



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

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

M4Q Ajax

M4Q contains a method for using ajax requests via XMLHttpRequest.

About

M4Q contains a method for using `ajax` requests via `XMLHttpRequest` . Method return a `Promise` . With this method, you can send a request to the server and get a response from it. You can send to the server next data types: `form` , `input[type=file]` , `json` and `string` .

```
$.ajax([...options...]);
```

Options

To use ajax method you must define options for `$.ajax` method.

Request options

- **url** (required or can be defined as `action` in the `form`)
- **method** (default `GET`)
- **async** (default `true`)
- **data** (default `null`)
- **timeout** (default `null`)
- **withCredentials** (default `false`)
- **headers** (object with key->value for additional request headers)
- **returnValue** (must be `xhr` or `response` (default))

Events options

All events options is a functions. These functions, with the exception of `onSuccess` and `onFail` , receive two arguments: `event` and `xhr` . Function `onSuccess` receive `response from server` , function `onFail` receive `xhr` object.

- **onLoad**
- **onSuccess**
- **onFail**
- **onState**
- **onError**
- **onTimeout**
- **onProgress**
- **onLoadstart**
- **onLoadend**
- **onAbort**

```
$.ajax({
  method: "GET",
  url: "http://website.com",
  onLoad: function(e, xhr){
    console.log(xhr.response);
  }
})
```

Return value

Method `$.ajax` return a `Promise` . You can use `then()` method of the Promise or events options for obtaining result.

```
$.ajax({
  method: "GET",
  url: "http://website.com"
}).then(
  function(response){...}, // success
  function(xhr){...} // error
)
```

```
$.ajax({
  method: "GET",
  url: "http://website.com",
  onSuccess: function(response){
    console.log(response);
  },
  onFail: function(xhr){
    console.log(xhr);
  }
})
```

Ajax method aliases

Table of contents

About

- Media
 - 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

M4Q also contains short aliases for `$.ajax` : `$.post` , `$.get` , `$.put` , `$.patch` , `$.delete` and `$.json`

```
$.get(url, data, options)
```

Examples

Simple get with ajax

```
$.ajax({
  url: "http://website.com",
  method: "GET",
  data: {
    id: user_id,
    name: user_name
  }
}).then(
  function(response) {
    console.log(response);
  },
  function(xhr) {
    console.log(xhr.status, xhr.statusText);
  }
)
```

Get

```
$.get("http://website.com").then(
  function(response) {
    console.log(response);
  },
  function(xhr) {
    console.log(xhr.status, xhr.statusText);
  }
)
```

Get JSON

```
$.json("http://website.com").then(
  function(response) {
    console.log(response); // this is json object
  },
  function(xhr) {
    console.log(xhr.status, xhr.statusText);
  }
)
```

Post data

```
$.post("http://website.com", {
  id: user_id,
  name: user_name
}).then(
  function(response) {
    console.log(response);
  },
  function(xhr) {
    console.log(xhr.status, xhr.statusText);
  }
)
```

Post form

```
<form action="http://website.com" method="post" id="form1">
  <input type="text" name="user_name">
</form>

<script>
  $.post(null, document.getElementById("form1")).then(
    function(response) {
      console.log(response);
    },
    function(xhr) {
      console.log(xhr.status, xhr.statusText);
    }
  )
</script>
```

Post input file element

```
<input type="file" name="user_files" id="user_files">

<script>
  $.post("http://website.com", document.getElementById("user_files")).then(
    function(response) {
      console.log(response);
    },
    function(xhr) {
      console.log(xhr.status, xhr.statusText);
    }
  )
</script>
```

Post array

```
var myArray = ["Ben", "Abbot", "23"]
$.post("http://website.com", JSON.stringify(myArray)).then(
```

```
        function(response) {
            console.log(response);
        },
        function(xhr) {
            console.log(xhr.status, xhr.statusText);
        }
    )
}
```

Using events

```
var max_size_in_bytes = 104267776;
$.get(
    "long_archive_file.zip",
    null, {
        onProgress: function(e, x){
            console.log(e.loaded + " bytes transferred\n");
            if (e.loaded > max_size_in_bytes) {
                x.abort();
                console.log("Aborted");
            }
        },
        onState: function(e, x) {
            console.log(x.readyState);
        },
        onAbort: function(e, x) {
            console.log("Request aborted");
        }
    }
).then(function(response) {
    console.log("Long response executed");
}, function(xhr) {
    console.log("Error", xhr);
});
```