

---

onFileChange(files: File[]): void {
    console.log('Files changed:', files);
  }
```

Built-in validation ensures that uploaded files meet specific criteria:

• Allowed Formats : Restrict file types with the allowedFormats input.
• File Size Limit : Prevent large files using the maxFileSize property.
• Duplicate Check : Detect and warn users about duplicate uploads.
Validation feedback is displayed inline with error messages.

Single vs Multiple

```
<div>
  <h1>Single File Mode</h1>
  <p>Only one file can be selected at a time</p>
  <aava-file-upload
    componentTitle="Select One File"
    [singleFileMode]="true"
    layout="tags"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    (selectedList)="onSingleFileChange($event)"
  >
  </aava-file-upload>

  <div class="result" *ngIf="singleFile.length > 0">
    <strong>Selected:</strong> {{singleFile[0].name}}
  </div>
</div>

<div>
  <h1>Multiple Files Mode</h1>
  <p>Multiple files can be selected (up to 5)</p>
  <aava-file-upload
    componentTitle="Select Multiple Files"
    [singleFileMode]="false"
    layout="tags"
    [maxFiles]="3"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    (selectedList)="onMultipleFileChange($event)"
  >
  </aava-file-upload>

  <div class="result" *ngIf="multipleFiles.length > 0">
    <strong>Selected Files:</strong> {{multipleFiles.length}}
    <ul>
      <li *ngFor="let file of multipleFiles">{{file.name}}</li>
    </ul>
  </div>
</div>

---

singleFile: File[] = [];
  multipleFiles: File[] = [];

  onSingleFileChange(files: File[]): void {
    console.log('Single file changed:', files);
    this.singleFile = files;
  }

  onMultipleFileChange(files: File[]): void {
    console.log('Multiple files changed:', files);
    this.multipleFiles = files;
  }
```

Configure whether users can upload a single file or multiple files at once:

- Use singleFileMode = true for a single-file input.
- Keep it false to allow multi-selection.

The component automatically manages file replacement and count validation.

States & Customization

```
<div>
  <h1>Normal State</h1>
  <aava-file-upload
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Disabled State</h1>
  <aava-file-upload
    [disabled]="true"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Custom Styling</h1>
  <aava-file-upload
    componentTitle="Custom Colors"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    layout="tags"
    [maxFiles]="3"
    borderColor="green"
    dividerColor="green"
    [customStyles]="{'--fileupload-upload-area-default-border': '2px dashed #007bff'}"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Custom Button Variants</h1>
  <aava-file-upload
    componentTitle="Custom Buttons"
    uploadButtonVariant="secondary"
    deleteButtonVariant="danger"
    layout="tags"
    uploadButtonLabel="Choose Files"
    deleteButtonLabel="Clear All"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
    (deletedList)="onFilesDeleted($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Animation Enabled</h1>
  <aava-file-upload
    componentTitle="Animated Upload"
    [enableAnimation]="true"
    layout="tags"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
```

```
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Custom Icons</h1>
  <aava-file-upload
    componentTitle="Custom Icons"
    uploadIconName="folder"
    deleteIconName="trash-2"
    tagIcon="x-circle"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div class="result" *ngIf="selectedFiles.length > 0">
  <strong>Files Selected:</strong> {{selectedFiles.length}}
</div>

<div class="result" *ngIf="deletedFiles.length > 0">
  <strong>Files Deleted:</strong> {{deletedFiles.length}}
</div>

---

selectedFiles: File[] = [];
  deletedFiles: UploadFile[] = [];

  onFileChange(files: File[]): void {
    console.log('Files changed:', files);
    this.selectedFiles = files;
  }

  onFilesDeleted(deletedFiles: UploadFile[]): void {
    console.log('Files deleted:', deletedFiles);
    this.deletedFiles = deletedFiles;
  }
```

Customize button variants, labels, colors, and disabled states for consistent UI theming.
The component supports:

- Custom border and divider colors
- Dynamic styles via customStyles
- Disabled state to prevent interactions
- Configurable button variants for upload and delete actions

Preview Layouts

```
<div>
  <h1>Default Preview Layout</h1>
  <aava-file-upload
    componentTitle="Default Preview"
    previewLayout="default"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="5"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Table Preview Layout</h1>
  <aava-file-upload
    componentTitle="Table Preview"
    layout="icon"
    previewLayout="table"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="5"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Table with Custom Height (200px)</h1>
  <aava-file-upload
    componentTitle="Scrollable Table"
    previewLayout="table"
    layout="icon"
    [previewLayoutHeight]="200"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="10"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Table with Custom Columns</h1>
  <aava-file-upload
    previewLayout="table"
    layout="icon"
    [tableColumns]="customColumns"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="5"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

---

selectedFiles: File[] = [];

  customColumns = [
    { key: 'fileName', label: 'File Name', visible: true },
    { key: 'fileSize', label: 'Size', visible: true },
    { key: 'fileType', label: 'Type', visible: false }, // Hidden column
```

```
    { key: 'uploadDate', label: 'Date', visible: true },
    { key: 'actions', label: 'Actions', visible: true },
  ];

  onFileChange(files: File[]): void {
    console.log('Files changed:', files);
    this.selectedFiles = files;
  }
```

Display uploaded files in different preview formats:

• default – Shows file cards with color-coded type indicators.
• table – Tabular view with configurable columns (fileName, fileSize, fileType, etc.)
Each layout provides structured visibility of uploaded files and supports deletion.

Advanced Features

```
ERROR: empty code block
```

Explore extended functionality for advanced scenarios:

• Custom table columns for preview layouts.
• Dynamic color mapping for file types.
• Custom upload animations.
• Event emitters (selectedList, deletedList) for parent integration.

These options make the File Upload component flexible for enterprise-grade workflows.

Icon Customization

```
<div>
  <h1>Upload Icon - Left Position</h1>
  <aava-file-upload
    componentTitle="Icon Left"
    uploadIconPosition="left"
    uploadIconName="upload"
    uploadButtonLabel="Upload Files"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Upload Icon - Right Position</h1>
  <aava-file-upload
    componentTitle="Icon Right"
    uploadIconPosition="right"
    uploadIconName="Upload"
    uploadButtonLabel="Choose Files"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>Upload Icon - Only (No Text)</h1>
  <aava-file-upload
    componentTitle="Icon Only"
    uploadIconPosition="only"
    uploadIconName="upload"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="3"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>

<div>
  <h1>All Custom Icons Combined</h1>
  <aava-file-upload
    componentTitle="All Custom"
    uploadIconPosition="right"
    uploadIconName="cloud-upload"
    uploadButtonLabel="Upload to Cloud"
    deleteIconPosition="left"
    deleteIconName="trash"
    deleteButtonLabel="Delete All"
    tagIcon="x"
    layout="tags"
    [allowedFormats]="['pdf', 'doc', 'jpg', 'png']"
    [maxFiles]="5"
    (selectedList)="onFileChange($event)"
  >
  </aava-file-upload>
</div>
```

```
---

selectedFiles: File[] = [];

  onFileChange(files: File[]): void {
    console.log('Files changed:', files);
    this.selectedFiles = files;
  }
```

Adjust icon styles and placement for both upload and delete actions using these inputs:

• uploadIconName / deleteIconName – Change icon symbols.
• uploadIconPosition / deleteIconPosition – Control left, right, or icon-only layouts.
• tagIcon – Set custom tag icons for tag-based uploads.

This allows seamless integration with your application’s visual language.

Features

Upload Methods

• Click to Upload : Traditional file selection via file picker
• Drag and Drop : Intuitive drag-and-drop file upload
• Multiple File Support : Upload single or multiple files
• File Replacement : Replace existing files in single file mode

File Validation

• Format Validation : Restrict file types based on extensions
• Size Validation : Enforce maximum file size limits
• Duplicate Prevention : Prevent duplicate file uploads
• Error Handling : Clear error messages for validation failures

Layout Options

• Inline : Compact form integration
• List : Simple list-based display
• Medium Variant : Enhanced with file tags
• Large Variant : Full-featured with preview
• Customizable : Flexible sizing and theming

User Experience

• Visual Feedback : Clear upload states and progress indicators
• File Preview : Preview uploaded files with metadata
• Responsive Design : Adapts to different screen sizes
• Accessibility : Full keyboard navigation and screen reader support

API Reference

Inputs

Property	Type	Default	Description
theme	'light' | 'dark'	'light'	Theme variant for the component
uploaderId	string	''	Unique identifier for the uploader
enableAnimation	boolean	false	Enable animation effects
allowedFormats	string[]	['jpeg', 'jpg', 'png', 'svg', 'doc', 'docx', 'xlsx', 'txt', 'pdf']	Allowed file extensions
singleFileMode	boolean	false	Restrict to single file upload
maxFiles	number | null	null	Maximum number of files allowed
componentTitle	string	'Upload File Here'	Title displayed in the upload area
supportedFormatLabel	string	'Supported file formats'	Label for supported formats section
maxFileSize	number	3145728 (3MB)	Maximum file size in bytes
showDialogCloseIcon	boolean	true	Show close icon in dialog mode
showUploadButton	boolean	true	Show upload button
layout	'default' | 'inline' | 'list' | 'md-variant' | 'lg-variant'	'default'	Layout variant for the component
previewLayout	'default' | 'table'	'default'	Preview layout type
previewLayoutHeight	number	0	Custom height for preview layout
singleFileSelectionPosition	'right' | 'left'	'right'	Position of single file selection button
uploadButtonSize	'xs' | 'sm' | 'md' | 'lg' | 'xl'	'md'	Upload button size
uploadButtonLabel	string	'Upload'	Upload button label text
uploadButtonVariant	ButtonVariant	'primary'	Upload button style variant

Property	Type	Default	Description
uploadIconPosition	'left' | 'right' | 'only'	'left'	Upload button icon position
uploadIconName	string	'upload'	Upload button icon name
deleteButtonSize	'xs' | 'sm' | 'md' | 'lg' | 'xl'	'md'	Delete button size
deleteButtonLabel	string	'Remove All'	Delete button label text
deleteButtonVariant	ButtonVariant	'secondary'	Delete button style variant
deleteIconPosition	'left' | 'right' | 'only'	'left'	Delete button icon position
deleteIconName	string	'circle-x'	Delete button icon name
disabled	boolean	false	Disable the file uploader component
previewData	any	undefined	External preview data
customStyles	Record	{}	CSS custom properties override
borderColor	string	'#9ca1aa'	Border color for upload area
ellipses	'start' | 'end' | 'middle'	'end'	Ellipses style for long file names
dividerColor	string	'#9ca1aa'	Divider line color
tagIcon	string	'x'	Icon for removable tags
tableColumns	customColumns[]	Predefined column config	Configurable columns for table layout
fileTypeColors	{ [key: string]: string }	undefined	Dynamic colors for file types
border	string	'var(--fileupload-upload-area-default-border)'	Border style for upload area

Outputs

Event	Type	Description
selectedList	EventEmitter	Emitted when a list of files is selected

Event	Type	Description
deletedList	EventEmitter	Emitted when a list of files is deleted

Properties

Property	Type	Description
uploadedFiles	File[]	Array of currently uploaded files
fileUploadedSuccess	boolean	Whether file upload was successful
fileFormatError	boolean	Whether there's a file format error
fileSizeError	boolean	Whether there's a file size error
maxFilesError	boolean	Whether maximum files limit is exceeded
isUploadActive	boolean	Whether upload is currently active
viewAll	boolean	Whether to show all files or truncated list
uniqueId	string	Unique identifier for the uploader

Methods

Method	Parameters	Return	Description
openFileSelector()	None	void	Open file selection dialog
resetUpload()	None	void	Reset upload state and clear files
closeUpload()	None	void	Close upload and reset state
removeNewFile()	file: File	void	Remove specific file from list
getFileExtension()	filename: string	string	Get file extension from filename

Method	Parameters	Return	Description
toggleViewAll()	None	void	Toggle between truncated and full file list
allowAccepted()	None	string	Get accepted file types string for input

Computed Properties

Property	Type	Description
allowedFormatsList	string[]	List of allowed file formats

Layout Variants

Default Layout

The standard file upload interface with drag-and-drop support and file list display.

Inline Layout

Compact uploader designed for form integration with minimal space requirements.

Features:

- Single file display
- Upload button with icon
- File name with remove option
- Minimal footprint

List Layout

Simple list-based file display with upload button and file management.

Features:

- List of uploaded files
- File icons and names
- Individual file removal
- Multiple file support

Medium Variant

Enhanced uploader with file tags and grid layout for better file organization.

Features:

• File count header
• Grid-based file display
• File tags with remove functionality
• Bulk remove all option
• Empty state with format information

Large Variant

Full-featured uploader with comprehensive file management and preview capabilities.

Features:

• Drag-and-drop interface
• File preview panel
• Detailed file information
• File type icons
• Comprehensive error handling
• Bulk file management

File Validation

Format Validation

The component validates file types based on the allowedFormats input:

// Default allowed formats allowedFormats = ["jpeg" , "jpg" , "png" , "svg" , "doc" , "docx" , "xlsx" , "txt" , "pdf" ,] ; // Custom format restriction allowedFormats = ["pdf" , "doc" , "docx"] ; // Only document files

Size Validation

File size validation is controlled by the maxFileSize input:

// 3MB default maxFileSize = 3 * 1024 * 1024 ; // 10MB limit maxFileSize = 10 * 1024 * 1024 ; // 1MB limit maxFileSize = 1024 * 1024 ;

Duplicate Prevention

The component automatically prevents duplicate file uploads by checking:

• File name
• File size
• File content (basic comparison)

Error Handling

File Format Error

Displayed when an unsupported file type is selected:

< div class = " error-message " role = " alert " > Invalid file type. Allowed formats: JPEG, JPG, PNG, SVG, DOC, DOCX, XLSX, TXT, PDF.

File Size Error

Displayed when a file exceeds the maximum size limit:

< div class = " error-message " role = " alert " > File is too large. Maximum size allowed is 3 MB

Maximum Files Error

Displayed when the maximum file limit is exceeded:

< div class = " error-message " role = " alert " > Maximum of 5 files allowed

Best Practices

Design Guidelines

• Layout Selection : Choose appropriate layout variant for your use case
• File Limits : Set reasonable file size and count limits
• Format Restrictions : Clearly communicate supported file types
• Visual Feedback : Provide clear upload states and progress indicators
• Error Handling : Display user-friendly error messages

Accessibility

• Keyboard Navigation : Ensure full keyboard support for all interactions
• Screen Reader Support : Provide clear labels and state announcements
• Drag and Drop : Support both mouse and keyboard file selection
• Error Announcements : Properly announce validation errors
• Focus Management : Clear focus indicators and logical tab order

Performance

• File Validation : Validate files on selection for immediate feedback
• Memory Management : Handle large files appropriately
• Batch Processing : Consider batch uploads for multiple files
• Progress Indicators : Show upload progress for better UX
• Error Recovery : Allow users to retry failed uploads

User Experience

• Clear Instructions : Provide clear upload instructions and format requirements
• Visual Feedback : Show upload states and file information
• File Preview : Allow users to review files before upload
• Bulk Operations : Support bulk file removal and management
• Responsive Design : Ensure uploader works on all device sizes

Integration

• Form Integration : Properly integrate with Angular forms
• Event Handling : Use the comprehensive event system for upload management
• State Management : Coordinate upload state with your application
• File Processing : Handle file processing and storage appropriately
• Error Handling : Implement proper error handling and user feedback

Responsive Behavior

Mobile Adaptations

The file upload component automatically adapts to mobile screens:

• Touch Optimization : Optimized touch targets for mobile interaction
• Mobile Layout : Appropriate layout adjustments for small screens
• File Selection : Mobile-friendly file selection methods
• Error Display : Mobile-optimized error message display

Breakpoint Behavior

• Desktop (>768px) : Full upload interface with all features
• Mobile (≤768px) : Compact layout with optimized spacing
• File Display : Responsive file list and preview
• Button Sizing : Appropriate button sizes for different screens

Content Considerations

• File Names : File names adapt to different screen widths
• Error Messages : Error messages maintain readability on small screens
• Upload Area : Upload area adapts to available space
• File List : File list maintains usability across screen sizes