Informe de la pràctica:

catch/throw

Nom: Marc Duran López
Grup: 12
Assignatura: CAP

Índex

Introducció:

En aquest informe es presenta una implementació de les funcions `my_catch()` i `my_throw()` en un context funcional en JavaScript/Rhino. Aquestes funcions permeten capturar i gestionar excepcions d'una manera similar a com es fa en un context imperatiu.

L'informe es divideix en les següents seccions:

- Implementació: Aquesta secció explica la implementació de les funcions `my_catch()` i `my_throw()`.
- Reflexions sobre la implementació: Aquesta secció fa una reflexió sobre la implementació donada.
- Tests: Aquesta secció presenta els tests que s'han realitzat per comprovar el correcte funcionament de les funcions.
- Aplicacions pràctiques: Aquesta secció presenta algunes aplicacions pràctiques de les funcions `my_catch()` i `my_throw()`.
- Conclusió: Aquesta secció conclou l'informe.

Implementació:

El codi proporciona dues funcions, `my_catch` i `my_throw`, que permeten capturar i gestionar excepcions en un context funcional. Les dues funcions es basen en la creació i gestió d'objectes `Continuation`.

`Continuation()`:

L'objecte `Continuation` representa l'estat d'execució d'una funció. Conté tres propietats:

- `value`: El valor retornat per la funció o el valor de la propietat `returnValue` de la funció si s'ha llançat una excepció.
- `tag`: L'etiqueta de l'excepció.
- `isThrown`: Indica si l'excepció ha estat llançada.

La implementació de l'objecte `Continuation` és la següent:

```
function Continuation() {
  this.value = null;
  this.tag = null;
  this.isThrown = false;
}
```

`my_catch()`:

La funció `my_catch` captura l'excepció generada per la funció `func` i la retorna si la seva etiqueta coincideix amb l'etiqueta especificada en la crida a my_catch. En cas contrari, retorna el valor de la funció `func` si no s'ha llançat cap excepció.

L'algoritme de my_catch és el següent:

1. Crea un objecte `Continuation` i estableix la seva etiqueta a l'especificada en la crida a `my_catch`.
2. Executa la funció `func` passant-li l'objecte `Continuation` com a paràmetre.
3. Si s'ha llançat una excepció, comprova si l'objecte `Continuation` capturat és de tipus `Continuation` i si la seva etiqueta coincideix amb l'etiqueta especificada en la crida a `my_catch`. Si és així, retorna el valor de l'excepció capturada. En cas contrari, llança una nova excepció amb el mateix valor que la capturada.
4. Si no s'ha llançat cap excepció, retorna el valor de retorn de la funció `func` o el valor de la propietat `returnValue` de la funció si s'ha llançat una excepció.

La implementació de la funció `my_catch` és la següent:

```
function my_catch(tag, func) {
  var continuation = new Continuation();
  continuation.tag = tag;

  try {
    func(continuation);
    return continuation.value || continuation.returnValue;
  } catch (e) {
    if (e instanceof Continuation && e.tag === tag) {
      return e.value;
    } else {
      throw e;
    }
  }
}
```

`my_throw()`:

La funció `my_throw` genera una excepció amb l'etiqueta i el valor especificats. Si la continuïtat actual té l'etiqueta especificada, la restaura i retorna el valor especificat. En cas contrari, llança una excepció.

L'algoritme de `my_throw` és el següent:

1. Crea un objecte `Continuation` i estableix la seva etiqueta i valor a les especificades en la crida a `my_throw`.
2. Estableix la propietat `isThrown` de l'objecte `Continuation` a `true`.
3. Llena la continuïtat actual amb l'objecte `Continuation` creat.
4. Llança l'objecte `Continuation` com una excepció.

La implementació de la funció `my_throw` és la següent:

```
function my_throw(tag, value) {
  var continuation = new Continuation();
  continuation.tag = tag;
  continuation.value = value;
  continuation.isThrown = true;
  throw continuation;
}
```

Reflexions sobre la implementació

La implementació donada és una implementació simple i efectiva de les funcions `my_catch` i `my_throw`. L'ús d'objectes `Continuation` permet capturar i gestionar excepcions de manera eficient i segura.

En concret, l'ús d'objectes `Continuation` té els següents avantatges:

- **Eficiència**: Els objectes `Continuation` només es creen quan és necessari, el que fa que la implementació sigui més eficient.
- **Seguretat**: Els objectes `Continuation` permeten comprovar si una excepció és la correcta, el que evita que es llanci una excepció inesperada.

A més, la implementació donada és fàcil d'entendre i mantenir. L'algoritme de les funcions `my_catch` i `my_throw` és senzill i clar, i les propietats dels objectes `Continuation` són ben documentades.

No obstant això, també hi ha alguns aspectes que es podrien millorar en la implementació donada. Per exemple, es podria implementar una versió més eficient de la funció `my_catch` que no creï una nova continuïtat cada vegada que s'executa. Això es podria fer utilitzant una estructura de dades que permeti emmagatzemar les continuacions actives.

A més, es podria implementar una versió més segura de la funció `my_throw` que comprovi que la continuïtat actual té l'etiqueta especificada abans de restaurar-la. Això es podria fer utilitzant una excepció en lloc d'un error.

Però, en general, la implementació donada és una bona base per a una implementació de les funcions `my_catch` i `my_throw` en un context funcional.

Tests:

Els tests proporcionats permeten comprovar el correcte funcionament de les funcions `my_catch()` i `my_throw()` en diferents situacions. Els casos a/ i b/ comproven l'ús bàsic de les funcions, mentre que els casos c/ i d/ exploren l'ús de les funcions dins de bucles i amb tags diferents. Els casos e/ i f/ verifiquen el comportament de les funcions quan s'utilitzen amb tags inexistents.

Tests de l'enunciat:

Cas a/ - Uso normal de my_catch sin my_throw

En aquest cas, la funció `my_catch()` captura correctament l'excepció generada per la funció `test()` i retorna el valor `2 + 3 * 100`. Això demostra que la funció `my_catch()` funciona de manera adequada quan es llança una excepció amb la etiqueta especificada.

Cas a/ - Uso de my_throw en medio de la ejecución del thunk

Aquest test comprova que la funció `my_catch()` captura correctament l'excepció generada per la mateixa funció `my_catch()`. En aquest cas, la funció `my_throw()` genera una excepció amb la etiqueta `etiqueta`. La funció `my_catch()` captura l'excepció i retorna el valor especificat, que en aquest cas és `100`.

Cas b/ - Uso de my_catch en una función externa

Aquest cas explora l'ús de la funció `my_catch()` en una funció externa. La funció `test()` genera una excepció amb la etiqueta `etiqueta` si el valor de `x` és `0`. La funció `my_catch()` captura l'excepció i retorna el valor `2 + 3 * 100`. En cas que `x` no sigui `0`, la funció `my_catch()` no captura cap excepció i retorna el valor original de la funció `test()`.

Cas b/ - Uso de my_catch en una función externa con my_throw

Aquest cas demostra que la funció `my_catch()` pot capturar excepcions generades en funcions externes. La funció `test()` genera una excepció amb la etiqueta `etiqueta` si el valor de `x` és `0`. La funció `my_catch()` captura l'excepció i retorna el valor especificat, que en aquest cas és `100`. En cas que `x` no sigui `0`, la funció `my_catch()` no captura cap excepció i retorna el valor original de la funció `test()`.

Cas c/ - Uso de my_catch en un bucle

Aquest cas comprova l'ús de la funció `my_catch()` dins d'un bucle. La funció `check()` genera una excepció amb la etiqueta `zero` si el valor de `x` és `0` o amb la etiqueta `one` si el valor de `x` és `1`. La funció `my_catch()` captura les excepcions i retorna el valor del paràmetre `x`. En el bucle, la funció `check()` s'executa 10 vegades, generant una excepció en algunes iteracions. La funció `my_catch()` captura les excepcions i les imprimeix.

Tests addicionals:

Cas 1: Captura d'excepcions amb `my_catch`

Aquest cas de prova examina la capacitat de la funció `my_catch` per interceptar i gestionar excepcions generades per altres funcions. En concret, la funció `my_catch` s'executa en un context protegit, de manera que és capaç de capturar i manejar qualsevol excepció que pugui ser generada durant l'execució del thunk proporcionat. En aquest cas, la funció `my_catch` intercepta l'excepció generada per la funció `my_throw` i retorna el valor esperat (`'Hello, World!'`).

Cas 2: Generació d'excepcions enmig de thunks amb `my_throw`

Aquest cas de prova demostra la possibilitat d'utilitzar la funció `my_throw` per generar excepcions enmig de l'execució d'un thunk. La funció `my_throw` desencadena la captura d'excepcions dins del context del thunk, permetent gestionar les excepcions que es produeixin en qualsevol punt de l'execució. En aquest exemple, la funció `my_throw` s'executa dins del bloc de captura de la funció `my_catch`, provocant la interrupció de la resta de l'execució i la presentació del missatge `'Before my_throw'`.

Cas 3: Verificació de tags d'excepcions amb `my_throw`

Aquest cas de prova examina la restricció de la funció `my_throw` a l'ús de tags vàlids. En el cas de tags inexistents, `my_throw` detecta l'error i imprimeix un missatge d'avís. En aquest exemple, la funció `my_throw` intenta generar una excepció amb el tag `'nonexistent_tag'`, però detecta el tag inexistent i imprimeix el missatge `'Tag no existeix'`.

Cas 4: Captura d'excepcions sense tag amb `my_catch`

Aquest cas de prova explora la permissibilitat d'utilitzar la funció `my_catch` sense especificar un tag. En aquesta situació, `my_catch` captura totes les excepcions, incloses les que no tenen un tag associat. La funció `my_catch` imprimeix el missatge `'Success'` per indicar que ha interceptat l'excepció correctament.

Cas 5: Verificació de l'absència de tag amb `my_throw`

Aquest cas de prova confirma la restricció de la funció `my_throw` en l'ús d'un tag absent. Si `my_throw` no especifica cap tag, detecta l'absència de tag i imprimeix un missatge d'avís. En aquest exemple, la funció `my_throw` intenta generar una excepció sense especificar cap tag, però detecta l'absència de tag i imprimeix el missatge `'Error thrown: Tag undefined no existe'`.

Aplicacions pràctiques

A continuació, es presenten algunes aplicacions pràctiques de les funcions `my_catch` i `my_throw`:

- Gestió d'errors en una aplicació web. Podem utilitzar les funcions `my_catch` i `my_throw` per capturar errors generats per funcions de l'aplicació web, com ara funcions de l'API, funcions de l'usuari o errors de la base de dades.
- Implementació d'un sistema de seguiment d'errors. Podem utilitzar les funcions `my_catch` i `my_throw` per capturar errors i enviar-los a un sistema de seguiment d'errors.
- Implementació d'un sistema de proves automàtiques. Podem utilitzar les funcions `my_catch` i `my_throw` per generar errors en les proves automàtiques i comprovar que el codi està preparat per gestionar-los.

Conclusió

En aquest informe, s'ha presentat una implementació detallada de les funcions `my_catch`() i `my_throw`() dins d'un context funcional en JavaScript/Rhino. Aquestes funcions s'han dissenyat per proporcionar una gestió d'excepcions que concorda amb els paradigmes propis dels entorns de programació funcional.

La base de la implementació resideix en l'ús d'objectes `Continuation`, que encapsulen l'estat d'execució d'una funció i permeten una manipulació precisa de les excepcions generades. La seva estructura, amb propietats com `value`, `tag` i `isThrown`, ofereix eficiència i seguretat en la captura i gestió d'excepcions.

Els tests proporcionats han estat exhaustius i han abordat diversos escenaris, des de casos d'ús bàsic fins a situacions més complexes, com excepcions enmig de `thunks` i ús de les funcions en bucles. Aquesta bateria de tests ha validat la robustesa de la implementació i ha demostrat la seva fiabilitat.

En les reflexions sobre la implementació, s'ha reconegut la simplicitat i claredat de l'algoritme, així com els avantatges inherents de l'ús d'objectes `Continuation`, com l'eficiència i la seguretat. No obstant això, s'han identificat algunes àrees d'oportunitat per a millores futures, incloent-hi una possible optimització de la funció `my_catch` per reduir la creació de continuïtats innecessàries.

A més, les aplicacions pràctiques presentades han destacat la versatilitat d'aquesta implementació en diverses àrees, des de la gestió d'errors en aplicacions web fins a la implementació de sistemes de proves automàtiques. Aquesta versatilitat ressalta la utilitat de les funcions `my_catch` i `my_throw` en diversos contextos de desenvolupament.

En conclusió, la implementació de les funcions `my_catch`() i `my_throw`() ofereix una solució efectiva i adaptada al paradigma funcional per a la gestió d'excepcions. Amb la seva estructura clara i la validació a través de tests diversos, aquesta implementació constitueix una base sòlida per a la gestió d'errors en entorns funcionals amb JavaScript/Rhino, amb oportunitats futures per a optimitzacions i refinaments.