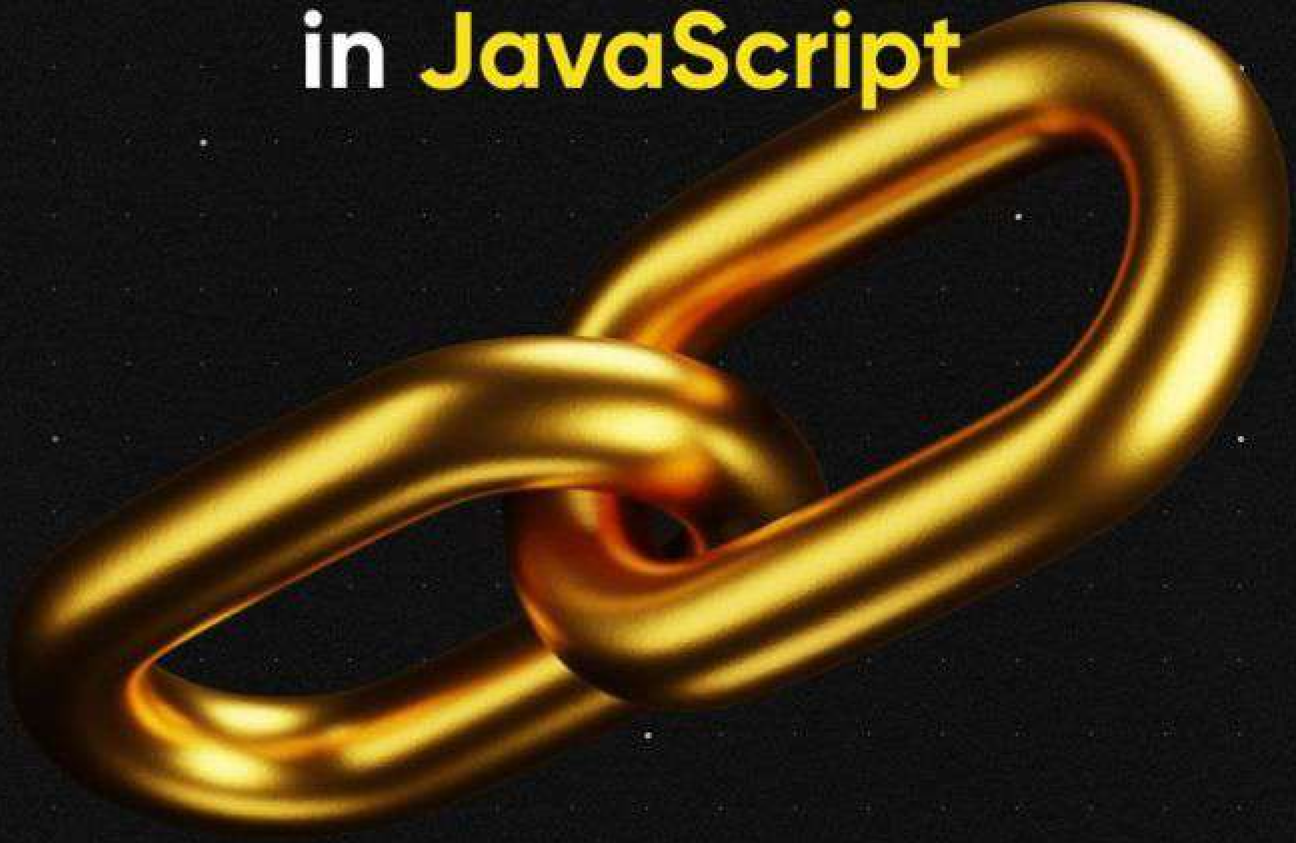
JS

# Working with URLs

in JavaScript



patreon.com/CodewithSloba

Swipe ⟶

## 1. Creating and Parsing URLs

Use the URL constructor to create or parse URLs easily.

```
const url = new URL('https://example.com/path?name=shravan');
console.log(url.hostname); // example.com
console.log(url.searchParams.get('name')); // shravan
```

## 2. Manipulating Query Parameters

Access, add, or modify query parameters with `URLSearchParams`.

```
const url = new URL('https://example.com?name=shravan');
url.searchParams.set('age', '24');
console.log(url.toString()); // https://example.com/?
name=shravan&age=24
```

Swipe ⟶

## 3. Extracting URL Components

Retrieve parts like protocol, pathname, and hash.

```
const url = new URL('https://example.com/path#section');
console.log(url.protocol); // https:
console.log(url.pathname); // /path
console.log(url.hash); // #section
```

## 4. Encoding & Decoding URLs

Use `encodeURIComponent` and `decodeURIComponent` to handle special characters.

```
const query = 'name=shravan&city=some space';
const encoded = encodeURIComponent(query);
console.log(encoded); //
name%3Dshravan%26city%3Dsome%20space
console.log(decodeURIComponent(encoded)); //
name=shravan&city=some space
```

Swipe ⟶

## 5. Handling Relative URLs

Use the URL constructor to resolve relative URLs. javascript.

```javascript
const base = new
URL('https://example.com/base/');
const relative = new URL('subpath', base);
console.log(relative.toString()); //
https://example.com/base/subpath
```

## 6. Checking URL Validity

Validate a URL with a try-catch block.

```javascript
try {
  const url = new URL('invalid-url');
} catch (error) {
  console.log('Invalid URL'); // Invalid
URL
}
```

## 7. Constructing URLs Dynamically

Build dynamic URLs based on user input or configurations.

```
const base = 'https://example.com';
const endpoint = '/api';
const query = { id: '123', name: 'shravan' };

const url = new URL(`${base}${endpoint}`);
Object.entries(query).forEach(([key, value]) =>
{
  url.searchParams.set(key, value);
});
console.log(url.toString()); //
https://example.com/api?id=123&name=shravan
```

## 8. Fetching Data with URLs

Combine URL and fetch for making API calls.

```
const url = new
URL('https://api.example.com/data');
url.searchParams.set('id', '123');
fetch(url)
  .then(response => response.json())
  .then(data => console.log(data));
```

Swipe ⟶

## 9. Shortening Long URLs

Create shorter URLs using third-party services or libraries.

```
const longURL =
'https://example.com/super/long/path?
with=query&parameters=true';
// Use APIs like bit.ly for URL shortening
```

## 10. URL Path Normalization

Use libraries like path or URL APIs for consistent paths.

```
const url = new
URL('https://example.com//a/../b');
console.log(url.pathname); // /b
```

Support me on

**PATREON**

patreon.com/CodewithSloba