KCY1 Token - Ръководство за тестване

🎯 Цел на документа

Този документ обяснява **КАК да тестваш токена KCY1** - какви са опциите, какво трябва да провериш, и стъпка по стъпка инструкции.

📊 ДВА НАЧИНА ЗА ТЕСТВАНЕ

Вариант 1: Ръчно тестване в Remix IDE ⭐ ПРЕПОРЪЧВАМ

- ✅ Най-бързо и лесно
- ✅ Не изисква инсталация
- ✅ Работи директно в browser
- ✅ Перфектно за твоя случай
- ❌ Трябва да тестваш всичко ръчно

Вариант 2: Автоматично тестване с Hardhat

- ✅ 40+ автоматични теста за секунди
- ✅ Професионален подход
- ✅ Code coverage отчет
- ❌ Изисква Node.js инсталация
- ❌ По-сложно за настройка

ВАРИАНТ 1: РЪЧНО ТЕСТВАНЕ В REMIX IDE

🚀 Стъпка 1: Отваряне на Remix

1. Отвори browser (Chrome, Firefox, Brave)
2. Отиди на: **https://remix.ethereum.org**
3. Изчакай да се зареди (5-10 секунди)

📝 Стъпка 2: Създаване на contract файл

1. **Ляв панел:** Виждаш "File Explorer"
2. **Кликни:** На иконата "Create new file" (+ символ)
3. **Име на файл:** `KCY1Token.sol`
4. **Копирай:** Целия Solidity код от първия артефакт
5. **Залепи:** В новия файл
6. **Запази:** Ctrl+S (или Cmd+S на Mac)

⚙️ Стъпка 3: Компилиране

1. **Ляв панел:** Кликни на "Solidity Compiler" (3-та икона отгоре)
2. **Версия:** Провери че е избрана **0.8.20** (или 0.8.x)
3. **Auto compile:** Включи го (чекбокс)
4. **Кликни:** "Compile KCY1Token.sol" (синият бутон)
5. **Чакай:** Зелена отметка = успешно ✅

Ако има грешки:

- Прочети грешката внимателно
- Провери дали кодът е копиран правилно
- Провери дали версията на Solidity е 0.8.20

🎮 Стъпка 4: Deploy на contract-а

1. **Ляв панел:** Кликни на "Deploy & Run Transactions" (4-та икона)
2. **Environment:** Избери "Remix VM (Shanghai)" (тестова среда)
3. **Account:** Виждаш адрес с ~100 ETH (тестови пари)
4. **Contract:** Трябва да е избран "KCY1Token"
5. **Кликни:** "Deploy" (оранжевият бутон)
6. **Чакай:** Под "Deployed Contracts" ще се появи твоят contract ✅

Важно: Запиши си owner адреса (той е от Account полето)

🧪 Стъпка 5: ТЕСТОВИ СЦЕНАРИИ

ТЕСТ 1: Проверка на началните параметри

Цел: Провери дали токенът е създаден правилно

Какво да провериш:

1. **Име и символ:**

```
Кликни: name()
Очаквано: "KCY1"

Кликни: symbol()
Очаквано: "KCY1"

Кликни: decimals()
Очаквано: 18
```

2. **Total Supply:**

```
Кликни: totalSupply()
Очаквано: 1000000000000000000000000
(това е 1,000,000 с 18 нули = 1 милион токена)
```

3. **Owner баланс:**

```
Кликни: balanceOf
Въведи: [твоят owner адрес]
Кликни: "call"
Очаквано: 600000000000000000000000
(600,000 токена)
```

4. **Contract баланс:**

```
Кликни: balanceOf
Въведи: [адресът на contract-a]
Кликни: "call"
Очakvano: 400000000000000000000000
(400,000 токена)
```

5. **Trading статус:**

Кликни: isTradingEnabled()
Очаквано: false (търговията е блокирана първите 48ч)

Кликни: timeUntilTradingEnabled()
Очаквано: ~172800 (48 часа в секунди)

✅ **PASS ако:** Всички стойности съвпадат
❌ **FAIL ако:** Някоя стойност е грешна

ТЕСТ 2: Задаване на Exempt адреси

Цел: Провери дали можеш да зададеш преференциални адреси

Стъпки:

1. **Създай тестови адреси:**
   - В Remix: Ляв панел "Deploy & Run" → виждаш списък с адреси
   - Копирай 2-3 адреса за тестване

2. **Задай exempt адреси:**

Кликни: setExemptAddresses

Въведи параметри:
_addresses: ["0xАдрес1", "0xАдрес2", "0x0000000000000000000000000000000000000000",
"0x0000000000000000000000000000000000000000", "0x0000000000000000000000000000000000000000"]

_router: "0x10ED43C718714eb63d5aA57B78B54704E256024E"

_factory: "0xcA143Ce32Fe78f1f7019d7d551a6402fC5350c73"

Кликни: "transact"

3. **Провери дали са зададени:**

Кликни: isExemptAddress
Въведи: "0xАдрес1"
Очаквано: true ✅

Кликни: isExemptAddress
Въведи: "0xСлучаен адрес"
Очаквано: false ❌

4. **Провери getExemptAddresses:**

```
Кликни: getExemptAddresses()
Очаквано: Виждаш масив с 5 адреса + router + factory
locked: false (все още не е lock-нат)
```

✅ **PASS ако:** Exempt адресите се задават правилно
❌ **FAIL ако:** isExemptAddress връща грешни резултати

ТЕСТ 3: Трансфер С такси (обикновен адрес)

Цел: Провери дали таксите работят (3% burn + 5% owner)

⚠️ **Важно:** Първо трябва да изминат 48 часа! В Remix можеш да "skip-неш" времето:

Skip на 48 часа:

```
1. Отвори Console в Remix (долен панел)
2. Напиши:
  await web3.currentProvider.send({
    jsonrpc: '2.0',
    method: 'evm_increaseTime',
    params: [172800], // 48 часа в секунди
    id: new Date().getTime()
  })
3. Натисни Enter
4. Напиши:
  await web3.currentProvider.send({
    jsonrpc: '2.0',
    method: 'evm_mine',
    params: [],
    id: new Date().getTime()
  })
5. Натисни Enter
```

Сега тествай трансфер:

1. **Избери ДРУГ адрес** (не owner):
   - Ляв панел → Account → избери втория или третия адрес

2. **Owner изпраща на този адрес:**
   - Върни се на owner адреса
   - Кликни: transfer
   - Параметри:

```
to: "0xАдресът който избра"
amount: 10000000000000000000 (10,000 токена)
```

- Кликни: "transact"

3. **Провери баланса на получателя:**

```
Кликни: balanceOf
Въведи: [адресът на получателя]
Очаквано: 10000000000000000000 (пълните 10,000 - owner е exempt, няма такси)
```

4. **Сега получателят изпраща на трети адрес:**
   - Избери получателя от Account списъка
   - Кликни: transfer
   - Параметри:

```
to: "0xТрети адрес"
amount: 10000000000000000000 (1,000 токена)
```

- Кликни: "transact"

5. **Провери резултата:**

```
Кликни: balanceOf
Въведи: [третият адрес]
Очаквано: ~92000000000000000000 (920 токена = 92% от 1000)

3% (30 токена) са изгорени
5% (50 токена) са отишли при owner
92% (920 токена) са получени
```

6. **Провери дали totalSupply е намаляло:**

Кликни: totalSupply()
Очаквано: По-малко от 1,000,000 (изгорените токени са премахнати)

✅ **PASS ако:** Получателят получава ~92%, totalSupply намалява
❌ **FAIL ако:** Получателят получава пълните 100% или грешен процент

ТЕСТ 4: Exempt адрес БЕЗ такси

Цел: Провери дали exempt адресите НЕ плащат такси

Стъпки:

1. **Задай един адрес като exempt** (ако не си го направил в ТЕСТ 2)
2. **Owner изпраща на exempt адреса:**

Кликни: transfer
to: "0xExempt адрес"
amount: 50000000000000000000 (5,000 токена)
Кликни: "transact"

3. **Провери баланса:**

Кликни: balanceOf
Въведи: [exempt адрес]
Очаквано: ТОЧНО 50000000000000000000 (пълните 5,000)

4. **Exempt адресът изпраща на обикновен адрес:**
   - Избери exempt адреса от Account
   - Кликни: transfer
   - Параметри:

to: "0xОбикновен адрес"
amount: 10000000000000000000 (1,000 токена)

5. **Провери резултата:**

Кликни: balanceOf
Въведи: [обикновен адрес]
Очаквано: ТОЧНО 10000000000000000000 (пълните 1,000 - БЕЗ такси!)

6. **Провери че totalSupply НЕ е намаляло:**

Кликни: totalSupply()
Очаквано: Същото като преди (exempt не изгаря токени)

✅ **PASS ако:** Exempt трансферите са 100%, без burn
❌ **FAIL ако:** Exempt адресите плащат такси

ТЕСТ 5: Max Transaction Limit (1000 токена)

Цел: Провери дали обикновените адреси не могат да изпращат над 1000 токена

Стъпки:

1. **Owner изпраща на обикновен адрес:**

transfer
to: "0xОбикновен адрес"
amount: 50000000000000000000 (5,000 токена)

2. **Обикновен адрес опитва да изпрати 1001 токена:**
   - Избери обикновения адрес от Account
   - Кликни: transfer
   - Параметри:

to: "0xДруг адрес"
amount: 10010000000000000000 (1,001 токена)

- Кликни: "transact"

3. **Очаквана грешка:**

❌ Transaction reverted: "Exceeds max transaction (1000 tokens)"

4. **Опитай с точно 1000 токена:**

transfer
to: "0xДруг адрес"
amount: 10000000000000000000 (1,000 токена)
Кликни: "transact"

5. **Този път трябва да успее:**

```
✅ Transaction successful
```

✅ **PASS ако:** 1001+ токена FAIL, 1000 токена SUCCESS
❌ **FAIL ако:** Позволява над 1000 токена

ТЕСТ 6: Max Wallet Limit (20,000 токена)

Цел: Провери дали един портфейл не може да държи над 20,000 токена

Стъпки:

1. **Owner изпраща 19,000 токена на обикновен адрес:**

```
transfer
to: "0xТестов адрес"
amount: 190000000000000000000
```

2. **Опитай да изпратиш още 2,000 токена:**

```
transfer
to: [същият адрес]
amount: 200000000000000000000
```

3. **Очаквана грешка:**

```
❌ Transaction reverted: "Recipient would exceed max wallet (20,000 tokens)"
```

4. **Опитай да изпратиш само 500 токена:**

```
transfer
to: [същият адрес]
amount: 50000000000000000000
```

5. **Този път трябва да успее:**

```
✅ Transaction successful
Общ баланс: ~19,500 токена (под лимита)
```

✅ **PASS ако:** Над 20K FAIL, под 20K SUCCESS
❌ **FAIL ако:** Позволява над 20,000 токена

ТЕСТ 7: Cooldown период (2 часа)

Цел: Провери дали не можеш да правиш 2 транзакции за по-малко от 2 часа

Стъпки:

1. **Обикновен адрес прави първи трансфер:**

transfer
to: "0xАдрес1"
amount: 500000000000000000000 (500 токена)
✅ SUCCESS

2. **Веднага опитай втори трансфер:**

transfer
to: "0xАдрес2"
amount: 500000000000000000000

3. **Очаквана грешка:**

❌ Transaction reverted: "Must wait 2 hours between transactions"

4. **Skip на 2 часа в Remix:**

```javascript
// В Console:
await web3.currentProvider.send({
  jsonrpc: '2.0',
  method: 'evm_increaseTime',
  params: [7200], // 2 часа
  id: new Date().getTime()
})

await web3.currentProvider.send({
  jsonrpc: '2.0',
  method: 'evm_mine',
  params: [],
  id: new Date().getTime()
})
```

5. **Опитай отново:**

transfer
to: "0xАдрес2"
amount: 500000000000000000000
✅ SUCCESS

✅ **PASS ако:** Втората транзакция FAIL без cooldown, SUCCESS след 2ч
❌ **FAIL ако:** Позволява множество транзакции без изчакване

ТЕСТ 8: Pause механизъм

Цел: Провери дали pause блокира всички трансфери за 48 часа

Стъпки:

1. **Owner активира pause:**

Кликни: pause()
✅ Transaction successful

2. **Провери статус:**

Кликни: isPaused()
Очаквано: true

3. **Опитай трансфер:**

transfer
to: "0xАдрес"
amount: 100000000000000000000

4. **Очаквана грешка:**

❌ Transaction reverted: "Contract is paused"

5. **Skip на 48 часа:**

```javascript
await web3.currentProvider.send({
  jsonrpc: '2.0',
  method: 'evm_increaseTime',
  params: [172800],
  id: new Date().getTime()
})

await web3.currentProvider.send({
  jsonrpc: '2.0',
  method: 'evm_mine',
  params: [],
  id: new Date().getTime()
})
```

6. **Опитай отново:**

```
transfer
to: "0xАдрес"
amount: 10000000000000000000
✅ SUCCESS (паузата е изтекла)
```

✅ **PASS ако:** По време на pause = FAIL, след 48ч = SUCCESS
❌ **FAIL ако:** Позволява трансфери по време на pause

ТЕСТ 9: Blacklist функция

Цел: Провери дали blacklist-натите адреси не могат да търгуват

Стъпки:

1. **Owner blacklist-ва адрес:**

```
Кликни: setBlacklist
Параметри:
  account: "0xТестов адрес"
  status: true
✅ Transaction successful
```

2. **Провери статус:**

Кликни: isBlacklisted
Въведи: "0xТестов адрес"
Очаквано: true

3. **Опитай да изпратиш НА blacklist адрес:**

transfer
to: "0xТестов адрес"
amount: 10000000000000000000

4. **Очаквана грешка:**

❌ Transaction reverted: "Recipient is blacklisted"

5. **Опитай да изпратиш ОТ blacklist адрес:**
   - Избери blacklist адреса от Account

transfer
to: "0xДруг адрес"
amount: 10000000000000000000
❌ Transaction reverted: "Sender is blacklisted"

6. **Премахни от blacklist:**

setBlacklist
account: "0xТестов адрес"
status: false

7. **Опитай отново:**

transfer
to: "0xТестов адрес"
amount: 10000000000000000000
✅ SUCCESS

✅ **PASS ако:** Blacklist адресите не могат да търгуват
❌ **FAIL ако:** Blacklist-ването не работи

ТЕСТ 10: Lock механизъм

Цел: Провери дали след lock не можеш да променяш exempt адресите

Стъпки:

1. **Задай exempt адреси:**

```
setExemptAddresses(
  ["0xАдрес1", "0xАдрес2", "0x0", "0x0", "0x0"],
  router,
  factory
)
```
✅ SUCCESS

2. **Провери lock статус:**

Кликни: exemptAddressesLocked()
Очаквано: false

3. **Активирай lock:**

Кликни: lockExemptAddresses()
✅ Transaction successful

4. **Провери lock статус:**

Кликни: exemptAddressesLocked()
Очаквано: true

5. **Опитай да промениш exempt адресите:**

```
setExemptAddresses(
  ["0xНОВ_адрес", "0x0", "0x0", "0x0", "0x0"],
  router,
  factory
)
```

6. **Очаквана грешка:**

❌ Transaction reverted: "Exempt addresses are locked forever"

✅ **PASS ако:** След lock не може да се променят exempt адресите
❌ **FAIL ако:** Позволява промяна след lock

📊 CHECKLIST ЗА ТЕСТВАНЕ

Използвай този checklist за да се убедиш, че си тествал всичко:

- [ ] Deploy и начални параметри
- [ ] Задаване на exempt адреси
- [ ] Трансфер с такси (3% + 5%)
- [ ] Exempt трансфер БЕЗ такси
- [ ] Max transaction limit (1000)
- [ ] Max wallet limit (20,000)
- [ ] Cooldown период (2 часа)
- [ ] Pause механизъм (48 часа)
- [ ] Blacklist функция
- [ ] Lock механизъм

Всичко минало? Готов си за production! 🚀

ВАРИАНТ 2: АВТОМАТИЧНО ТЕСТВАНЕ С HARDHAT

📦 Инсталация и настройка

Стъпка 1: Инсталирай Node.js

```bash
# Свали от: https://nodejs.org
# Инсталирай LTS версията
# Провери:
node --version
npm --version
```

Стъпка 2: Създай проект

```bash
mkdir kcy1-token
cd kcy1-token
npm init -y
```

Стъпка 3: Инсталирай dependencies

```bash
npm install --save-dev hardhat
npm install --save-dev @nomicfoundation/hardhat-toolbox
npm install --save-dev @nomicfoundation/hardhat-chai-matchers
npm install --save-dev chai ethers
```

Стъпка 4: Инициализирай Hardhat

```bash
npx hardhat init
# Избери: "Create a JavaScript project"
# Натисни Enter за всички въпроси
```

Стъпка 5: Копирай файловете

Структура:

```
kcy1-token/
├── contracts/
│   └── KCY1Token.sol          ← Solidity код от артефакт 1
├── test/
│   └── KCY1Token.test.js      ← JavaScript тестове от артефакт 2
├── hardhat.config.js
└── package.json
```

Стъпка 6: Настрой hardhat.config.js

```javascript
require("@nomicfoundation/hardhat-toolbox");

module.exports = {
  solidity: {
    version: "0.8.20",
    settings: {
      optimizer: {
        enabled: true,
        runs: 200
      }
    }
  }
};
```

Стъпка 7: Пусни тестовете

```bash
npx hardhat test
```

📊 Очаквани резултати

Успешно изпълнение:

```
KCY1 Token - Full Test Suite
  1. Deploy и начални параметри
    ✓ Трабва да има правилно име и символ (250ms)
    ✓ Трябва да има правилен total supply (100ms)
    ✓ Трябва да разпредели токените правилно (150ms)
    ✓ Owner трябва да е правилен (50ms)
    ✓ Търговията трябва да е блокирана първите 48 часа (75ms)

  2. Exempt адреси - Задаване и Lock
    ✓ Трябва да може да зададе exempt адреси ПРЕДИ lock (200ms)
    ✓ Трябва да може да променя exempt адресите МНОГОКРАТНО преди lock (300ms)
    ✓ Lock трябва да блокира промени ЗАВИНАГИ (150ms)
    ✓ getExemptAddresses() трябва да връща правилна информация (100ms)

  3. Трансфери и такси
    ✓ Обикновен трансфер трябва да има 3% burn + 5% owner fee (250ms)
    ✓ Exempt адреси НЕ трябва да плащат такси (200ms)
    ✓ Owner трансферите НЕ трябва да имат такси (150ms)

  ... още тестове ...

40 passing (8s)
0 failing
```

При грешки:

```
1 failing

1) KCY1 Token - Full Test Suite
     3. Трансфери и такси
       Обикновен трансфер трябва да има 3% burn + 5% owner fee:

    AssertionError: expected 1000 to equal 920
```

🎯 Полезни команди

```bash
# Компилирай само contract-а
npx hardhat compile

# Пусни тестовете с детайли
npx hardhat test --verbose

# Пусни ЕДИН конкретен тест
npx hardhat test --grep "Deploy и начални параметри"

# Gas reporter
REPORT_GAS=true npx hardhat test

# Code coverage
npx hardhat coverage
```

🐛 Troubleshooting

Проблем: "Cannot find module 'hardhat'"

```bash
rm -rf node_modules package-lock.json
npm install
```

Проблем: Тестовете timeout

```javascript
// В test файла добави:
this.timeout(30000); // 30 секунди
```

Проблем: "Invalid opcode"

- Провери Solidity версията (трябва да е 0.8.20)

✅ ФИНАЛЕН CHECKLIST

Преди Production Deploy:

- [ ] Всички тестове минават (40/40 passing)
- [ ] Code coverage > 90%
- [ ] Exempt адресите са правилни
- [ ] PancakeSwap адресите са верни (BSC Mainnet)
- [ ] Разбираш как работи Lock механизмът
- [ ] Имаш план за emergency pause
- [ ] Знаеш кои адреси ще blacklist-неш при нужда
- [ ] Contract-ът е преглед от audit компания (препоръчително)
- [ ] Ликвидността е готова за добавяне
- [ ] Community е информирано за launch

Версия: 1.0
Дата: 2024
Статус: Ready for Testing ✅