instructions.md

---

applyTo: '**'

Discord.js Display Components - Copilot Instructions

This document provides comprehensive guidance for working with Discord's Display Components system using Discord.js. Use this as a reference when building Discord bots that utilize the new display components.

Overview

Display components provide a comprehensive set of layout and content elements that go beyond traditional Discord embeds. They offer more styling and formatting options for your app messages.

Key Requirements

To use display components, you **must**:

- Pass the `MessageFlags.IsComponentsV2` flag when sending a message
- Only use this flag when sending messages, not when deferring interaction responses

Important Limitations

When opting into display components (`IsComponentsV2`):

- ❌ Cannot send `content`, `poll`, `embeds`, or `stickers`
- ❌ Cannot opt out when editing a message
- ✅ Can opt in when editing by explicitly setting `content`, `poll`, `embeds`, and `stickers` to `null`
- 📊 Maximum 40 total components (nested components count)
- 📝 Maximum 4000 characters across all text display components
- 📎 All attached files must be explicitly referenced in a component

Component ID System

- Optional `id` field: 32-bit integer identifier for components
- Different from `custom_id` used for interactive components
- Discord auto-populates missing IDs sequentially starting from 1
- Value `0` is treated as empty
- Explicitly specify IDs if you need to reference them later

Component Types

1. Text Display Components

Replace the traditional `content` field with markdown-formatted text.

```
const { TextDisplayBuilder, MessageFlags } = require('discord.js');

const exampleTextDisplay = new TextDisplayBuilder()
    .setContent('This text supports **any __markdown__** formatting!');

await channel.send({
    components: [exampleTextDisplay],
    flags: MessageFlags.IsComponentsV2,
});
```

⚠️ **Warning**: User and role mentions in text display components will notify users! Control mentions with `allowedMentions`.

2. Section Components

Combine 1-3 Text Display components with an accessory (image thumbnail or button).

```
const { SectionBuilder, ButtonStyle, MessageFlags } = require('discord.js');

const exampleSection = new SectionBuilder()
    .addTextDisplayComponents(
        textDisplay => textDisplay
            .setContent('First text component with **markdown**'),
        textDisplay => textDisplay
            .setContent('Second text component'),
        textDisplay => textDisplay
            .setContent('Third text component with button accessory!')
    )
    .setButtonAccessory(
        button => button
            .setCustomId('exampleButton')
            .setLabel('Button inside Section')
            .setStyle(ButtonStyle.Primary)
    );

await channel.send({
    components: [exampleSection],
    flags: MessageFlags.IsComponentsV2,
});
```

Use Cases:

- Rich text content with interactive elements
- Combining multiple paragraphs with a call-to-action button
- Text with accompanying thumbnail image

3. Thumbnail Components

Visual elements similar to embed thumbnails, used as accessories in Section components.

```
const { AttachmentBuilder, SectionBuilder, MessageFlags } = require('discord.js');

const file = new AttachmentBuilder('../assets/image.png');

const exampleSection = new SectionBuilder()
    .addTextDisplayComponents(
        textDisplay => textDisplay
            .setContent('Text with thumbnail accessory')
    )
    .setThumbnailAccessory(
        thumbnail => thumbnail
            .setDescription('Alt text for accessibility')
            .setURL('attachment://image.png') // Or external URL
    );

await channel.send({
    components: [exampleSection],
    files: [file],
    flags: MessageFlags.IsComponentsV2,
});
```

Features:

- Alt text support for accessibility
- Spoiler marking capability
- Attachment or external URL support

4. Media Gallery Components

Display up to 10 media attachments in a grid layout.

```
const { AttachmentBuilder, MediaGalleryBuilder, MessageFlags } =
require('discord.js');

const file = new AttachmentBuilder('../assets/image.png');
```

```
const exampleGallery = new MediaGalleryBuilder()
    .addItems(
        mediaGalleryItem => mediaGalleryItem
            .setDescription('Alt text for attached image')
            .setURL('attachment://image.png'),
        mediaGalleryItem => mediaGalleryItem
            .setDescription('Alt text for external image')
            .setURL('https://i.imgur.com/AfFp7pu.png')
            .setSpoiler(true) // Displays as blurred
    );

await channel.send({
    components: [exampleGallery],
    files: [file],
    flags: MessageFlags.IsComponentsV2,
});
```

Best Practices:

- Always provide alt text for accessibility
- Use spoiler marking for sensitive content
- Combine local attachments with external URLs as needed

5. File Components

Display individual uploaded files within the message body.

```
const { AttachmentBuilder, FileBuilder, MessageFlags } = require('discord.js');

const file = new AttachmentBuilder('../assets/document.pdf');

const exampleFile = new FileBuilder()
    .setURL('attachment://document.pdf');

await channel.send({
    components: [exampleFile],
    files: [file],
    flags: MessageFlags.IsComponentsV2,
});
```

Notes:

- Cannot have alt text (unlike Thumbnail/Media Gallery)
- Can be marked as spoiler

- Use multiple File components for multiple files

6. Separator Components

Add vertical padding and optional visual division between components.

```
const { TextDisplayBuilder, SeparatorBuilder, SeparatorSpacingSize, MessageFlags }
= require('discord.js');

const textDisplay = new TextDisplayBuilder()
    .setContent('Content above separator');

const separator = new SeparatorBuilder()
    .setDivider(false) // No visual line
    .setSpacing(SeparatorSpacingSize.Large);

const textDisplay2 = new TextDisplayBuilder()
    .setContent('Content below separator');

await channel.send({
    components: [textDisplay, separator, textDisplay2],
    flags: MessageFlags.IsComponentsV2,
});
```

Options:

- Spacing: `SeparatorSpacingSize.Small` or `SeparatorSpacingSize.Large`
- Visual divider: `true` (default) or `false`

⚠️ **Warning**: Messages with only Separator components will have no visible content.

7. Container Components

Group child components in a visually distinct rounded box with optional accent color.

```
const { ContainerBuilder, UserSelectMenuBuilder, ButtonStyle, MessageFlags } =
require('discord.js');

const exampleContainer = new ContainerBuilder()
    .setAccentColor(0x0099FF)
    .addTextDisplayComponents(
        textDisplay => textDisplay
            .setContent('Text inside container with **markdown**')
    )
    .addActionRowComponents(
        actionRow => actionRow
```

```
            .setComponents(
                new UserSelectMenuBuilder()
                    .setCustomId('userSelect')
                    .setPlaceholder('Select users')
            )
    )
    .addSeparatorComponents(
        separator => separator
    )
    .addSectionComponents(
        section => section
            .addTextDisplayComponents(
                textDisplay => textDisplay
                    .setContent('Section text 1'),
                textDisplay => textDisplay
                    .setContent('Section text 2')
            )
            .setButtonAccessory(
                button => button
                    .setCustomId('containerButton')
                    .setLabel('Section Button')
                    .setStyle(ButtonStyle.Primary)
            )
    );

await channel.send({
    components: [exampleContainer],
    flags: MessageFlags.IsComponentsV2,
});
```

Features:

- Accent color on the left border
- No color = matches background color
- Spoiler marking blurs entire container
- Can contain any combination of display components

Common Patterns and Best Practices

Error Handling

```
try {
    await channel.send({
        components: [myComponent],
```

```
        flags: MessageFlags.IsComponentsV2,
    });
} catch (error) {
    if (error.code === 50035) {
        console.error('Component validation failed:', error.message);
    }
    // Handle other Discord API errors
}
```

Component Composition

```
// Reusable component builder
function createInfoSection(title, description, buttonLabel, buttonId) {
    return new SectionBuilder()
        .addTextDisplayComponents(
            textDisplay => textDisplay.setContent(`**${title}**`),
            textDisplay => textDisplay.setContent(description)
        )
        .setButtonAccessory(
            button => button
                .setCustomId(buttonId)
                .setLabel(buttonLabel)
                .setStyle(ButtonStyle.Secondary)
        );
}

// Usage
const infoSection = createInfoSection(
    'Help Topic',
    'This is helpful information about the topic.',
    'Learn More',
    'help_button'
);
```

File Attachment Management

```
const { AttachmentBuilder } = require('discord.js');

// Always reference attachments explicitly in components
const imageFile = new AttachmentBuilder('./image.png');
const docFile = new AttachmentBuilder('./document.pdf');

const gallery = new MediaGalleryBuilder()
    .addItems(
        item => item
```

```
            .setURL('attachment://image.png')
            .setDescription('Image description')
    );

const fileComponent = new FileBuilder()
    .setURL('attachment://document.pdf');

await channel.send({
    components: [gallery, fileComponent],
    files: [imageFile, docFile],
    flags: MessageFlags.IsComponentsV2,
});
```

Migration from Embeds

```
// Old embed approach
const embed = new EmbedBuilder()
    .setTitle('Title')
    .setDescription('Description')
    .setThumbnail('https://example.com/image.png')
    .addFields({ name: 'Field', value: 'Value' });

// New display components approach
const container = new ContainerBuilder()
    .setAccentColor(0x0099FF)
    .addTextDisplayComponents(
        textDisplay => textDisplay.setContent('**Title**\nDescription')
    )
    .addSectionComponents(
        section => section
            .addTextDisplayComponents(
                textDisplay => textDisplay.setContent('**Field**\nValue')
            )
            .setThumbnailAccessory(
                thumbnail => thumbnail
                    .setURL('https://example.com/image.png')
                    .setDescription('Thumbnail alt text')
            )
    );
```

Troubleshooting

Common Issues

1. **Component limit exceeded**: Keep track of nested components (they count toward the 40 limit)

2. **Text character limit**: Monitor total characters across all text components (4000 max)
3. **File not referenced**: All attached files must be explicitly referenced in components
4. **Missing MessageFlags.IsComponentsV2**: Required for all display component messages

Debugging Tips

```
// Count total components recursively
function countComponents(component) {
    let count = 1;
    if (component.components) {
        count += component.components.reduce((sum, child) =>
            sum + countComponents(child), 0);
    }
    return count;
}

// Count text characters
function countTextCharacters(components) {
    // Implement based on component types used
    return totalCharacters;
}
```

Integration with Interaction Handlers

```
// Handle button interactions from display components
client.on('interactionCreate', async interaction => {
    if (!interaction.isButton()) return;

    if (interaction.customId === 'exampleButton') {
        await interaction.reply({
            components: [
                new TextDisplayBuilder()
                    .setContent('Button was clicked!')
            ],
            flags: MessageFlags.IsComponentsV2,
            ephemeral: true
        });
    }
});
```

This guide provides comprehensive coverage of Discord's Display Components system. Always refer to the latest Discord.js documentation for updates and additional features.

Things to Note:

- **File imports**: Make sure you import all the required builders
- **Flags requirement**: Every interaction must include MessageFlags.IsComponentsV2
- **No regular content**: You cannot use the content field when using V2 components
- **Update consistency**: All updates must also use V2 components and flags