Resurse suplimentare
pe care recomand să le parcurgi

Sintaxa de bază a Fetch

```
fetch(url)
  .then((response) => response.json())
  .then((data) => console.log(data))
  .catch((error) => console.error('Error:', error))
```

Cu async/await (recomandat):

```
async function fetchData() {
  try {
    const response = await fetch(url)
    const data = await response.json()
    console.log(data)
  } catch (error) {
    console.error('Error:', error)
  }
}
```

Cereri GET simple

Cererea GET este utilizată pentru a obține date de la un server:

```javascript
// Cerere GET simplă către JSONPlaceholder
async function getUsers() {
  try {
    const response = await fetch('<https://jsonplaceholder.typicode.com/users>')

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`)
    }

    const users = await response.json()
    console.log('Utilizatori:', users)
    return users
  } catch (error) {
    console.error('Eroare la obținerea utilizatorilor:', error)
  }
}

// Apelarea funcției
getUsers()
```

Obținerea unui singur utilizator:

```javascript
async function getUser(userId) {
  try {
    const response = await fetch(
      `https://jsonplaceholder.typicode.com/users/${userId}`
    )

    if (!response.ok) {
      throw new Error(`User not found! status: ${response.status}`)
    }

    const user = await response.json()
    console.log('Utilizator:', user)
    return user
  } catch (error) {
    console.error('Eroare:', error)
  }
}

// Obținerea utilizatorului cu ID-ul 1
getUser(1)
```

Cereri POST pentru trimiterea datelor

Cererea POST este utilizată pentru a trimite date către server:

```
async function createPost(title, body, userId) {
  try {
    const response = await fetch('<https://jsonplaceholder.typicode.com/posts>', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        title: title,
        body: body,
        userId: userId,
      }),
    })

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`)
    }

    const newPost = await response.json()
    console.log('Post creat:', newPost)
    return newPost
  } catch (error) {
    console.error('Eroare la crearea post-ului:', error)
  }
}

// Crearea unui post nou
createPost('Titlul meu', 'Conținutul post-ului', 1)
```

Cereri PUT pentru actualizarea datelor

```javascript
async function updatePost(postId, title, body, userId) {
  try {
    const response = await fetch(
      `https://jsonplaceholder.typicode.com/posts/${postId}`,
      {
        method: 'PUT',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          id: postId,
          title: title,
          body: body,
          userId: userId,
        }),
      }
    )

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`)
    }

    const updatedPost = await response.json()
    console.log('Post actualizat:', updatedPost)
    return updatedPost
  } catch (error) {
    console.error('Eroare la actualizarea post-ului:', error)
  }
}

// Actualizarea post-ului cu ID-ul 1
updatePost(1, 'Titlu actualizat', 'Conținut actualizat', 1)
```

Cereri DELETE pentru ștergerea datelor

```javascript
async function deletePost(postId) {
  try {
    const response = await fetch(
      `https://jsonplaceholder.typicode.com/posts/${postId}`,
      {
        method: 'DELETE',
      }
    )

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`)
    }

    console.log(`Post cu ID-ul ${postId} a fost șters`)
    return true
  } catch (error) {
    console.error('Eroare la ștergerea post-ului:', error)
    return false
  }
}

// Ștergerea post-ului cu ID-ul 1
deletePost(1)
```

Gestionarea răspunsurilor și erorilor

Verificarea statusului răspunsului:

```javascript
async function fetchWithErrorHandling(url) {
  try {
    const response = await fetch(url)

    // Verifică dacă cererea a fost reușită
    if (!response.ok) {
      throw new Error(`HTTP ${response.status}: ${response.statusText}`)
    }

    // Verifică tipul de conținut
    const contentType = response.headers.get('content-type')
    if (!contentType || !contentType.includes('application/json')) {
      throw new Error('Răspunsul nu este JSON valid')
    }

    const data = await response.json()
    return data
  } catch (error) {
    if (error.name === 'TypeError') {
      console.error('Eroare de rețea:', error.message)
    } else {
      console.error('Eroare:', error.message)
    }
    throw error
  }
}
```

Gestionarea diferitelor tipuri de răspunsuri:

```
async function handleResponse(response) {
  const contentType = response.headers.get('content-type')

  if (contentType && contentType.includes('application/json')) {
    return await response.json()
  } else if (contentType && contentType.includes('text/')) {
    return await response.text()
  } else {
    return await response.blob()
  }
}
```

Headers și configurări avansate

```
async function fetchWithHeaders() {
  try {
    const response = await fetch('<https://api.example.com/data>', {
      method: 'GET',
      headers: {
        Authorization: 'Bearer your-token-here',
        Accept: 'application/json',
        'User-Agent': 'MyApp/1.0',
      },
      // Alte opțiuni
      mode: 'cors',
      cache: 'default',
      credentials: 'same-origin',
    })

    const data = await response.json()
    return data
  } catch (error) {
    console.error('Eroare:', error)
  }
}
```

Timeout pentru cereri

```
function fetchWithTimeout(url, options = {}, timeout = 5000) {
  return Promise.race([
    fetch(url, options),
    new Promise((_, reject) =>
      setTimeout(() => reject(new Error('Request timeout')), timeout)
    ),
  ])
}

// Utilizare
async function example() {
  try {
    const response = await fetchWithTimeout(
      '<https://api.example.com/slow-endpoint>',
      {},
      3000
    )
    const data = await response.json()
    console.log(data)
  } catch (error) {
    if (error.message === 'Request timeout') {
      console.error('Cererea a expirat')
    } else {
      console.error('Altă eroare:', error)
    }
  }
}
```

Lucrul cu Query Parameters

GENERAȚIA TECH

```
function buildURLWithParams(baseURL, params) {
  const url = new URL(baseURL)
  Object.keys(params).forEach((key) => {
    url.searchParams.append(key, params[key])
  })
  return url.toString()
}

async function searchPosts(query, limit = 10) {
  const params = {
    q: query,
    _limit: limit,
  }

  const url = buildURLWithParams(
    '<https://jsonplaceholder.typicode.com/posts>',
    params
  )

  try {
    const response = await fetch(url)
    const posts = await response.json()
    return posts
  } catch (error) {
    console.error('Eroare la căutare:', error)
  }
}

// Căutarea post-urilor
searchPosts('JavaScript', 5)
```