AmortizationCalculation

Method Summary

Boolean	#addCompoundPeriodChange(newPeriod, date) Adds a compound period change.
Boolean	#addLoan(amount, firstDate, [lastDate], [period], [number], [startday]) Adds a loan.
Boolean	#addPayment(amount, firstDate, [lastDate], [period], [number], [startday]) Adds a payment.
Boolean	#addRateChange(newRate, date) Sets a new interest rate.
Boolean	#calculateAmortizationSchedule() Calculates the amortization schedule.
JSDataSet	#getAmortizationSchedule() Gets the amortization schedule as a JSDataSet.
Number	#getError() Returns the error that remains when solving for the unknown.
JSDataSet	#getEvents() Returns all the amortization events - such as rate changes, loan events, payment events, compounding period changes.
Number	#getRestBalance() Gets the rest balance after the amortization schedule.
Number	#getUnknown() Returns the solveForUnknown value.
Boolean	#isValidPeriod(period) Returns true if the period is valid, or false if the period is not valid.
Number	#roundMoney(amount) Rounds a number up to the nearest cents.
Boolean	#solveForUnknown() Returns true if successful or false if the call failed.
void	#sortEvents() Sorts the amortization events ascending by date.

Method Details

addCompoundPeriodChange

Boolean **addCompoundPeriodChange**(newPeriod, date)

Adds a compound period change.

Parameters
newPeriod
date

Returns
Boolean

Sample

```
var c = plugins.amortization.newCalculation();
c.addRateChange(r, new Date(2005, 0, 1));
c.addCompoundPeriodChange(12, new Date(2005, 0, 1));
c.addLoan(2000, new Date(2005, 0, 1));
c.addPayment(500, new Date(2005, 1, 28), null, 12, 5, 31);
```

addLoan

Boolean **addLoan**(amount, firstDate, [lastDate], [period], [number], [startday])

Adds a loan.

Parameters
amount
firstDate
[lastDate]
[period]
[number]
[startday]

Returns
Boolean

Sample

```
var c = plugins.amortization.newCalculation();
c.addRateChange(r, new Date(2005, 0, 1));
c.addCompoundPeriodChange(12, new Date(2005, 0, 1));
c.addLoan(2000, new Date(2005, 0, 1));
c.addPayment(500, new Date(2005, 1, 28), null, 12, 5, 31);
```

addPayment

Boolean **addPayment**(amount, firstDate, [lastDate], [period], [number], [startday])

Adds a payment.

Parameters

amount
firstDate
[lastDate]
[period]
[number]
[startday]

Returns

Boolean

Sample

```
var c = plugins.amortization.newCalculation();
c.addRateChange(r, new Date(2005, 0, 1));
c.addCompoundPeriodChange(12, new Date(2005, 0, 1));
c.addLoan(2000, new Date(2005, 0, 1));
c.addPayment(500, new Date(2005, 1, 28), null, 12, 5, 31);
```

addRateChange

Boolean **addRateChange**(newRate, date)

Sets a new interest rate.

Parameters

newRate
date

Returns

Boolean

Sample

```
var c = plugins.amortization.newCalculation();
c.addRateChange(r, new Date(2005, 0, 1));
c.addCompoundPeriodChange(12, new Date(2005, 0, 1));
c.addLoan(2000, new Date(2005, 0, 1));
c.addPayment(500, new Date(2005, 1, 28), null, 12, 5, 31);
```

calculateAmortizationSchedule

Boolean **calculateAmortizationSchedule**()

Calculates the amortization schedule.

Returns

Boolean

Sample

```
plugins.amortization.calculateAmortizationSchedule();
```

getAmortizationSchedule

JSDataSet **getAmortizationSchedule**()

Gets the amortization schedule as a JSDataSet.

Returns

JSDataSet

Sample

```
plugins.amortization.getAmortizationSchedule();
```

getError

Number **getError**()

Returns the error that remains when solving for the unknown.
Please note that the error should be less or equal to 1E-8 - otherwise, the solveForUnknown value is incorrect.

Returns

Number

Sample

```
//Returns the error that remains when solving for the unknown.
Please note that the error should be less or equal to 1E-8 - otherwise, the solveForUnknown value is incorrect.
var c = plugins.amortization.newCalculation();
// sets the rate to -1 for unknown.
c.addRateChange(-1, new Date(2005, 0, 1));
c.addCompoundPeriodChange(12, new Date(2005, 0, 1));
c.addLoan(2000, new Date(2005, 0, 1));
var lastDate = null;
var period = 12;
var number_count = 5;
var startday = 31;
c.addPayment(500, new Date(2005, 1, 28), lastDate, period,number_count, startday);
// solves for the interest rate.
c.solveForUnknown();
// gets the interest rate and the error in the calculation.
// which should be small (otherwise the calculation did
// not converge for some reason.
var r = c.getUnknown();
var e = c.getError();
```

getEvents

JSDataSet **getEvents**()

Returns all the amortization events - such as rate changes, loan events, payment events, compounding period changes.

Returns

JSDataSet

Sample

```
plugins.amortization.getEvents();
```

getRestBalance

Number **getRestBalance**()

Gets the rest balance after the amortization schedule.

Returns

Number

Sample

```
var rb = plugins.amortization.getRestBalance();
```

getUnknown

Number **getUnknown**()

Returns the solveForUnknown value.

Returns

Number

Sample

```
plugins.amortization.getUnknown();
```

isValidPeriod

Boolean **isValidPeriod**(period)

Returns true if the period is valid, or false if the period is not valid.

Parameters

period

Returns

Boolean

Sample

```
//Returns true if the period is valid, or false if the period is not valid.

var v_period = plugins.amortization.isValidPeriod(12);
```

roundMoney

Number **roundMoney**(amount)

Rounds a number up to the nearest cents.

Parameters

amount

Returns

Number

Sample

```
//Rounds a number up to the nearest cents.

//rounds the number up to 34.35
var rm = plugins.amortization.roundMoney(34.349384);
```

solveForUnknown

Boolean **solveForUnknown**()

Returns true if successful or false if the call failed.

Returns

Boolean

Sample

```
//Returns true if successful or false if the call failed.

plugins.amortization.solveForUnknown();
```

sortEvents

void **sortEvents**()

Sorts the amortization events ascending by date.

Returns

void

Sample

```
//Sorts the amortization events ascending by date.

plugins.amortization.sortEvents();
```