https://imasters.com.br/front-end/trabalhando-com-datas-em-javascript-com-momentjs
https://momentjs.com/docs/#/parsing/string-format/

Trabalhando com datas em JavaScript com MomentJS

Trabalhar com datas nunca é tão fácil; independente da linguagem que estamos utilizando, exige um certo cuidado e esforço. Pensando nisso, neste artigo tentarei simplificar e facilitar a nossa vida, apresentando a biblioteca MomentJS. Com ela, manipulação e criação de datas em JavaScript torna-se muito fácil.

Está pronto? Então vamos nessa!

Conhecendo a MomentJS

A biblioteca MomentJS realiza criações, parses, manipulações e validações. Também conseguimos mostrar as datas e horas, dentre outras diversas funcionalidades. Ela pode ser instalada de diferentes maneiras em nosso projeto; para a mais simples, é só realizar o download, ou se preferir o download minificado, simplesmente copie o conteúdo para um arquivo **.js** dentro do projeto; assim poderemos importá-lo através da tag **script.**

Ela também pode ser instalada via gerenciadores de pacote, como **yarn** ou **npm**:

Instalação com o Yarn

Para realizar a instalação com o Yarn, podemos utilizar o comando **add**:

```
yarn add moment
```

Instalação com o Npm

Para realizar a instalação com o Npm, podemos utilizar o comando **install**:

```
npm install moment --save
```

Ou de forma abreviada:

```
npm i moment --save
```

Também podemos utilizar outros gerenciadores e maneiras de instalação que podem ser consultados na documentação.

Não será por falta de instalação que você deixará de conhecê-la ou utilizá-la.

Começando a usar

Para começarmos a utilizar a MomentJS, devemos realizar o download de seu arquivo minificado (de preferência), como informado anteriormente, após ter o arquivo criado, precisaremos importá-lo em nossa página:

```
<script src="assets/js/moment.min.js" type="text/javascript"></script>
```

Utilizando a MomentJS

Pronto, já devemos ter tudo importado, configurado e instalado, portanto, já podemos começar a utilizar a MomentJS.

Para pegar a data atual, podemos simplesmente invocar a função **moment**. Ela irá devolver um objeto que funciona como um wrapper para o **Date** do JavaScript.

Pegando a data e hora atual

```
const agora = moment();
```

Seria a mesma coisa que realizarmos:

```
const agora = moment(new Date());
```

Ou, com JavaScript puro (Vanilla JS):

```
const agora = new Date();
```

Outras maneiras de obter a data atual, seria:

```
const agora = moment([]);
const agora = moment({});
```

Observação: Essas duas últimas maneiras funcionam apenas da versão 2.14.0 em diante, antes dela, o comportamento de ambos será diferente.

Passando uma data como parâmetro

> Mas, Matheus, eu queria poder passar uma data para a criação, isso é possível?

Sim. Para isso precisamos simplesmente passá-la como **String** para a função **moment**:

```
const dataNascimento = moment("1994-05-25");
```

Também podemos passar uma data com hora:

```
const dataNascimentoPrecisa = moment("1994-05-25 13:00");
```

Informando o formato da data

Também podemos passar uma data no formato que desejamos. Para isso, como segundo parâmetro devemos dizer qual parte da **String** representa os pedaços da data, ou seja, qual parte é o mês, dia ou ano. Em outras palavras, qual a formatação da data.

```
const dataNascimento = moment("25/05/1994", "DD/MM/YYYY");
```

> Epa, calma ai! Que doidera é essa de **DD/MM/YYYY?**

Esses são os formatos para representação de datas; cada um deles tem um significado:

- **D ou DD**: dia do mês de 1 a 31.
- **M ou MM**: mês do ano de 1 a 12.
- **YYYY:** ano com quatro (4) dígitos.

Além desses formatos, existem outros. Por exemplo:

- **YY:** ano com dois (2) dígitos.
- **MMM** ou **MMMM:** nome do mês por extenso.
- **DDD** ou **DDDD:** dia do ano de 1 à 365.

Para saber todos os formatos, consulte a documentação.

Assim como no primeiro exemplo, também podemos passar uma data com hora no formato que desejamos, lembrando apenas de informar a formatação:

```
const dataNascimentoPrecisa = moment("25/05/1994 13:00", "DD/MM/YYYY HH:mm");
```

Veja que para as horas também temos formatos, sendo eles:

- **H** ou **HH:** hora no formato 24, ou seja, de 00 a 23.
- **m** ou **mm:** minutos de 0 a 59.

Além dos formatos utilizados, também podemos contar com:

- **h** ou **hh**: hora no formato 12, ou seja, AM e PM.
- **s** ou **ss:** segundos de 0 a 59.
- **S**, **SS** ou **SSS:** segundo fracionados de 0 a 999.

Novamente, para saber todos os formatos, consulte a documentação.

> Mas como podemos garantir que a data passada está no mesmo formato que desejamos?

Veremos no próximo tópico!

Verificando se o formato está correto

O primeiro parâmetro deve respeitar a formatação do segundo. Para isso, como terceiro parâmetro podemos informar um **booleano**, dizendo para a Moment verificar se o formato combina exatamente com a data passada:

```
const dataNascimento = moment("25/05/1994", "DD/MM/YYYY", true);
```

Tudo certo, pois passamos a data no mesmo formato. E se passarmos uma data com hora?

```
const dataNascimento = moment("25/05/1994 13:00", "DD/MM/YYYY", true);
```

No segundo exemplo a data está correta? Não. Passamos uma data com hora, porém, na formatação informamos apenas a data sem hora, portanto, a mesma será uma data inválida.

> Tudo bem, Matheus, você falou que a data será inválida, mas como podemos confirmar isso?

Para verificar se uma data é inválida ou não, podemos utilizar a função **isValid**:

```
const dataNascimento = moment("25/05/1994 13:00", "DD/MM/YYYY", true);
console.log(dataNascimento.isValid());
```

Pronto. Será impresso **false** no nosso console. Para testarmos, vamos retirar a hora e passar apenas data:

```
const dataNascimento = moment("25/05/1994", "DD/MM/YYYY", true);
console.log(dataNascimento.isValid());
```

Agora como saída do console, temos **true**.

Informando um locale

Além dos parâmetros já mencionados anteriormente, também podemos passar um *locale* para nossa data, assim, a data pode utilizar o idioma inglês **en** ou português **pt.** O *locale* deve ser informado como terceiro parâmetro:

```
const data = moment("25/05/1994", "DD/MM/YYYY", "pt", true);
```

Como podemos ver, o parâmetro para verificação de formato mudou para o quarto, portanto, as formas (básicas) para criar uma data, são:

```
moment(String, String); // Data, Formato
moment(String, String, String); // Data, Formato, Locale
moment(String, String, Boolean); // Data, Formato, Verificar Formato
moment(String, String, String, Boolean); // Data, Formato, Locale, Verificar
Formato
```

Bom, agora já sabemos criar uma data de diversas maneiras (básicas). Para confirmar todas as maneiras, consulte a documentação.

O que mais podemos fazer com MomentJS?

Clonando data

Podemos realizar clones de data de uma maneira muito simples através da função **clone**:

```
const agora = moment();
const dataClonada = agora.clone();
```

Ou de forma mais simplista, podemos passar um objeto como parâmetro para o **moment**:

```
const agora = moment();
const dataClonada = moment(agora);
```

Assim, a data passada como parâmetro será uma nova data.

Pegando partes da data

Com toda certeza vamos precisar pegar (*get*) pedaços de uma determinada data. Para isso, temos as funções de **get**, sendo elas:

- **millisecond ou milliseconds**: pega os milisegundos de uma determinada data.
- **second ou seconds**: pega os segundos de uma determinada data.
- **minute ou minutes:** pega os minutos de uma determinada data.
- **hour ou hours:** pega a hora de uma determinada data.

- **date ou dates:** pega o dia de uma determinada data.
- **day ou days:** pega o dia da semana de uma determinada data.
- **month ou monthds:** pega o mês de uma determinada data.
- **year ou years:** pega o ano de uma determinada data.

Para saber todas as partes que podemos extrair de uma data, consulte a documentação.

```
const agora = moment();

console.log(agora.date()); // Imprimindo o dia
console.log(agora.month()); // Imprimindo o mês
console.log(agora.year()); // Imprimindo o ano
console.log(agora.hour()); // Imprimindo a hora
console.log(agora.minute()); // Imprimindo os minutos
console.log(agora.second()); // Imprimindo os segundos
```

Além dessa maneira, também podemos utilizar a função **get** e passar como parâmetro uma **String** referente ao nome da informação que queremos pegar:

```
const agora = moment();

console.log(agora.get("date")); // Imprimindo o dia
console.log(agora.get("month")); // Imprimindo o mês
console.log(agora.get("year")); // Imprimindo o ano
console.log(agora.get("hour")); // Imprimindo a hora
console.log(agora.get("minute")); // Imprimindo os minutos
console.log(agora.get("second")); // Imprimindo os segundos
```

Essas são as maneiras (simples) que podemos extrais informações e valores de nossas datas, mas e se surgir a necessidade de setar (set) alguma informação, como posso fazer?

Observação: Tanto para pegar, como para setar, o mês começa em 0. Ou seja, de 0 a 11, onde 0 é janeiro e 11 é Dezembro.

Setando valores para a data

O jeito mais simples de setar (*set*) algum valor em nossa data, é através das mesmas funções que utilizamos para pegar (*get*), fazendo uma pequena modificação. Devemos informar o valor que desejamos setar como parâmetro para as funções:

```
const agora = moment();

agora.date(25); // Setando o dia
agora.month(4); // Setando o mês
agora.year(1994); // Setando o ano
agora.hour(13); // Setando a hora
agora.minute(00); // Setando os minutos
agora.second(00); // Setando os segundos
```

Neste exemplo será criado um objeto com a data referente ao dia 25/05/1994, 13:00:00.

Além dessa maneira, também podemos utilizar a função set que recebe dois parâmetros:

- **Primeiro:** qual informação desejamos setar em nossa data, por exemplo: mês, dia, hora, etc
- **Segundo:** o valor que desejamos setar

```
const agora = moment();

agora.set("date", 25); // Setando o dia
agora.set("month", 4); // Setando o mês
agora.set("year", 1994); // Setando o ano
agora.set("hour", 13); // Setando a hora
agora.set("minute", 00); // Setando os minutos
agora.set("second", 00); // Setando os segundos
```

Esses são os exemplos que temos para pegar (get) e setar (set) valores em uma data.

Adicionando valores em uma data

Caso tenha a necessidade de adicionar valor em nossa data, podemos utilizar a função **add**:

```
const agora = moment();
agora.add(2, "year"); // também pode ser no plural, ou seja, years
```

No exemplo acima, foi adicionado dois anos para a data atual. Repare que a função recebe dois parâmetros, sendo eles:

- **Primeiro:** o valor que desejamos adicionar.
- **Segundo:** qual informação que desejamos adicionar

Em versões mais antigas da MomentJS, você pode ver os parâmetros em ordem contrária, porém, a forma contrária foi depreciada; ou seja, está em desuso.

Podemos utilizar alguns atalhos na digitação da informação, sendo eles:

Chave	Abreviação
years	y
months	M
days	d
hours	h
minutes	m
seconds	s

Esses são apenas alguns exemplos de atalhos (shorthands). Para visualizar todos, consulte a documentação.

A adição também pode ser intercalada com várias chamadas, por exemplo:

```
data.add(7, "d").add(1, "y").add(10, "minutes");
```

Veja que no exemplo acima adicionei 7 (sete) dias, 1 (um) ano e 10 (dez) minutos para um determianda data.

Retirando valores em uma data

Assim como podemos adicionar, também podemos subtrair utilizando a função **subtract**, seguindo a mesma ideia da função **add**:

```
const agora = moment();
agora.subtract(3, "hours");
```

Dessa maneira, estou retirando 3 (três) horas da data atual.

Formatando datas

Para formatar datas, podemos utilizar a função **format**, passando como parâmetro uma **String** onde uma formatação deve ser passada, seguindo a ideia já mencionada anteriormente durante a criação de datas. Caso tenha esquecido, abaixo vou listar algumas opções:

- **D ou DD**: dia do mês de 1 a 31.
- **M ou MM**: mês do ano de 1 a 12.
- **YYYY:** ano com quatro (4) dígitos.
- **YY:** ano com dois (2) dígitos.
- **MMM ou MMMM:** nome do mês por extenso.
- **DDD ou DDDD:** dia do ano de 1 a 365.
- **H ou HH:** hora no formato 24, ou seja, de 00 a 23.
- **m ou mm:** minutos de 0 a 59.
- **h ou hh:** hora no formato 12, ou seja, AM e PM.
- **s ou ss:** segundos de 0 a 59.
- **S, SS ou SSS**: segundo fracionados de 0 a 999.

Dessa maneira, para formatarmos uma data no formato que estamos acostumados, podemos fazer da seguinte maneira:

```
const agora = moment();
console.log(agora.format("DD/MM/YYYY HH:mm"));
```

No console, será impresso uma data e hora no formato brasileiro: **25/05/1994, 10:00**. Para saber todas as possíveis formatações que podem ser feitas, leia a documentação.

Bom, essas foram apenas algumas funções que a MomentJS oferece. Podemos ir muito mais além, mas o artigo se estenderia muito.

Gostou? Já conhecia a MomentJS? Não deixe de comentar e se inscrever na newsletter para receber novidades de artigos!