A Divine Data Comedy in JavaScript

Mike Harris

Midway upon the journey of our life
I found myself within a forest dark,
For the straightforward pathway had been lost.

–Dante (tr. Longfellow), Inferno
Canto I, Lines 1-3

Canto

- Data Flow in a System
- The Language of Data Flow Manipulations
- Data Flow in Context

Canto

- **Data Flow in a System**
- The Language of Data Flow Manipulations
- Data Flow in Context

Example Data Set

```sql
create table stock_price (
    symbol varchar(5) not null
   ,price_date date not null
   ,price money not null
);

insert into stock_price
(symbol, price_date, price)
values
 ('ZVZZT', '2017-01-03', 10.02)
,('CBO'  , '2017-01-03', 18.99)
,('ZVZZT', '2017-01-04',  9.99)
,('CBO'  , '2017-01-04', 19.01)
;
```

Example Data Set

Symbol	Price Date	Price
ZVZZT	2017-01-03	10.02
CBO	2017-01-03	18.99
ZVZZT	2017-01-04	9.99
CBO	2017-01-04	19.01

SQL Statement

```sql
select symbol, price
  from stock_price
  where symbol = 'ZVZZT'
  order by price_date
;
```

Symbol	Price Date	Price
ZVZZT	2017-01-03	10.02
ZVZZT	2017-01-04	9.99

Logical Order of a T-SQL Statement

1. FROM
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH ROLLUP
7. HAVING
8. SELECT
9. DISTINCT
10. ORDER BY
11. TOP

Logical Order of a T-SQL Statement

1. FROM
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH ROLLUP
7. HAVING
8. SELECT
9. DISTINCT
10. ORDER BY
11. TOP

Extract

Aggregate

Map

Order

Filter

Logical Order of a T-SQL Statement

1. **FROM**
2. ON
3. JOIN
4. **WHERE**
5. GROUP BY
6. WITH ROLLUP
7. HAVING
8. **SELECT**
9. DISTINCT
10. **ORDER BY**
11. TOP

SQL Statement Data Flow

SQL Statement

```
select symbol, price
  from stock_price
  where symbol = 'ZVZZT'
  order by price_date
;
```

Symbol	Price Date	Price
ZVZZT	2017-01-03	10.02
ZVZZT	2017-01-04	9.99

Folded SQL

Folded SQL Statement

```
select max(price_date) as price_date
  from stock_price
;
```

Price Date
2017-01-04

Logical Order of a T-SQL Statement

1. **FROM**
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH ROLLUP
7. HAVING
8. **SELECT**
9. DISTINCT
10. ORDER BY
11. TOP

Folded SQL Statement Data Flow

Folded SQL Statement

```
select max(price_date) as price_date
  from stock_price
;
```

Price Date
2017-01-04

Group By Folded SQL

Group By Folded SQL Statement

```sql
select
   symbol
  ,max(price_date) as price_date
  from stock_price
  group by symbol
;
```

Symbol	Price Date
CBO	2017-01-04
ZVZZT	2017-01-04

Logical Order of a T-SQL Statement

1. **FROM**
2. ON
3. JOIN
4. WHERE
5. **GROUP BY**
6. WITH ROLLUP
7. HAVING
8. **SELECT**
9. DISTINCT
10. ORDER BY
11. TOP

Group By Folded SQL Statement Data Flow

Group By Folded SQL Statement

```
3  select
       symbol
      ,max(price_date) as price_date
1     from stock_price
2     group by symbol
;
```

Symbol	Price Date
CBO	2017-01-04
ZVZZT	2017-01-04

Window Folded SQL

Window Folded SQL Statement

```sql
select distinct
   symbol
  ,max(price_date) over(
    partition by symbol) as price_date
  from stock_price
;
```

Symbol	Price Date
CBO	2017-01-04
ZVZZT	2017-01-04

Logical Order of a T-SQL Statement

1. **FROM**
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH ROLLUP
7. HAVING
8. **SELECT**
9. **DISTINCT**
10. ORDER BY
11. TOP

Window SQL Statement Data View

```
max(price_date) over(
 partition by symbol) as price_date
```

Symbol	Price_Date
CBO	2017-01-03
CBO	2017-01-04

Price_Date
2017-01-04
2017-01-04

Window SQL Statement Data Flow

Window Folded SQL Statement

```
  select distinct
2  symbol
   ,max(price_date) over(
     partition by symbol) as price_date
1 from stock_price
;
```

Symbol	Price Date
CBO	2017-01-04
ZVZZT	2017-01-04

Window SQL

Window SQL Statement

```sql
select
    symbol
   ,price
   ,lag(price, 1) over (
      partition by symbol
      order by price_date) as prev_price
   ,price - lag(price, 1) over (
      partition by symbol
      order by price_date) as price_change
   ,lead(price, 1) over (
     partition by symbol
     order by price_date) as next_price
   from stock_price
;
```

Window SQL Statement

Symbol	Price	prev_price	price_change	next_price
CBO	18.99	(null)	(null)	19.01
CBO	19.01	18.99	0.02	(null)
ZVZZT	10.02	(null)	(null)	9.99
ZVZZT	9.99	10.02	-0.03	(null)

Window SQL Statement Data View

```sql
lag(price, 1) over (
  partition by symbol
  order by price_date) as prev_price
```

Symbol	Price_Date	Price
CBO	2017-01-03	18.99
CBO	2017-01-04	19.01

prev_price
(null)
18.99

Window SQL Statement Data View

```sql
lead(price, 1) over (
  partition by symbol
  order by price_date) as next_price
```

Symbol	Price_Date	Price
CBO	2017-01-03	18.99
CBO	2017-01-04	19.01

next_price
19.01
(null)

Logical Order of a T-SQL Statement

1. **FROM**
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. WITH ROLLUP
7. HAVING
8. **SELECT**
9. DISTINCT
10. ORDER BY
11. TOP

Window SQL Statement Data Flow

Window SQL Statement

```
select
  symbol
 ,price
 ,lag(price, 1) over (
    partition by symbol
    order by price_date) as prev_price
 ,price - lag(price, 1) over (
    partition by symbol
    order by price_date) as price_change
 ,lead(price, 1) over (
   partition by symbol
   order by price_date) as next_price
 from stock_price
;
```

2

1

Canto

- Data Flow in a System
- **The Language of Data Flow Manipulations**
- Data Flow in Context

Imperative Example

```
var _ = require('lodash');
```

Imperative Example

```
const s =
  ‘Midway in our life\'s journey, I went astray';
var count = 0;
for(var word of _.split(s, ' ')) {
  var stripped = _.replace(word, '(\'s)|\W+', '');
  if (stripped.length > 2)
    count++;
}
```

Imperative Data Flow

Nested Example

```
var _ = require('lodash');
```

Nested Example

```
const s =
  'Midway in our life\'s journey, I went astray';
const count = _.size(
  _.filter(
    _.map(
      _.split(s, ' '),
      word => _.replace(word, '(\'s)|\W+', '')
    ),
    stripped => _.gt(stripped.length, 2)
  )
);
```

Nested Data Flow

Piped Example

```
var { flow, map, filter, split, size, replace } = require('lodash/fp');
```

Piped Example

```
const count = flow([
  split(' '),
  map(
    replace('(\'s)|\W+')('')),
  filter(
    stripped => _.gt(stripped.length, 2)),
  size
])('Midway in our life\'s journey, I went astray');
```

Piped Data Flow

The Language of Data Flow Manipulations

- map
- flat map
- filter
- fold
- mutate
- group by
- order by

The Language of Data Flow Manipulations

- map: **map**
- flat map: **flatMap**
- filter: **filter**
- fold: **reduce**
- mutate: **each**
- group by: **groupBy**
- order by: **orderBy**

The Language of Data Flow Manipulations

- **map**
- flat map
- filter
- fold
- mutate
- group by
- order by

Map

Map Example

```
const numbers =
_.map([3, 9, 10], x => x * 10);
```

```
output>
  [30, 90, 100]
```

Map Data Flow

map Source Code

```
function map(array, iteratee) {
  let index = -1
  const length = array == null ? 0 : array.length
  const result = new Array(length)


  while (++index < length) {
    result[index] = iteratee(array[index], index, array)
  }
  return result
}
```

https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/map.js#L19-L28

Map Example

```
const numbers =
_.map([3, 9, 10], x => x * 10);
```

```
output>
  [30, 90, 100]
```

The Language of Data Flow Manipulations

- map
- **flat map**
- filter
- fold
- mutate
- group by
- order by

Flat Map

Flat Map Example

```
const list = _.flatMap([
    'Midway in our life\'s journey,',
    'I went astray'
  ],
  s => _.split(s, ' ')
);
```

```
output>
  ['Midway', 'in', ‘our',
   'life\'s', 'journey,',
   ‘I', 'went', ‘astray' ]
```

Flat Map Query Syntax Data Flow

flatMap Source Code

```
function flatMap(collection, iteratee) {
  return baseFlatten(map(collection, iteratee), 1)
}
```

https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/flatMap.js#L24-L26

baseFlatten Source Code

```
function baseFlatten(array, depth, predicate, isStrict, result) {
  predicate || (predicate = isFlattenable)
  result || (result = [])


  if (array == null) {
    return result
  }


  for (const value of array) {
    if (depth > 0 && predicate(value)) {
      if (depth > 1) {
        // Recursively flatten arrays (susceptible to call stack limits).
        baseFlatten(value, depth - 1, predicate, isStrict, result)
      } else {
        result.push(...value)
      }
    } else if (!isStrict) {
      result[result.length] = value
    }
  }
  return result
}
```

https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/.internal/baseFlatten.js#L14-L35

baseFlatten Source Code

```
function baseFlatten(array, depth, predicate, isStrict, result) {
  predicate || (predicate = isFlattenable)
  result || (result = [])
  // check for null ..

  for (const value of array) {
    if (depth > 0 && predicate(value)) {
      if (depth > 1) {
        baseFlatten(value, depth - 1, predicate, isStrict, result)
      } else {
        result.push(...value)
      }
    }// else ..
  }
  return result
}
```

https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/.internal/baseFlatten.js#L14-L35

Flat Map Example

```
const list = _.flatMap([
    'Midway in our life\'s journey,',
    'I went astray'
  ],
  s => _.split(s, ' ')
);
```

```
output>
  ['Midway', 'in', ‘our',
   'life\'s', 'journey,',
   ‘I', 'went', ‘astray' ]
```

The Language of Data Flow Manipulations

- map
- flat map
- **filter**
- fold
- mutate
- group by
- order by

Filter

Filter Example

```
const numbers =
  _.filter(
    [3, 9, 10, 33, 100],
    n => n % 2 != 0
  );
```

```
output>
  [3, 9, 33]
```

Filter Data Flow

filter Source Code

```
function filter(array, predicate) {
  let index = -1
  let resIndex = 0
  const length = array == null ? 0 : array.length
  const result = []

  while (++index < length) {
    const value = array[index]
    if (predicate(value, index, array)) {
      result[resIndex++] = value
    }
  }
  return result
}
```

https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/filter.js#L24-L37

Filter Example

```
const numbers =
  _.filter(
    [3, 9, 10, 33, 100],
    n => n % 2 != 0
  );
```

```
output>
  [3, 9, 33]
```

The Language of Data Flow Manipulations

- map
- flat map
- filter
- **fold**
- mutate
- group by
- order by

Fold

Fold Example

```
const sum =
  _.reduce(
    [3, 9, 10, 33, 100],
    (m, x) => m + x,
    0
  );
```

```
output>
155
```

Fold Data Flow

reduce Source Code

```
function reduce(collection, iteratee, accumulator) {
  const func = Array.isArray(collection)
    ? arrayReduce : baseReduce
  const initAccum = arguments.length < 3
  return func(collection, iteratee, accumulator,
              initAccum, baseEach)
}
```

https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/reduce.js#L38-L42

arrayReduce Source Code

```
function arrayReduce(array, iteratee, accumulator, initAccum) {
  let index = -1
  const length = array == null ? 0 : array.length

  if (initAccum && length) {
    accumulator = array[++index]
  }
  while (++index < length) {
    accumulator = iteratee(
      accumulator, array[index], index, array)
  }
  return accumulator
}
```

https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/.internal/arrayReduce.js#L12-L23

Fold Example

```
const sum =
  _.reduce(
    [3, 9, 10, 33, 100],
    (m, x) => m + x,
    0
  );
```

```
output>
155
```

The Language of Data Flow Manipulations

- map
- flat map
- filter
- fold
- **mutate**
- group by
- order by

Mutate

Mutate Example

```
let r = [];
_.each(
  [3, 9, 10, 33, 100],
  n => { if (n % 3 === 0) { r.push(n); } }
);
```

```
output>
// void

r = [3, 9, 33]
```

Mutate Data Flow

The Language of Data Flow Manipulations

- map
- flat map
- filter
- fold
- mutate
- **group by**
- order by

Group By

Group By Example

```
const groups =
  _.groupBy(
    [3, 9, 10, 33, 100],
    n => n % 2 === 0
  );
```

```
output>
{'true': [10, 100],
 'false': [3, 9, 33]}
```

Group By Data Flow

The Language of Data Flow Manipulations

- map
- flat map
- filter
- fold
- mutate
- group by
- **order by**

Order By

Order By Example

```
const ordered =
  _.orderBy([33, 9, 100, 3, 10]);
```

```
output>
   [3, 9, 10, 33, 100]
```

Order By Data Flow

Canto

- Data Flow in a System
- The Language of Data Flow Manipulations
- **Data Flow in Context**

You shall leave everything you love most dearly:
this is the arrow that the bow of exile
shoots first. You are to know the bitter taste
of others' bread, how salt it is, and know
how hard a path it is for one who goes
descending and ascending others' stairs.

–Dante (tr. Barolini), Paradiso
Canto XVII, Lines 55-60

Functions in Context

Identity Context

Example of Composed Functions

```
class Identity {
  static of(value) { return [value]; }
}

Identity.of('33')
  .map(toInteger)
  .map(add(1))
  .shift();
```

```
output>
34
```

Example of Composed Functions

Example of Composed Functions

```
Identity.of('33')
.map(toInteger)
.map(add(1))
.shift();
```

Context

Lift

Remove

```
output>
34
```

Identity Context

Identity in Context Example

```
const count = Identity.of(
  ‘Midway in our life\'s journey, I went astray')
  .map(split(' '))
  .reduce(concat)
  .map(replace('(\'s)|\W+')(''))
  .filter(word => _.gt(word.length, 2))
  .reduce(number => ++number, 0);
```

```
output>
6
```

Identity in Context Data Flow

Identity in Context Example

Lift

```
const count = Identity.of(
  'Midway in our life\'s journey, I went astray')
  .map(split(' '))
  .reduce(concat)
  .map(replace('(\'s)|\W+')(''))
  .filter(word => _.gt(word.length, 2))
  .reduce(number => ++number, 0);
```

Context

Remove

```
output>
6
```

String Context

String in Context Example

```
const count =
  ‘Midway in our life\'s journey, I went astray'
  .split(' ')
  .map(replace('(\'s)|\W+')(''))
  .filter(word => _.gt(word.length, 2))
  .reduce(number => ++number, 0);
```

```
output>
6
```

String in Context Example

Lift

```
const count =
  'Midway in our life\'s journey, I went astray'
  .split(' ')
  .map(replace('(\'s)|\W+')(''))
  .filter(word => _.gt(word.length, 2))
  .reduce(number => ++number, 0);
```

Context

Remove

```
output>
6
```

Maybe Context

If Context Example

Lift

```
const size = s => {
  if (s) {
    return s.trim().length;
  } else {
    return 0;
  }
};
```

Context

```
const Maybe = require('data.maybe');
```

Maybe Context Example

Lift

```
const size = s =>
  Maybe.fromNullable(s)
  .map(s => s.trim())
  .cata({
    Nothing: _ => 0,
    Just: t => t.length});
```

Context

Remove

Maybe in Context Data Flow

Maybe Context

```
const Maybe = require('data.maybe');
```

Maybe Context Example

Lift

```
const size = s =>
  Maybe.fromNullable(s)
  .map(s => s.trim())
  .cata({
    Nothing: _ => 0,
    Just: t => t.length});
```

Context

Remove

Maybe in Context Examples

```
counter(
  'Midway in our life\'s journey, I went astray’);
```

```
output>
6
```

```
counter(null);
```

```
output>
0
```

```
const Maybe = require('data.maybe');
```

Maybe in Context Example

```
const counter = s =>
  Maybe.fromNullable(s)
    .map(s => s.split(' ')
               .map(replace('(\'s)|\W+')(''))
               .filter(w => _.gt(w.length, 2)))
    .cata({
      Nothing: _ => 0,
      Just: words => words.length
    });
```

```
const Maybe = require('data.maybe');
```

Maybe in Context Example

```
const counter = s =>
  Maybe.fromNullable(s)
    .map(s => s.split(' ')
                .map(replace('(\'s)|\W+')(''))
                .filter(w => _.gt(w.length, 2)))
    .cata({
      Nothing: _ => 0,
      Just: words => words.length
    });
```

Lift

Context

Remove

Maybe in Context Data Flow

```
const Maybe = require('data.maybe');
```

Maybe in Context Example

```
const counter = s =>
  Maybe.fromNullable(s)
    .map(s => s.split(' ')
                .map(replace('(\'s)|\W+')(''))
                .filter(w => _.gt(w.length, 2)))
    .cata({
      Nothing: _ => 0,
      Just: words => words.length
    });
```

Lift

Context

Remove

List Context

List Context Example

Lift

```
const count = flow([
  split(' '),
  map(
    replace('(\'s)|\W+')('')),
  filter(
    stripped => _.gt(stripped.length, 2)),
  size
])
('Midway in our life\'s journey, I went astray');
```

Context

Remove

List in Context Data Flow

List Context Example

Lift

```
const count = flow([
  split(' '),
  map(
    replace('(\'s)|\W+')('')),
  filter(
    stripped => _.gt(stripped.length, 2)),
  size
])
('Midway in our life\'s journey, I went astray');
```

Context

Remove

QVI COELVM CEPIT MEDIVMQVE IMVMQVE TRIBVNAL
LVSTRAVIT QVE ANIMO CVNCTA POETA SVO
DOCTVS ADEST DANTES SVA QVEM FLORENTIA SAEPE
SENSIT CONSILIIS AC PIETATE PATREM
NIL POTVIT TANTO MORS SAEVA NOCERE POETAE
QVEM VIVVM VIRTVS CARMEN IMAGO FACIT

Thank you!

Mike Harris

@MikeMKH
http://comp-phil.blogspot.com/

https://github.com/MikeMKH/talks/tree/master/a-divine-data-comedy-in-javascript

Next Steps

- StrangeLoop (conference)
  https://www.thestrangeloop.com/

- Brian Lonsdorf - Professor Frisby Introduces Composable Functional JavaScript (video series)
  https://egghead.io/courses/professor-frisby-introduces-composable-functional-javascript

- Aditya Bhargava - Functors, Applicatives, And Monads In Pictures (blog post)
  http://adit.io/posts/2013-04-17-functors,_applicatives,_and_monads_in_pictures.html

- Luis Atencio - Functional Programming in JavaScript (book)
  https://www.manning.com/books/functional-programming-in-javascript

- Reginald Braithwaite - JavaScript Allongé, The “Six” Edition (book)
  https://leanpub.com/javascriptallongesix/read

Code from Lodash Source Code

- map
  - https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/map.js#L19-L28
- flatMap
  - https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/flatMap.js#L24-L26
  - https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/.internal/baseFlatten.js#L14-L35
- filter
  - https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/filter.js#L24-L37
- reduce
  - https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/reduce.js#L38-L42
  - https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/.internal/arrayReduce.js#L12-L23

Code from Lodash Source Code

- foreach
  - https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/forEach.js#L28-L31
  - https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/.internal/arrayEach.js#L9-L19
- groupBy
  - https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/groupBy.js#L24-L34
- orderBy
  - https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/orderBy.js#L30-L41
  - https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/.internal/baseOrderBy.js#L14-L24
  - https://github.com/lodash/lodash/blob/6ad829fa90af199150b11ba1d3c944b648a39ce5/.internal/baseSortBy.js#L11-L19

Code

- SQL Code, https://github.com/MikeMKH/talks/blob/master/a-divine-data-comedy-in-javascript/example.sql

- JavaScript Code, https://github.com/MikeMKH/talks/blob/master/a-divine-data-comedy-in-javascript/test.js

Yes, Lodash’s source code does not have ;

- https://github.com/lodash/lodash/commit/6cb3460fcefe66cb96e55b82c6febd2153c992cc

Biography

- Barolini, Teodolinda. Commento Baroliniano, Digital Dante. New York, NY: Columbia University Libraries, 2017. https://digitaldante.columbia.edu/dante/divine-comedy/
- Ben-Gan, Itzik. Microsoft SQL Server 2012 high-performance T-SQL using Window functions: Microsoft Press, 2012.
- Byham, Rick. Microsoft Docs. https://docs.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql
- Cook, William R. and Herzman, Ronald B. Dante's Divine Comedy. The Great Courses. http://www.thegreatcourses.com/courses/dante-s-divine-comedy.html
- Equity Regulatory Alert #2016 - 3 Guidance On Test Stock Usage https://www.nasdaqtrader.com/MicroNews.aspx?id=ERA2016-3
- Petricek, Tomas. Beyond the Monad fashion (I.): Writing idioms in LINQ- http://tomasp.net/blog/idioms-in-linq.aspx/
- Securities Industry Automation Corporation. New York, 10 September 2015. https://www.nyse.com/publicdocs/ctaplan/notifications/trader-update/CTS_CQS%20%20NEW%20DEDICATED%20TEST%20SYMBOL_09102015.pdf
- Wickham, Hadley, and Garrett Grolemund. R for data science : import, tidy, transform, visualize, and model data. Sebastopol, CA: O'Reilly Media, 2016. http://r4ds.had.co.nz/
- Quil. (15 October 2017) “A Monad in Practicality: First-Class Failures” [blog post]. http://robotlolita.me/2013/12/08/a-monad-in-practicality-first-class-failures.html

Images

- By Domenico di Michelino - Jastrow, Self-photographed, Public Domain, https://commons.wikimedia.org/w/index.php?curid=970608
- By Sailko - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=49841035
- By Lua - Sotheby's, London, 17 December 2015, lot 22, Public Domain, https://commons.wikimedia.org/w/index.php?curid=32782393
- By Sailko - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=56266161
- By William Blake - http://www.blakearchive.org/exist/blake/archive/work.xq?workid=but812&java=no, Public Domain, https://commons.wikimedia.org/w/index.php?curid=27118518
- By Salvatore Postiglione - http://www.hampel-auctions.com/en/92-325/onlinecatalog-detail-n229.html, Public Domain, https://commons.wikimedia.org/w/index.php?curid=26540023
- Di Giovanni Stradano - Opera propria, 2007-10-25, Pubblico dominio, https://commons.wikimedia.org/w/index.php?curid=2981981