3 minutes

La Web API URL

La `Web API URL`

L'objet `URL` est un objet natif qui permet de gérer facilement des `URL` en ayant des propriétés et des méthodes adaptées.

Pour l'utiliser, rien de plus simple :

```
const url = new URL(url)
```

Les propriétés des objets `URL`

Les propriétés suivantes sont accessibles sur les objets `URL` et sont très pratiques :

`hash` : chaîne de caractères contenant un `#` suivi de l'identifiant du fragment de l'URL (exemple : `#test` dans `https://restapi.fr/#test`).

`hostname` : chaîne de caractères contenant le domaine (exemple : `restapi.fr` pour `https://restapi.fr:300/test`).

`host` : chaîne de caractères contenant l'hôte c'est à dire l'`hostname` et le `port` (exemple : `restapi.fr:3000` pour `https://restapi.fr:300/test`).

`href` : chaîne de caractères contenant l'`URL` entière (exemple : `https://restapi.fr:300/#test` pour `https://restapi.fr:300/#test`).

`origin` : chaîne de caractères contenant l'origine, c'est-à-dire le protocole, le domaine et le port (exemple : `https://restapi.fr:3000` pour `https://restapi.fr:3000/test`).

`pathname` : chaîne de caractères contenant le chemin (exemple : `/api/test` pour `https://restapi.fr:3000/api/test`).

`port` : chaîne de caractères contenant le port (exemple : `3000` pour `https://restapi.fr:3000/api/test`).

`protocol` : chaîne de caractères contenant le protocol suivi d'un `:` (exemple : `https:` pour `https://restapi.fr:3000/api/test`).

`search` : chaîne de caractères contenant les paramètres (exemple : `?user=paul&gender=m` pour `https://restapi.fr:3000/api/test?user=paul&gender=m`).

`password` : jamais utilisée en pratique. Contient le mot de passe si envoyé au format : `https://paul:123@restapi.fr:3000/` (ici `password` vaudra 123).

`username` : jamais utilisée en pratique. Contient le nom de l'utilisateur si envoyé au format : `https://paul:123@restapi.fr:3000/` (ici `username` vaudra paul).

La propriété `searchParams`

La propriété `searchParams` est particulière car elle ne retourne pas une chaîne de caractères mais un objet `URLSearchParams`.

Elle permet de créer des paramètres de recherche et de les encoder correctement.

Les `URL` doivent respecter le standard `RFC 3986` : les espaces, les caractères non latin et certains caractères spéciaux doivent être encodés en `UTF-8`, c'est-à-dire que le caractère est remplacé par le code `UTF-8`.

Par exemple un espace sera encodé en `%20` car son code `UTF-8` est 20 en hexadécimal.

Pour encoder un paramètre de recherche, il suffit de faire :

```
const url = new URL('https://ru.wikipedia.org/wiki/');

url.searchParams.set('param1', 'ъ');
```

Ici le caractère russe sera encodé correctement et nous aurons :

```
{
  href: "https://ru.wikipedia.org/wiki/?param1=%D1%8A"
  origin: "https://ru.wikipedia.org"
  protocol: "https:"
  username: ""
  password: ""
  host: "ru.wikipedia.org"
  hostname: "ru.wikipedia.org"
  port: ""
  pathname: "/wiki/"
  search: "?param1=%D1%8A"
  searchParams: URLSearchParams {}
```

```
  hash: ""
}
```

Remarquez que le caractère a été remplacé par `%D1%8A` car son code `UTF-8` est `D18A` en hexadécimal et que les `%` sont ajoutés pour l'encodage `URL-encoded UTF8` qui est l'encodage utilisé pour les `URLs`.

Méthodes disponibles sur l'objet `URLSearchParams`

Il n'y a pas que la méthode `set()` disponible sur ces objets. Nous pouvons retrouver les méthodes suivantes :

`append(nom, valeur)` permet d'ajouter le paramètre de recherche à l'`URL` avec le nom et la valeur spécifiés.

`set(nom, valeur)` permet d'ajouter, ou de remplacer s'il existe, le paramètre de recherche à l'`URL` avec le nom et la valeur spécifiés. Cela permet de s'assurer qu'il y a un seul paramètre avec le nom.

`delete(nom)` permet de supprimer le paramètre de recherche de l'`URL` avec le nom spécifié.

`get(nom)` permet de récupérer le premier paramètre de recherche de l'`URL` avec le nom spécifié.

`getAll(nom)` permet de récupérer tous les paramètres de recherche de l'`URL` ayant le nom spécifié.

`has(nom)` permet de vérifier si le paramètre de recherche ayant le nom spécifié est présent sur l'`URL`.

Voici un autre exemple :

```
const url = new URL('https://google.com/search');

url.searchParams.set('q', 'super recherche !');
```

Nous obtiendrons :

```
{
  href: "https://google.com/search?q=super+recherche+%21"
  origin: "https://google.com"
  protocol: "https:"
  username: ""
  password: ""
  host: "google.com"
  hostname: "google.com"
  port: ""
```

```
  pathname: "/search"
  search: "?q=super+recherche+%21"
  searchParams: URLSearchParams {}
  hash: ""
}
```

Remarquez que l'encodage des espaces a été fait avec des `+`, c'est une exception historique.

L'espace est le seul caractère pouvant être encodé par `+` et non par son code `UTF-8`. Il peut donc être encodé par `+` ou par `%20`.