

Search...

Getting started  
Components

- M4Q
- About
- Population
- Constructor
- Ajax
- Animation
- Loops
- Visibility
- Effects
- Subtree functions
- Attributes
- Html, text and value
- Css and classes
- Position and size
- Manipulation
- DataSet
- Events
- Utils
- Base
- Containers
- Grid system
- Typography
- Tables
- Forms
- Buttons
- Images
- Figures
- Lists

Form components

- Checkbox
- File input
- Input
- Input material
- Keypad
- Rating
- Radio
- Select
- Slider
- Double Slider
- Spinner
- Switch
- Tag input
- Textarea
- Menus
- App bar
- Bottom navigation
- Bottom sheet
- Menu
- Ribbon Menu
- Side bar
- Side navigation
- Controls
- Accordion
- Badge
- Carousel
- Cards
- Cube
- Counter
- Charms
- Chat
- Donut
- Image compare
- Image magnifier
- Gravatar
- List

- ListView
- Master
- NavView
- Panels
- Progress & Activity
- Streamer
- Stepper
- Splitter
- Tabs
- Tabs material
- Table
- Tiles
- TreeView
- Wizard
- Information
- Dialogs
- Info box
- Hints
- Notify system
- Popovers
- Toasts
- Windows

- Date & time
- Calendar
- Calendar picker
- Date picker
- Time picker
- Countdown
- Formatting date
- Media

# List View

The `ListView` control provides the infrastructure to display a set of data items in different layouts or views.

## About

To creating `listview` we use `<ul>` element with role `data-role="listview"` . The items are defined with `<li>` element. A list can have multiple views: `list` , `content` , `icons` , `icons-small` , `icons-large` and `tiles` .

Video

Images

Documents

Downloads

Desktop

desktop.ini

setup.exe

file1.dat

```
<ul data-role="listview">
...
</ul>
```

## Add item

Each `listview` `item` are defined with `<li>` element and special attributes for it: `data-icon` , `data-caption` .

```
<li data-icon="<span class='mif-file-empty'>" data-caption="desktop.ini"></li>
<li data-icon="images/setup.png" data-caption="setup.exe"></li>
```

To create `icon` you can define `tag` (for font icons) or path to image for `<img>` .

## View types

A `listview` can have multiple views. To set view type use attribute `data-view="..."` with values: `list` (default), `content` , `tiles` , `icons` , `icons-medium` or `icons-large` . You can change this attribute at runtime to change `listview` view.

List

Table

Content

Tiles

Icons

Medium icons

Large icons

Video

Images

Documents

Downloads

Desktop

desktop.ini

setup.exe

file1.dat

12/21/2017

12/21/2017

12/21/2017

12/21/2017

12/21/2017

desktop.ini

setup.exe

file1.dat

Video library

My images

My documents

System folder

System folder

```
<ul data-role="listview" data-view="tiles">
...
</ul>
```

## View as content

For `content` view, you can define additional data with attribute `data-content="..."` . Value for this attribute must be simple string or valid html. Also supports Metro 4 components definition.

Video

Images

desktop.ini

setup.exe

file1.dat

Created: 12.21.2017

Executable

Data file

```
<ul data-role="listview" data-view="content" data-select-node="true">
<li data-icon="<span class='mif-folder fg-orange'>"
  data-caption="Video"
  data-content="<div class='mt-1' data-role='progress' data-value='35' data-small='true'>"></li>
<li data-icon="<span class='mif-folder fg-cyan'>"
  data-caption="Images"
  data-content="<div class='mt-1' data-role='progress' data-value='78' data-small='true'>"></li>
<li data-icon="<span class='mif-file-empty'>"
  data-caption="desktop.ini"
  data-content="<span class='text-muted'>Created: 12.21.2017</span>"></li>
<li data-icon="images/setup.png"
  data-caption="setup.exe"
```

### Table of contents

- ListView
- About
- Add item
- View types
- Grouping
- Selectable
- Events
- Methods









- Video player
- Audio player
- Tools
- Collapse
- Color module
- Draggable
- Dropdown
- Form validator
- Hotkeys
- Micro templates
- Ripple
- Storage
- Session storage
- Sorter
- Touch and swipe

Utilities  
Animations  
Additional

```
data-content="<span class='text-muted'>Executable</span>"></li>  
<li data-icon="<span class='mif-file-empty'>"  
  data-caption="file1.dat"  
  data-content="<span class='text-muted'>Data file</span>"></li>  
</ul>
```

View as table

For `table` view, you must define additional structure for items with attribute `data-structure="{...}"` . Value for this attribute must be valid json object stored as string. This structure define additional columns. Each item in structure writes as key/value, where key is a name for additional field and value can be true or false. True - field displayed, false - field not displayed. Now for each listview item you can set additional fields with data attribute `data-key_name="..."` where key is a key from structure data.

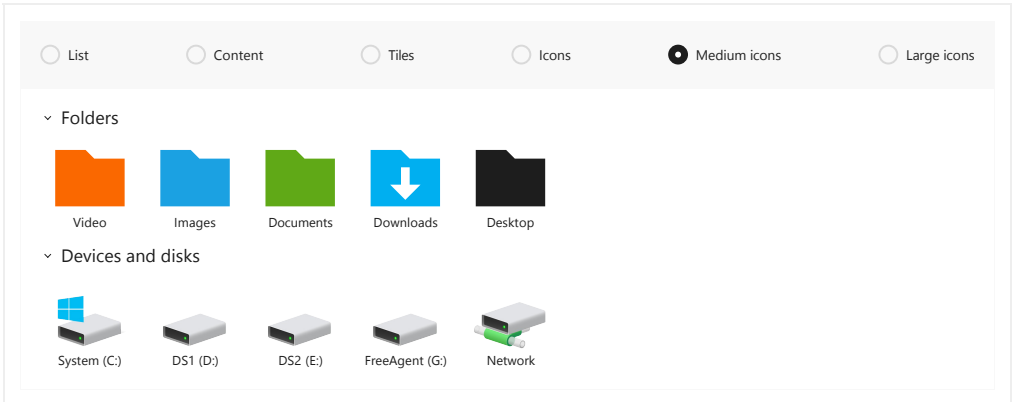
	Video	12/21/2017	Video library
	Images	12/21/2017	My images
	Documents	12/21/2017	My documents
	Downloads	12/21/2017	System folder
	Desktop	12/21/2017	System folder
	desktop.ini		
	setup.exe		
	file1.dat		

```
<ul data-role="listview"  
  data-view="table"  
  data-select-node="true"  
  data-structure="{"date": true, "name": true}">  
  
  <li data-icon="<span class='mif-folder fg-orange'>"  
    data-caption="Video"  
    data-date="12/21/2017"  
    data-name="Video library"></li>  
  
  <li data-icon="<span class='mif-folder fg-cyan'>"  
    data-caption="Images"  
    data-date="12/21/2017"  
    data-name="My images"></li>  
  
</ul>
```

You can set styles for additional items. Each additional item stored in element with classes `.node-data` + `.item-data-key_name` .

Grouping

Component `listview` is supports a **one level grouping** of items. To create `grouping` you must place list of items to second level. It's all.

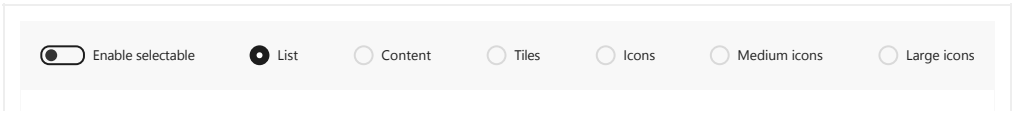


```
<ul data-role="listview" data-view="icons-medium">  
  <li data-caption="Folders">  
    <ul>  
      <li data-icon="<span class='mif-folder fg-orange'>" data-caption="Video"></li>  
      <li data-icon="<span class='mif-folder fg-cyan'>" data-caption="Images"></li>  
      <li data-icon="<span class='mif-folder fg-green'>" data-caption="Documents"></li>  
      <li data-icon="<span class='mif-folder-download fg-blue'>" data-caption="Downloads"></li>  
      <li data-icon="<span class='mif-folder'>" data-caption="Desktop"></li>  
    </ul>  
  </li>  
  <li data-caption="Devices and disks">  
    <ul>  
      <li data-icon="images/disk-os.png" data-caption="System (C:)"></li>  
      <li data-icon="images/disk.png" data-caption="DS1 (D:)"></li>  
      <li data-icon="images/disk.png" data-caption="DS2 (E:)"></li>  
      <li data-icon="images/disk.png" data-caption="FreeAgent (G:)"></li>  
      <li data-icon="images/disk-network.png" data-caption="Network"></li>  
    </ul>  
  </li>  
</ul>
```

If you need to create collapsing node, add attribute `data-collapsed="true"` to item definition.

Selectable

Attribute `data-selectable` enable or disable `selectable` mode for `listview` . You can change this attribute at runtime. Each checkbox linked with listview item on data attribute `node` .





```
<ul data-role="listview" data-selectable="true">
...
</ul>
```

## Events

When `listview` works, it generated the numbers of events. You can use callback for this events to behavior with `listview`.

Event	Data-*	Desc
onNodeClick(node)	<code>data-on-node-click</code>	Fired when user click on node caption
onNodeDbClick(node)	<code>data-on-node-dbclick</code>	Fired when user dbclick on node caption
onNodeInsert(node)	<code>data-on-node-insert</code>	Fired when node was inserted
onNodeDelete(node)	<code>data-on-node-delete</code>	Fired when node was deleted
onNodeClean(node)	<code>data-on-node-clean</code>	Fired when node was cleaned
onGroupNodeClick(state, check, node)	<code>data-on-radio-click</code>	Fired when user click on radio input
onExpandNode(node)	<code>data-on-expand-node</code>	Fired when node was expanded
onCollapseNode(node)	<code>data-on-collapse-node</code>	Fired when node was collapsed
onListViewCreate()	<code>data-on-listview-create</code>	Fired when node was created

## Methods

You can use `listview` methods to interact with the component at runtime.

- **toggleNode(node)** - toggle node state
- **add(parent, data)** - add node
- **addGroup(data)** - add group
- **insertBefore(node, data)** - add new node before specified node
- **insertAfter(node, data)** - add new node after specified node
- **del(node)** - delete node from tree
- **clean(node)** - clean node
- **getSelected()** - get selected nodes

