JavaScript Fetch API Cheat Sheet

1. GET - Fetch data

```
fetch('https://api.example.com/resource')
  .then(response => response.json())
  .then(data => console.log("GET:", data))
  .catch(error => console.error("GET Error:", error));
```

2. POST - Create new data

```
fetch('https://api.example.com/resource', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    name: 'New Item',
    value: 123
  })
})
  .then(response => response.json())
  .then(data => console.log("POST:", data))
  .catch(error => console.error("POST Error:", error));
```

3. PUT - Update existing data

```
fetch('https://api.example.com/resource/1', {
  method: 'PUT',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    name: 'Updated Item',
    value: 999
  })
})
  .then(response => response.json())
  .then(data => console.log("PUT:", data))
  .catch(error => console.error("PUT Error:", error));
```

4. DELETE - Remove data

```
fetch('https://api.example.com/resource/1', {
  method: 'DELETE'
})
  .then(response => {
    if (response.ok) {
      console.log("DELETE: Success");
    } else {
      console.log("DELETE: Failed");
    }
  })
```

JavaScript Fetch API Cheat Sheet

```
  .catch(error => console.error("DELETE Error:", error));
```

5. Reusable Template Function

```
function apiRequest(url, method = 'GET', data = null) {
  return fetch(url, {
    method,
    headers: {
      'Content-Type': 'application/json'
    },
    body: data ? JSON.stringify(data) : null
  })
    .then(res => res.json())
    .catch(err => console.error(\`\${method} Error:\`, err));
}

// Usage examples:
apiRequest('https://api.example.com/items'); // GET
apiRequest('https://api.example.com/items', 'POST', { name: 'Item' }); // POST
```