Dunwoody City Council Voting Tracker - Efficient Version

This is a refactored, more efficient version of the Dunwoody City Council voting transparency dashboard. The key improvement is separating meeting data into individual JSON files, making the system more maintainable and scalable.

File Structure

```
project/
├── index.html              # Main application file
├── config.json             # Configuration file
├── data-loader.js          # Data loading utility (optional)
├── meetings/
│   ├── 2025-07-28.json     # Individual meeting files
│   ├── 2025-08-25.json
│   └── 2025-09-08.json
└── README.md
```

Key Improvements

1. **Separated Data Files**

- Each meeting is now stored in its own JSON file
- Easy to add new meetings without touching the main HTML file
- Reduces file size and improves loading performance
- Better organization and maintainability

2. **Modular Architecture**

- `DunwoodyVotingTracker` class handles all application logic
- Separate data loading with error handling
- Cleaner separation of concerns

3. **Better Error Handling**

- Graceful handling of missing or corrupted meeting files
- Loading indicators and error messages
- Failed file loads don't break the entire application

4. **Configuration Management**

- Council members and meeting file lists stored in config.json
- Easy to update without code changes
- Centralized configuration

How to Add New Meetings

Step 1: Create the Meeting JSON File

Create a new file in the `meetings/` directory following the naming convention `YYYY-MM-DD.json`:

```json
{
  "date": "October 12, 2025",
  "status": "completed",
  "attendance": {
    "Mayor Deutsch": "present",
    "Harris": "present",
    "Heneghan": "zoom",
    "Lambert": "present",
    "Lautenbacher": "present",
    "Price": "absent",
    "Seconder": "present"
  },
  "motions": [
    {
      "title": "Motion Title Here",
      "description": "Detailed description of what the motion covers...",
      "introducedBy": "Council Member Name",
      "secondedBy": "Council Member Name",
      "votes": ["yes", "no", "yes", "yes", "abstain", "no", "yes"],
      "result": "passed"
    }
  ]
}
```

Step 2: Update Configuration

Add the new meeting file to the `meetingFiles` array in `config.json`:

```json
```

```
{
  "meetingFiles": [
    "meetings/2025-07-28.json",
    "meetings/2025-08-25.json",
    "meetings/2025-09-08.json",
    "meetings/2025-10-12.json"
  ]
}
```

That's it! The application will automatically load and display the new meeting data.

Data Format Specifications

Meeting File Structure

```json
{
  "date": "Month DD, YYYY",          // Human-readable date
  "status": "completed|upcoming|cancelled",
  "specialMeeting": "Optional description",  // For special/emergency meetings
  "attendance": {
    "Member Name": "present|zoom|absent"
  },
  "motions": [
    {
      "title": "Motion title",
      "description": "Detailed description",
      "introducedBy": "Member name (optional)",
      "secondedBy": "Member name (optional)",
      "votes": ["yes|no|abstain", ...],     // Array matching council member order
      "result": "passed|failed|tabled"
    }
  ]
}
```

Attendance Values

- `"present"` - Physically present at meeting
- `"zoom"` - Attended via video conference
- `"absent"` - Did not attend

Vote Values

- `"yes"` - Voted in favor
- `"no"` - Voted against
- `"abstain"` - Abstained from voting

Configuration Options

The `config.json` file supports these options:

```json
{
  "councilMembers": ["List", "of", "council", "members"],
  "meetingFiles": ["list/of", "meeting/files.json"],
  "siteName": "Site title",
  "siteSubtitle": "Site subtitle",
  "disclaimerText": "Disclaimer text...",
  "officialMinutesUrl": "URL to official minutes"
}
```

Development

For Simple Updates

Just edit the JSON files and refresh the page. No code changes needed.

For Advanced Features

The main application class `DunwoodyVotingTracker` can be extended with new methods. The modular structure makes it easy to add features like:

- Export functionality
- Advanced analytics
- Member comparison views
- Historical trend analysis

Error Handling

The system is designed to be resilient:

- Missing files are logged but don't break the app
- Invalid JSON is caught and reported

- Network issues are handled gracefully
- Users see loading states and error messages

Performance Benefits

1. **Faster Initial Load**: Smaller HTML file loads faster
2. **Parallel Loading**: Meeting files can be loaded simultaneously
3. **Caching**: Browser can cache individual meeting files
4. **Incremental Updates**: Only changed meetings need re-downloading
5. **Scalability**: Can easily handle hundreds of meetings

Browser Compatibility

Works in all modern browsers that support:

- ES6 Classes
- Async/Await
- Fetch API
- CSS Grid and Flexbox

Deployment

1. Upload all files to your web server
2. Ensure the `meetings/` directory is accessible
3. Test that JSON files load correctly
4. No server-side processing required - pure client-side application

This efficient structure makes it much easier to maintain the voting tracker as more meetings are added over time.