

# Tutoriel API CoinCap

Exploitez les données des cryptomonnaies



## Auteurs :

Lepine François  
Borry Lenny  
Ghrib Yacine  
Veau-Bigot Damien



21 mars 2025













## Table des matières

<b>1</b>	<b>Introduction a l'API CoinCap</b>	<b>2</b>
<b>2</b>	<b>Prerequis</b>	<b>2</b>
<b>3</b>	<b>Points de terminaison de l'API CoinCap</b>	<b>2</b>
3.1	Endpoints principaux . . . . .	3
3.2	Exemples d'utilisation . . . . .	4
3.2.1	Exemple en JavaScript . . . . .	4
3.2.2	Exemple en Python . . . . .	5
3.2.3	Exemple en Angular . . . . .	6
3.2.4	Exemple en Angular . . . . .	7
3.3	Recuperer les taux de change pour les monnaies fiduciaires . . . . .	8
3.4	Recuperer des donnees OHLCV (bougies) . . . . .	8
<b>4</b>	<b>Authentification et limites de taux</b>	<b>9</b>
4.1	Limites de taux . . . . .	9
4.2	Authentification par cle API . . . . .	9
<b>5</b>	<b>Bonnes pratiques et conseils</b>	<b>10</b>
5.1	Gestion des limites de taux . . . . .	10
5.2	Gestion des erreurs . . . . .	11
5.3	Mise en cache des donnees . . . . .	11
<b>6</b>	<b>Exemples d'application</b>	<b>13</b>
6.1	Convertisseur de cryptomonnaies . . . . .	13
6.2	Dashboard de portfolio . . . . .	16
<b>7</b>	<b>Conclusion</b>	<b>19</b>

## 1 Introduction a l'API CoinCap

L'API CoinCap est un outil puissant pour accéder a des données en temps réel et historiques sur les cryptomonnaies. Que vous soyez un développeur, un analyste ou un passionné de crypto, cette API vous permet de récupérer des informations telles que les prix, les capitalisations boursières, les volumes d'échange, et bien plus encore.

**Lien de la documentation officielle :** <https://docs.coincap.io/>

MARKET CAP \$2.89T		EXCHANGE VOL \$73.21B		ASSETS 2,319		EXCHANGES 123		MARKETS 5,410		BTC DOM INDEX 61.1%	
Rank	Name	Price	Market Cap	VWAP (24hr)	Supply	Volume (24hr)	Change (24hr)				
1	 Bitcoin BTC	\$88,954.40	\$1.76t	\$88,666.58	19.83m	\$16.96b	-1.35%				
2	 Ethereum ETH	\$2,177.22	\$262.47b	\$2,197.00	120.60m	\$6.71b	-4.12%				
3	 XRP XRP	\$2.51	\$145.51b	\$2.54	57.95b	\$2.68b	-2.79%				
4	 Tether USDT	\$1.00	\$142.89b	\$0.99	142.80b	\$30.34b	0.01%				
5	 BNB BNB	\$598.90	\$86.29b	\$596.01	144.01m	\$298.45m	-0.30%				
6	 Solana SOL	\$142.58	\$72.49b	\$143.64	508.71m	\$1.66b	-4.33%				
7	 USDC USDC	\$1.00	\$57.46b	\$1.00	57.45b	\$5.04b	0.02%				
8	 Cardano ADA	\$0.86	\$30.61b	\$0.88	35.22b	\$1.32b	-6.59%				
9	 Dogecoin DOGE	\$0.20	\$30.13b	\$0.20	148.33b	\$683.54m	-2.16%				
10	 TRON TRX	\$0.24	\$20.96b	\$0.24	86.05b	\$283.12m	1.18%				
11	 Lido Staked ETH STETH	\$2,176.62	\$20.42b	\$2,197.19	9.38m	\$9.02m	-4.40%				
12	 Wrapped Bitcoin WBTC	\$88,830.94	\$11.47b	\$89,222.96	129.18k	\$82.29m	-1.50%				

## 2 Prerequis

Pour utiliser l'API CoinCap, vous aurez besoin de :

- Un langage de programmation (JavaScript, Python, Angular, etc.)
- Une bibliothèque pour effectuer des requêtes HTTP (comme `fetch` ou `axios` en JavaScript, `requests` en Python, `HttpClient` en Angular)
- Une connaissance de base des API REST et du format JSON

## 3 Points de terminaison de l'API CoinCap

L'API CoinCap offre plusieurs points de terminaison (endpoints) pour accéder a différents types de données. L'URL de base de l'API est <https://api.coincap.io/v2/>.

### 3.1 Endpoints principaux

#### Points de terminaison principaux

1	GET /assets	# Liste des cryptomonnaies
2	GET /assets/{id} specifique	# Details d'une cryptomonnaie
3	GET /assets/{id}/history cryptomonnaie	# Historique des prix d'une
4	GET /assets/{id}/markets echangee	# Marches ou la cryptomonnaie est
5	GET /rates fiduciaires	# Taux de change pour les monnaies
6	GET /exchanges	# Liste des exchanges
7	GET /markets exchanges	# Paires de trading sur les
8	GET /candles cryptomonnaies	# Donnees OHLCV (bougies) pour les

## 3.2 Exemples d'utilisation

### 3.2.1 Exemple en JavaScript

#### Exemple en JavaScript (Marches pour Bitcoin)

```
1  async function getCryptoMarkets(cryptoId) {
2    try {
3      const response = await fetch('https://api.coincap.io/v2/
      assets/${cryptoId}/markets?limit=10');
4      const data = await response.json();
5
6      if (data.data) {
7        console.log('Principaux marches pour ${cryptoId}:');
8        data.data.forEach(market => {
9          console.log(`${market.exchangeId}: ${market.baseSymbol
          }/${market.quoteSymbol}`);
10         console.log('  Prix: ${parseFloat(market.priceUsd).
          toFixed(2)}');
11         console.log('  Volume (24h): ${parseInt(market.
          volumeUsd24Hr).toLocaleString()}');
12         console.log('  Pourcentage du volume: ${parseFloat(
          market.volumePercent).toFixed(2)}%');
13         console.log('---');
14       });
15     }
16   } catch (error) {
17     console.error('Erreur lors de la recuperation des marches
      :', error);
18   }
19 }
20
21 getCryptoMarkets('bitcoin');
```

### 3.2.2 Exemple en Python

#### Exemple en Python (Marches pour Bitcoin)

```
1 import requests
2
3 def get_crypto_markets(crypto_id):
4     try:
5         response = requests.get(
6             f'https://api.coincap.io/v2/assets/{crypto_id}/
              markets',
7             params={'limit': 10}
8         )
9
10        markets = response.json()['data']
11
12        print(f"Principaux_marches_pour_{crypto_id}:")
13        for market in markets:
14            print(f"{market['exchangeId']}:_{market['
              baseSymbol']}/{market['quoteSymbol']}")
15            print(f"    _Prix:_{float(market['priceUsd']):.2f}")
16            print(f"    _Volume_(24h):_{int(float(market['
              volumeUsd24Hr'])):,}")
17            print(f"    _Pourcentage_du_volume:_{float(market['
              volumePercent']):.2f}%")
18            print("---")
19
20        except Exception as e:
21            print(f"Erreur_lors_de_la_recuperation_des_marches:{e
              }")
22
23 get_crypto_markets('bitcoin')
```

### 3.2.3 Exemple en Angular

#### Exemple en Angular (Marches pour Bitcoin)

```
1 import { Component, OnInit } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { ActivatedRoute } from '@angular/router';
4
5 interface Market {
6   exchangeId: string;
7   baseSymbol: string;
8   quoteSymbol: string;
9   priceUsd: string;
10  volumeUsd24Hr: string;
11  volumePercent: string;
12 }
13
14 @Component({
15   selector: 'app-crypto-markets',
16   template: `
17     <div *ngIf="loading">Chargement des marches...</div>
18     <div *ngIf="error">Erreur: {{error}}</div>
19     <div *ngIf="markets.length">
20       <h3>Principaux marches</h3>
21       <div class="market-card" *ngFor="let market of markets">
22         <h4>{{market.exchangeId}}: {{market.baseSymbol}}/{{
23           market.quoteSymbol}}</h4>
24         <p>Prix: ${{parseFloat(market.priceUsd).toFixed(2)}}</
25           p>
26         <p>Volume (24h): ${{formatVolume(market.volumeUsd24Hr)
27           }}</p>
28         <p>Pourcentage du volume: {{parseFloat(market.
29           volumePercent).toFixed(2)}}%</p>
30       </div>
31     </div>
32   `,
33   styles: [`
34     .market-card {
35       border: 1px solid #ddd;
36       border-radius: 4px;
37       padding: 15px;
38       margin-bottom: 15px;
39       background-color: #f9f9f9;
40     }
41     h4 {
42       margin-top: 0;
43       color: #333;
44     }
45   `]
46 })
```

### 3.2.4 Exemple en Angular

#### Exemple en Angular 2 (Marches pour Bitcoin)

```
1 export class CryptoMarketsComponent implements OnInit {
2   markets: Market[] = [];
3   loading = true;
4   error = '';
5
6   constructor(
7     private http: HttpClient,
8     private route: ActivatedRoute
9   ) { }
10
11   ngOnInit(): void {
12     const cryptoId = this.route.snapshot.paramMap.get('id');
13
14     if (cryptoId) {
15       this.http.get<any>('https://api.coincap.io/v2/assets/${
16         cryptoId}/markets?limit=10')
17         .subscribe(
18           response => {
19             this.markets = response.data;
20             this.loading = false;
21           },
22           error => {
23             this.error = error.message;
24             this.loading = false;
25           }
26         );
27     } else {
28       this.error = 'ID de cryptomonnaie manquant';
29       this.loading = false;
30     }
31
32   }
33
34   parseFloat(value: string): number {
35     return parseFloat(value);
36   }
37
38   formatVolume(value: string): string {
39     return parseInt(value).toLocaleString();
40   }
41 }
```



**Exemple en Angular 3 (Marches pour Bitcoin)**

```
1
2 % Section 8 : Autres endpoints utiles
3 \section{Autres endpoints utiles}
4
5 \subsection{Recuperer les informations sur les exchanges}
6
7 \textbf{Endpoint :} \texttt{https://api.coincap.io/v2/}
8 \texttt{exchanges}
9
10 \begin{tcolorbox}[colback=codebackground, colframe=primary,
11 title=Exemple de requête, width=\textwidth]
12 \begin{lstlisting}[language=JavaScript]
13 fetch('https://api.coincap.io/v2/exchanges?limit=10')
14   .then(response => response.json())
15   .then(data => console.log(data.data))
16   .catch(error => console.error('Erreur:', error));
17 \end{lstlisting}
18 \end{tcolorbox}
```

**3.3 Recuperer les taux de change pour les monnaies fiduciaires****Endpoint :** `https://api.coincap.io/v2/rates`**Exemple de requête**

```
1 fetch('https://api.coincap.io/v2/rates')
2   .then(response => response.json())
3   .then(data => console.log(data.data))
4   .catch(error => console.error('Erreur:', error));
```

**3.4 Recuperer des donnees OHLCV (bougies)****Endpoint :** `https://api.coincap.io/v2/candles`**Parametres requis**

```
1 exchange # ID de l'exchange (ex: binance)
2 interval # Intervalle de temps (m1, m5, m15, m30, h1, h2, h6,
3           h12, d1)
4 baseId   # ID de la cryptomonnaie de base (ex: bitcoin)
5 quoteId  # ID de la cryptomonnaie de quote (ex: tether)
```

## Exemple de requête

```
1  const params = new URLSearchParams({
2    exchange: 'binance',
3    interval: 'h1',
4    baseId: 'bitcoin',
5    quoteId: 'tether'
6  });
7
8  fetch('https://api.coincap.io/v2/candles?${params.toString()}')
9    .then(response => response.json())
10   .then(data => console.log(data.data))
11   .catch(error => console.error('Erreur:', error));
```

## 4 Authentification et limites de taux

### 4.1 Limites de taux

L'API CoinCap a les limites de taux suivantes :

- **Sans authentification** : 200 requêtes par minute
- **Avec authentification** : Jusqu'à 500 requêtes par minute (selon votre forfait)

### 4.2 Authentification par cle API

Pour obtenir une cle API, vous devez vous inscrire sur le site CoinCap. Une fois inscrit, vous pouvez utiliser votre cle API dans l'en-tête HTTP Authorization :

## Exemple d'authentification en JavaScript

```
1  const apiKey = 'votre-cle-api-coincap';
2
3  fetch('https://api.coincap.io/v2/assets', {
4    headers: {
5      'Authorization': 'Bearer ${apiKey}'
6    }
7  })
8    .then(response => response.json())
9    .then(data => console.log(data.data))
10   .catch(error => console.error('Erreur:', error));
```

### Exemple d'authentification en Python

```
1 import requests
2
3 api_key = 'votre-cle-api-coincap'
4 headers = {
5     'Authorization': f'Bearer {api_key}'
6 }
7
8 response = requests.get('https://api.coincap.io/v2/assets',
9     headers=headers)
10 data = response.json()['data']
11 print(data)
```

### Exemple d'authentification en Angular

```
1 import { HttpClient, HttpHeaders } from '@angular/common/http';
2
3 import { Injectable } from '@angular/core';
4 import { Observable } from 'rxjs';
5 import { map } from 'rxjs/operators';
6
7 @Injectable({
8     providedIn: 'root'
9 })
10 export class CoincapService {
11     private baseUrl = 'https://api.coincap.io/v2';
12     private apiKey = 'votre-cle-api-coincap';
13
14     constructor(private http: HttpClient) { }
15
16     getAssets(limit: number = 20): Observable<any[]> {
17         const headers = new HttpHeaders({
18             'Authorization': 'Bearer ${this.apiKey}'
19         });
20
21         return this.http.get<any>(`${this.baseUrl}/assets?limit=${limit}`, { headers })
22             .pipe(map(response => response.data));
23     }
24 }
```

## 5 Bonnes pratiques et conseils

### 5.1 Gestion des limites de taux

Pour éviter de dépasser les limites de taux, adoptez ces bonnes pratiques :

- Mettez en cache les données qui ne changent pas fréquemment

- Limitez la fréquence des requêtes, surtout pour les données en temps réel
- Implementez une logique de backoff exponentiel en cas d'erreurs 429 (Too Many Requests)
- Utilisez des requêtes batch lorsque possible plutôt que de multiples requêtes individuelles

## 5.2 Gestion des erreurs

Assurez-vous de gérer correctement les erreurs de l'API :

### Exemple de gestion d'erreurs en JavaScript

```
1  async function fetchWithErrorHandling(url) {
2    try {
3      const response = await fetch(url);
4
5      if (!response.ok) {
6        if (response.status === 429) {
7          console.error('Limite de taux depassee. Reessayez plus
8            tard.');
```

```
          // Logique de backoff exponentiel ici
9        } else {
10         console.error('Erreur HTTP: ${response.status}');
11       }
12       return null;
13     }
14
15     return await response.json();
16   } catch (error) {
17     console.error('Erreur reseau:', error);
18     return null;
19   }
20 }
21
22 fetchWithErrorHandling('https://api.coincap.io/v2/assets')
23   .then(data => {
24     if (data) {
25       console.log(data.data);
26     }
27   });
```

## 5.3 Mise en cache des données

Mettez en cache les données pour réduire les appels API et améliorer les performances :

## Exemple de mise en cache en JavaScript

```
1 class CoinCapCache {
2   constructor(ttl = 60000) { // Temps de vie du cache en ms (1
      minute par défaut)
3     this.cache = {};
4     this.ttl = ttl;
5   }
6   async get(endpoint) {
7     const now = Date.now();
8     const cacheKey = endpoint;
9     // Verifier si les donnees sont dans le cache et toujours
      valides
10    if (this.cache[cacheKey] && now < this.cache[cacheKey].
        expiry) {
11      console.log('Donnees recuperees du cache');
12      return this.cache[cacheKey].data;
13    }
14
15    // Recuperer les donnees fra ches de l'API
16    console.log('Recuperation des donnees depuis l\'API');
17    try {
18      const response = await fetch('https://api.coincap.io/v2/
        ${endpoint}');
19      const result = await response.json();
20
21      // Stocker dans le cache
22      this.cache[cacheKey] = {
23        data: result,
24        expiry: now + this.ttl
25      };
26
27      return result;
28    } catch (error) {
29      console.error('Erreur lors de la recuperation des
        donnees:', error);
30      throw error;
31    }
32  }
33
34  clearCache() {
35    this.cache = {};
36  }
37 }
38 // Utilisation
39 const coinCapCache = new CoinCapCache(5 * 60 * 1000); // Cache
      de 5 minutes
40
41 coinCapCache.get('assets?limit=10')
42   .then(data => console.log(data.data))
43   .catch(error => console.error(error));
```

## 6 Exemples d'application

### 6.1 Convertisseur de cryptomonnaies

Creez un convertisseur qui permet aux utilisateurs de convertir entre différentes cryptomonnaies et monnaies fiduciaires.

#### Exemple de convertisseur en JavaScript

```
1 class CryptoConverter {
2   constructor() {
3     this.cryptoData = {};
4     this.ratesData = {};
5   }
6
7   async initialize() {
8     try {
9       // Recuperer les donnees des cryptomonnaies
10      const cryptoResponse = await fetch('https://api.coincap.io/v2/assets?limit=100');
11      const cryptoResult = await cryptoResponse.json();
12
13      // Recuperer les taux de change des monnaies fiduciaires
14      const ratesResponse = await fetch('https://api.coincap.io/v2/rates');
15      const ratesResult = await ratesResponse.json();
16
17      // Indexer les donnees pour un acces facile
18      cryptoResult.data.forEach(crypto => {
19        this.cryptoData[crypto.id] = crypto;
20      });
21
22      ratesResult.data.forEach(rate => {
23        this.ratesData[rate.id] = rate;
24      });
25
26      console.log('Convertisseur initialise avec succes');
27      return true;
28    } catch (error) {
29      console.error('Erreur lors de l\'initialisation du
30        convertisseur:', error);
31      return false;
32    }
33
34    convert(amount, fromCurrency, toCurrency) {
35      if (!amount || !fromCurrency || !toCurrency) {
36        return { success: false, error: 'Parametres manquants' };
37      }
38    }
39  }
```

## Exemple de convertisseur en JavaScript

```
1      try {
2          amount = parseFloat(amount);
3
4          // Convertir en USD d'abord
5          let usdValue;
6
7          if (this.cryptoData[fromCurrency]) {
8              // De crypto a USD
9              usdValue = amount * parseFloat(this.cryptoData[
10                  fromCurrency].priceUsd);
11          } else if (this.ratesData[fromCurrency]) {
12              // De fiat a USD
13              usdValue = amount / parseFloat(this.ratesData[
14                  fromCurrency].rateUsd);
15          } else {
16              return { success: false, error: 'Devise source
17                  inconnue' };
18          }
19
20          // Convertir de USD a la devise cible
21          let targetValue;
22
23          if (this.cryptoData[toCurrency]) {
24              // De USD a crypto
25              targetValue = usdValue / parseFloat(this.cryptoData[
26                  toCurrency].priceUsd);
27          } else if (this.ratesData[toCurrency]) {
28              // De USD a fiat
29              targetValue = usdValue * parseFloat(this.ratesData[
30                  toCurrency].rateUsd);
31          } else {
32              return { success: false, error: 'Devise cible inconnue
33                  ' };
34          }
35      }
```

## Exemple de convertisseur en JavaScript

```
1      return {
2          success: true,
3          from: {
4              id: fromCurrency,
5              amount: amount
6          },
7          to: {
8              id: toCurrency,
9              amount: targetValue
10         },
11         rate: targetValue / amount
12     };
13 } catch (error) {
14     console.error('Erreur lors de la conversion:', error);
15     return { success: false, error: 'Erreur de conversion'
16         };
17 }
18 }
19
20 // Utilisation
21 async function testConverter() {
22     const converter = new CryptoConverter();
23     const initialized = await converter.initialize();
24
25     if (initialized) {
26         // Convertir 1 BTC en ETH
27         const btcToEth = converter.convert(1, 'bitcoin', 'ethereum');
28         console.log('1 BTC = ${btcToEth.to.amount.toFixed(4)} ETH');
29
30         // Convertir 100 EUR en BTC
31         const eurToBtc = converter.convert(100, 'euro', 'bitcoin');
32         console.log('100 EUR = ${eurToBtc.to.amount.toFixed(8)} BTC');
33
34         // Convertir 1000 USD en EUR
35         const usdToEur = converter.convert(1000, 'united-states-dollar', 'euro');
36         console.log('1000 USD = ${usdToEur.to.amount.toFixed(2)} EUR');
37     }
38 }
39
40 testConverter();
```



## 6.2 Dashboard de portfolio

Creez un tableau de bord simple pour suivre la valeur d'un portfolio de cryptomonnaies.

### Exemple de portfolio en JavaScript

```
1 class CryptoPortfolio {
2   constructor() {
3     this.holdings = [];
4     this.cryptoData = {};
5   }
6
7   async updatePrices() {
8     try {
9       const response = await fetch('https://api.coincap.io/v2/
10         assets');
11       const result = await response.json();
12
13       // Indexer les donnees pour un acces facile
14       result.data.forEach(crypto => {
15         this.cryptoData[crypto.id] = crypto;
16       });
17
18       return true;
19     } catch (error) {
20       console.error('Erreur lors de la mise a jour des prix:',
21         error);
22       return false;
23     }
24   }
25
26   addHolding(cryptoId, amount, purchasePrice = null) {
27     this.holdings.push({
28       cryptoId,
29       amount: parseFloat(amount),
30       purchasePrice: purchasePrice ? parseFloat(purchasePrice)
31         : null
32     });
33     console.log('Ajoute ${amount} ${cryptoId} au portfolio');
34   }
35
36   removeHolding(index) {
37     if (index >= 0 && index < this.holdings.length) {
38       const removed = this.holdings.splice(index, 1)[0];
39       console.log('Retire ${removed.amount} ${removed.cryptoId}
40         } du portfolio');
41       return true;
42     }
43     return false;
44   }
45 }
```

## Exemple de portfolio en JavaScript

```
1
2  getPortfolioSummary() {
3    let totalValueUsd = 0;
4    let totalInvestmentUsd = 0;
5    const details = [];
6
7    this.holdings.forEach(holding => {
8      const crypto = this.cryptoData[holding.cryptoId];
9
10     if (crypto) {
11       const currentPriceUsd = parseFloat(crypto.priceUsd);
12       const currentValueUsd = holding.amount *
13         currentPriceUsd;
14       totalValueUsd += currentValueUsd;
15
16       let profitLossUsd = null;
17       let profitLossPercentage = null;
18
19       if (holding.purchasePrice !== null) {
20         const investmentValueUsd = holding.amount * holding.
21           purchasePrice;
22         totalInvestmentUsd += investmentValueUsd;
23         profitLossUsd = currentValueUsd - investmentValueUsd
24           ;
25         profitLossPercentage = (profitLossUsd /
26           investmentValueUsd) * 100;
27       }
28
29       details.push({
30         cryptoId: holding.cryptoId,
31         symbol: crypto.symbol,
32         name: crypto.name,
33         amount: holding.amount,
34         currentPriceUsd,
35         currentValueUsd,
36         purchasePriceUsd: holding.purchasePrice,
37         profitLossUsd,
38         profitLossPercentage
39       });
40     }
41   });
42
43   const totalProfitLossUsd = totalInvestmentUsd > 0 ?
44     totalValueUsd - totalInvestmentUsd : null;
45   const totalProfitLossPercentage = totalInvestmentUsd > 0 ?
46     (totalProfitLossUsd / totalInvestmentUsd) * 100 : null
47     ;
```

## Exemple de portfolio en JavaScript

```
1
2     return {
3         totalValueUsd,
4         totalInvestmentUsd: totalInvestmentUsd > 0 ?
5             totalInvestmentUsd : null,
6         totalProfitLossUsd,
7         totalProfitLossPercentage,
8         details
9     };
10
11 printPortfolio() {
12     const summary = this.getPortfolioSummary();
13
14     console.log('\n=== PORTFOLIO SUMMARY ===');
15     console.log('Total Value: ${summary.totalValueUsd.toFixed(
16         2)}}');
17
18     if (summary.totalInvestmentUsd !== null) {
19         console.log('Total Investment: ${summary.
20             totalInvestmentUsd.toFixed(2)}}');
21         console.log('Total Profit/Loss: ${summary.
22             totalProfitLossUsd.toFixed(2)} (${summary.
23             totalProfitLossPercentage.toFixed(2)}%)');
24     }
25
26     console.log('\n=== HOLDINGS ===');
27     summary.details.forEach(item => {
28         console.log(`${item.name} (${item.symbol})`);
29         console.log('  Amount: ${item.amount}');
30         console.log('  Current Price: ${item.currentPriceUsd.
31             toFixed(2)}}');
32         console.log('  Current Value: ${item.currentValueUsd.
33             toFixed(2)}}');
34
35         if (item.purchasePriceUsd !== null) {
36             console.log('    Purchase Price: ${item.purchasePriceUsd.
37                 toFixed(2)}}');
38             console.log('    Profit/Loss: ${item.profitLossUsd.
39                 toFixed(2)} (${item.profitLossPercentage.toFixed(2)}
40                 %)}');
41         }
42
43         console.log('---');
44     });
45 }
46
47 }
```

## Exemple de portfolio en JavaScript

```
1 // Utilisation
2 async function testPortfolio() {
3   const portfolio = new CryptoPortfolio();
4
5   // Mettre a jour les prix
6   await portfolio.updatePrices();
7
8   // Ajouter des cryptomonnaies au portfolio
9   portfolio.addHolding('bitcoin', 0.5, 50000); // 0.5 BTC
10  achete a 50000 USD
11  portfolio.addHolding('ethereum', 5, 3000);    // 5 ETH
12  achetes a 3000 USD
13  portfolio.addHolding('dogecoin', 1000);       // 1000 DOGE,
14  sans prix d'achat
15
16  // Afficher le portfolio
17  portfolio.printPortfolio();
18 }
19
20 testPortfolio();
```

## 7 Conclusion

L'API CoinCap est un outil puissant et flexible qui permet d'accéder facilement aux données des cryptomonnaies en temps réel. Grâce à ce tutoriel, vous avez appris à :

- Comprendre la structure et les endpoints de l'API CoinCap
- Récupérer la liste des cryptomonnaies et leurs détails
- Obtenir l'historique des prix et les informations sur les marchés
- Utiliser l'authentification pour augmenter les limites de taux
- Appliquer des bonnes pratiques comme la mise en cache et la gestion des erreurs
- Créer des applications pratiques comme un convertisseur et un portfolio

N'hésitez pas à explorer davantage la documentation officielle pour découvrir toutes les fonctionnalités offertes par l'API : <https://docs.coincap.io/>