URL

Встроенный класс **URL** предоставляет удобный интерфейс для создания и разбора URL-адресов.Нет сетевых методов, которые требуют именно объект URL, обычные строки вполне подходят. Так что, технически, мы не обязаны использовать URL. Но иногда он может быть весьма удобным. Синтаксис создания нового объекта URL:

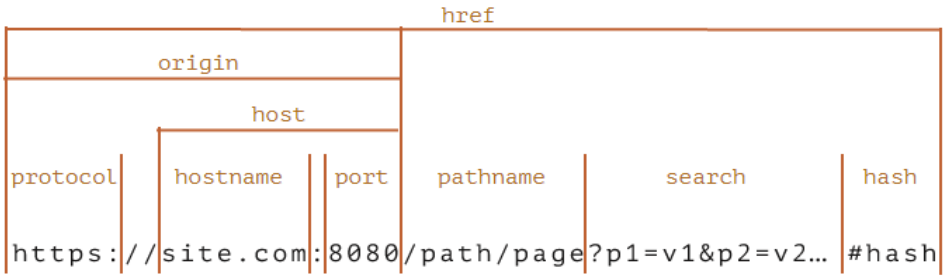
new URL(url, [base])

- **url** – полный URL-адрес или только путь, если указан второй параметр,
- **base** – необязательный «базовый» URL: если указан и аргумент url содержит только путь, то адрес будет создан относительно него (пример ниже).

```
1) let url1 = new URL('https://javascript.info/profile/admin');
let url2 = new URL('/profile/admin', 'https://javascript.info');
alert(url1); // https://javascript.info/profile/admin
alert(url2); // https://javascript.info/profile/admin

2) let url = new URL('https://javascript.info/profile/admin');
let newUrl = new URL('tester', url);
alert(newUrl); // https://javascript.info/profile/tester
```

Компоненты URL



- **href** это полный URL-адрес, то же самое, что url.toString()
- **protocol** – протокол, заканчивается символом двоеточия :
- **search** строка параметров, начинается с вопросительного знака ?
- **hash** начинается с символа #
- также есть свойства **user** и **password**, если используется HTTP-аутентификация: **http://login:password@site.com** (не нарисованы сверху, так как редко используются).

Мы можем использовать объект URL в методах fetch или XMLHttpRequest и почти во всех других, где ожидается URL-строка. Вообще, объект URL можно передавать почти куда угодно вместо строки, так как большинство методов сконвертируют объект в строку, при этом он станет строкой с полным URL-адресом.

Параметры в строке поиска должны быть правильно закодированы, чтобы они могли содержать не-латинские буквы, пробелы и т.п. Для этого есть свойство **url.searchParams** – объект типа **URLSearchParams**. Он предоставляет удобные методы для работы с параметрами:

- **append(name, value)** – добавить параметр по имени,
- **delete(name)** – удалить параметр по имени,
- **get(name)** – получить параметр по имени,
- **getAll(name)** – получить все параметры с одинаковым именем name (такое возможно, например: ?user=John&user=Pete),
- **has(name)** – проверить наличие параметра по имени,
- **set(name, value)** – задать/заменить параметр,
- **sort()** – отсортировать параметры по имени, используется редко,
- …и является **перебираемым**, аналогично Map.

```
let url = new URL('https://google.com/search');
url.searchParams.set('q', 'test me!'); // добавим параметр, содержащий пробел и !
alert(url); // https://google.com/search?q=test+me%21

url.searchParams.set('tbs', 'qdr:y'); // параметр с двоеточием :
// параметры автоматически кодируются
alert(url); // https://google.com/search?query=test+me%21&tbs=qdr%3Ay

// перебрать параметры (в исходном виде)
for(let [name, value] of url.searchParams) {
  alert(`${name}=${value}`); // q=test me!, далее tbs=qdr:y
}
```

Кодирование

Существует стандарт RFC3986, который определяет список разрешённых и запрещённых символов в URL.Запрещённые символы, например, **нелатинские буквы и пробелы**, должны быть закодированы – заменены соответствующи ми кодами UTF-8 с префиксом %, например: **%20** (пробел в URL-адресе). К счастью, **объекты URL** делают всё это **автоматически**. Мы просто указываем параметры в обычном, незакодированном, виде, а затем конвертируем URL в строку: Каждая кириллическая буква представляется двумя байтами в кодировке UTF-8

Раньше, до того как появились объекты URL, люди использовали для URL-адресов обычные **строки**. Сейчас URL часто удобнее, но строки всё ещё можно использовать. Во многих случаях код с ними короче. Однако, если мы используем строку, то надо самим позаботиться о кодировании специальных символов. Для этого есть встроенные функции:

- encodeURI – кодирует URL-адрес целиком.
- decodeURI – декодирует URL-адрес целиком.
- encodeURIComponent – кодирует компонент URL, например, параметр, хеш, имя пути и т.п.
- decodeURIComponent – декодирует компонент URL.

Какая разница между encodeURIComponent и encodeURI? В URL-адресе разрешены символы :, ?, =, &, #. …С другой стороны, если взглянуть на один компонент, например, URL-параметр, то в нём такие символы должны быть закодированы, чтобы не поломать форматирование. **encodeURI** кодирует только символы, полностью запрещённые в URL.
encodeURIComponent кодирует эти же символы плюс, в дополнение к ним, символы **#, $, &, +, ,, /, :, ;, =, ? и @.**

Так что для URL целиком можно использовать encodeURI:

```
let url = encodeURI('http://site.com/привет');
alert(url); // http://site.com/%D0%BF%D1%80%D0%B8%D0%B2%D0%B5%D1%82
```

…А для параметров лучше будет взять encodeURIComponent:

```
let music = encodeURIComponent('Rock&Roll');
let url = `https://google.com/search?q=${music}`;
alert(url); // https://google.com/search?q=Rock%26Roll
```

Классы URL и URLSearchParams базируются на последней спецификации URI, описывающей устройство адресов: RFC3986, в то время как функции encode* – на устаревшей версии стандарта RFC2396. Различий мало, но они есть, например, по-разному кодируются адреса IPv6

```
// допустимый URL-адрес IPv6
let url = 'http://[2607:f8b0:4005:802::1007]/';
alert(encodeURI(url)); // http://%5B2607:f8b0:4005:802::1007%5D/
alert(new URL(url)); // http://[2607:f8b0:4005:802::1007]/
```