مدیریت فایل ها

وظیفه‌ی این ماژول بارگذاری، خواندن و ارسال فایل ها به سرور است. برای خواندن فایل ها از HTML5 File API استفاده می شود. همچنین برای ارسال فایل ها به سرور وب‌سوکت از Socket.IO استفاده می شود که امکان ارسال فایل ها بر اتصالات وب‌سوکت را می دهد. برای مستندات سرور سوکت فایل socket.md را مشاهده کنید.

نحوه کار

نحوه کار این ماژول به این صورت است که شما باید ابتدا یک شی `File` دریافت کنید و سپس با استفاده از آن نمونه ای جدید از کلاس `FileManager` بگیرید. پس از این مراحل بسته به نیازتان میتوانید از متد های این کلاس استفاده کنید که در ادامه به آنها اشاره خواهد شد.

نحوه ارسال فایل به سرور به این صورت می‌باشد که فایل قطعه قطعه می‌شود و این قطعه‌ها، به سرور ارسال می‌شوند. برای جلوگیری از حجم زیاد کار مرورگر که ممکن است باعث مشکلاتی برای کاربران و سرور شود، بعد از هر `n` قطعه ارسال شده، برای مدت زمان مشخصی استراحت داده می‌شود و سپس به کار ادامه داده می‌شود.

همچنین امکان `pause` و `resume` آپلود وجود دارد.

دریافت File

برای دریافت یک شی از نوع File می توانید از یکی از راه های زیر اقدام کنید:

input

ورودی های `<input>` از نوع `file` بعد از انتخاب فایل توسط کاربر،دارای، دارای مشخصه‌ی `files` هستند که یک `FileList` است. این شی مجموعه‌ای از فایل هایی است که کاربر انتخاب کرده است. دقت کنید که این نوع داده توسط `FileManager` پشتیبانی نمی‌شود و شما باید برای هر فایل یک نمونه جداگانه بگیرید؛ این به دلیل سختی مدیریت فایل های مختلف در صورت گروهی بودن است.

نمونه استفاده:

```
var selecedFile = input.files[0]; // first selected file
```

درگ دراپ

راه دیگر دریافت اطلاعات فایل‌های کاربر با استفاده از Drag/Drop است. به این صورت که در رویداد `drop` شما میتوانید با خواندن مشخصه `dataTransfer.files` شی رویداد فایل های دراپ شده روی عنصر مورد نظر توسط کاربر رو دریافت کنید، روش استفاده مانند ورودی است.

```
$(document.body).on('drop', function(e) {
    console.log( e.dataTransfer.files[0] );
});
```

برای اطلاعات بیشتر در مورد اشیاء به لینک های زیر مراجعه کنید:

FileList

File

Using Files from Web Pages

ساخت شی جدید

پس از دریافت فایل باید شی جدیدی را از کلاس `FileManager` بسازیم. تمامی تنظیمات قابل انجام در زیر توضیح داده شده‌اند.

```
new FileManager({
    url: 'localhost:8000',
    type: 'BinaryString',
    file: null,
    parts: 200,
    rest: 50,
    restDuration: 500,
    data: {}
});
```

url

این مشخصه آدرس سرور سوکت را مشخص میکند.

type

این مشخصه نحوه‌ی خواندن فایل را با استفاده از `FileReader` مشخص میکند که باید بسته به نحوه‌ی عملکرد سرور در برابر داده‌های ورودی تنظیم شود.

برای اطلاعات بیشتر در مورد این تنظیم و شی `FileReader` به لینک های زیر مراجعه کنید:

FileReader

file

فایل دریافتی که از نوع `File` می باشد.

parts

سایز هر قطعه فایل ارسالی به واحد کیلوبایت (به صورت پیش‌فرض ۲۰۰ کیلوبایت).

rest

این مشخصه تعیین کننده تعداد قطعات ارسالی بین دو استراحت است. یعنی بعد از ارسال موفق ۵۰ قطعه، استراحتی به مرورگر داده می‌شود.

restDuration

برای تعیین میزان زمان استراحت از این مشخصه استفاده کنید که واحد میلی‌ثانیه دارد.

data

این مشخصه از یک شی است که همراه با اولین درخواست برای آغاز آپلود به سرور ارسال می‌شود و در سرور به سوکت کاربر فعلی تخصیص داده‌می‌شود.

در حال حاظر در قسمت فایل ها اطلاعاتی که به سرور ارسال می‌شود شامل موارد زیر است:

- addr: آدرس فایل
- parent: پوشه والد فایل
- size: سایز فایل
- file: نام فایل
- user: کاربر صاحب فایل

متدهای کلاس

در این کلاس تعدادی متد وجود دارد که در اکثر مواقع شما نیازی به استفاده از آنها ندارید و این متدها به صورت داخلی استفاده می‌شوند، ولی هیچ الزامی وجود ندارد.

در ادامه متدهای این کلاس برای شما توضیح داده می‌شود.

لازم به ذکر است که اکثر متدها یک شی `Promise` ساخته شده با `jQuery.Deferred` بازگشت داده می‌شود که می‌توان از آن برای اجرای تابعی پس از اتمام کار متد استفاده کرد.

load → Promise

این متد **فایل حاظر** را بارگذاری می‌کند و نتیجه‌ی بارگذاری را در مشخصه `result` شی قرار می‌دهد. این متد یک `Promise` بازمیگرداند که می‌توانید از آن برای استفاده مستقیم از شی رویداد `onload` در صورت موفقیت آمیز بودن و `onerror` در صورت وجود خطا استفاده کنید.

برای اطلاعات دقیق‌تر در مورد این رویداد ها لینک زیر را مشاهده کنید: FileReader

رویدادها:

- file:error
- file:load

```
fm.load().then(function(e) {
    // e = FileReader.onload(e)
    this.result === e.target.result;
});
```

slice → this

(start: Number, end: Number)

این متد برای قطعه کردن فایل استفاده می‌شود. دقت کنید که قطعه کردن فایل متناسب با فایل اصلی است نه قسمت قطعه شده فعلی. بعد از قطعه کردن **فایل حاظر** تغییر می‌کند که باعث می‌شود متد هایی مثل `load` بجای بارگذاری کل فایل فقط این قسمت از فایل را بارگذاری کنند.

بعد از قطعه کردن فایل، فایل اصلی را میتوانید از مشخصه `originalFile` در دسترس داشته باشید و مشخصه `file` به فایل قطعه‌شده یا فایل حاظر اشاره می‌کند. همچنین مشخصه `range` یک آرایه دو آیتمی است که بعد از هر تغییر در فایل حاظر مشخص کننده محدوده‌ی تعیین شده برای فایل حاظر است.

رویدادها:

- file:slice

```
var file = new File(new Uint8ClampedArray(1000), 'myfile');
var f = new FileManager({file: file});
console.log(f.range); // [0, 1000]

f.slice(500, 600).load(...); // chain
console.log(f.range); // [500, 600];
```

full → this

این تابع فایل را به فایل اصلی باز می‌گرداند یعنی همانند `(this.slice(0, this.size` عمل می‌کند.

createStream → Promise

این متد برای شروع آپلود مورد استفاده است. روش کار این متد به این صورت است که درخواست اولیه‌ای با نام `meta` به سرور سوکت می‌فرستد که همانطور که گفته شد شامل نام و سایز فایل نیز می‌باشد. data

پس از ارسال درخواست بر روی پیغامی با نام `ready` شنود می‌کند و پس از دریافت پاسخ، که باید آی دی فایل ساخته شده در دیتابیس باشد، مشخصه‌ی `data.fileID` را معین می‌کند.

دلیل تعیین این مشخصه روی شی `data` این است که اگر در میان آپلود، اختلالی به وجود بیاید در ارسال

بعدی می‌توانیم این آی‌دی را ارسال کنیم تا سرور بداند که این فایل از قبل وجود دارد.

این متد `Promise` بازمیگرداند که پس از دریافت پاسخ `ready` حل می‌شود.

send → Promise

این متد فایل حاظر را با نام `data` به سرور ارسال می‌کند و پس از دریافت پاسخ `data-answer` مقدار بازگشتی خود را حل می‌کند.

upload → Promise

(start: Number, end: Number)

این متد اصلی ترین متد این کلاس است که معمولا جز آن نیازی به متد دیگری ندارید. کار این متد قطعه قطعه کردن فایل به تکه‌هایی با حجم تعیین شده در تنظیمات و ارسال آنها به صورت سری به سرور است. شما میتوانید با استفاده از آرگومان های `start` و `end` فقط قسمتی از فایل را به سرور بفرستید.

نحوه کار مرحله به مرحله:

1. اطمینان از اتصال سوکت: در صورت متصل نبودم سوکت منتظر اتصال خواهد ماند
2. اطمینان از وجود جریانی بین کاربر و سرور: در صورتی که قبلا متد `createStream` اجرا نشده باشد، این متد از اجرای آن اطمینان حاصل می‌کند
3. جرقه زدن رویداد `upload:start`
4. ساخت حلقه
5. در هر اجرای حلقه فایل را به کمک `slice` به قسمت‌های مساوی تعیین میکند
6. با استفاده از متد `send` قطعه فایل به سرور ارسال می‌شود.
7. پس از اطمینان از ارسال این قطعه، رویداد `upload:progress` جرقه می‌خورد که اطلاعاتی مربوط به تعداد تکه‌های ارسال شده، درصد و غیره دارد.
8. در صورتی که به تعداد تنظیم `rest` قطعه ارسال شده باشد، استراحت کوتاهی میکند و سپس ادامه می‌دهد.
9. اجرای دوباره حلقه

در مورد رویداد `upload:progress` شی زیر به عنوان آرگومان دوم به شنونده‌ها ارسال می‌شود:

```
fm.on('upload:progress', function(e, data) {
    console.log(data.min); // برابر با مقدار آرگومان start
    console.log(data.max); // برابر با مقدار آرگومان end
    console.log(data.sent); // مقدار بایت‌های ارسالی
    console.log(data.percentage); // درصد پیشرفت
});
```

pause → void

این متد روند آپلود را به طور موقت متوقف می‌کند.

resume → void

این متد در صورت قطع بودن آپلود آن را ادامه می‌دهد.

on → void

از این متد برای اتصال شنونده‌های رویدادها استفاده کنید.

destroy → void

این متد برای از بین بردن شی ساخته شده استفاده می‌شود که رویدادهای شنود شده را حذف می‌کند و همچنین مشخصاتی که ممکن است باعث اشغال بیهوده‌ی رم شوند را حذف می‌کند.