
nhl

Matt Hostetter

Sep 17, 2020

CONTENTS

1	API	1
1.1	nhl	1
1.2	nhl.rink	13
1.3	nhl.statsapi	14
2	Examples	31
3	Indices and tables	33
	Python Module Index	35
	Index	37

nhl
nhl.rink
nhl.statsapi

1.1 nhl

Classes

<i>Conference</i> (*args, **kwargs)	NHL conference object.
<i>Division</i> (*args, **kwargs)	NHL division object.
<i>Event</i> (*args, **kwargs)	NHL event object.
<i>Franchise</i> (*args, **kwargs)	NHL franchise object.
<i>Game</i> (*args, **kwargs)	NHL game object.
<i>GameInfo</i> (id, season_id, type, start, end, ...)	NHL game info.
<i>Gametime</i> (*args, **kwargs)	NHL gametime object.
<i>List</i> ([iterable])	Searchable, sortable, and filter-able <code>list</code> subclass
<i>Location</i> (*args, **kwargs)	NHL location object.
<i>Official</i> (*args, **kwargs)	NHL official object.
<i>Player</i> (*args, **kwargs)	NHL player object.
<i>PlayerStats</i> (player, shifts, events)	NHL player statistics.
<i>Shift</i> (*args, **kwargs)	NHL shift object.
<i>Team</i> (*args, **kwargs)	NHL team object.
<i>Venue</i> (*args, **kwargs)	NHL venue object.

class `nhl.Conference` (*args, **kwargs)

Bases: `nhl.flyweight.Flyweight`

NHL conference object.

This is the detailed docstring.

classmethod `from_key` (id)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or `None` if key not found

Return type `cls` or `None`

classmethod `has_key` (id)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

abbreviation: str

Conference abbreviation

Type str

id: int

The NHL statsapi universal conference ID

Type int

name: str

Conference name

Type str

name_short: str

Conference name shortened

Type str

class nhl.Division(*args, **kwargs)

Bases: nhl.flyweight.Flyweight

NHL division object.

This is the detailed docstring.

classmethod from_key(*id*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

abbreviation: str

Division abbreviated name

Type str

id: int

The NHL statsapi universal division ID

Type int

name: str

Division name

Type str

name_short: str

Division short name

Type int

```

class nhl.Event (*args, **kwargs)
    Bases: nhl.flyweight.Flyweight

    NHL event object.

    This is the detailed docstring.

    classmethod from_key (game_id, id)
        Return flyweight object with specified key, if it has already been created.

        Returns Previously constructed flyweight object with given key or None if key not found

        Return type cls or None

    classmethod has_key (game_id, id)
        Check whether flyweight object with specified key has already been created.

        Returns True if already created, False if not

        Return type bool

    by_player: nhl.player.Player
    by_players_on_ice: nhl.list.List
    property by_strength
    by_team: nhl.team.Team
    game_id: int
    gametime: nhl.gametime.Gametime
    id: int
    property lead
    location: nhl.location.Location
    property name
    on_player: nhl.player.Player
    on_players_on_ice: nhl.list.List
    property on_strength
    on_team: nhl.team.Team
    score: tuple
    property strength
    property subname
    subtype: str
    type: str
    value: float
    property valuenam

class nhl.Franchise (*args, **kwargs)
    Bases: nhl.flyweight.Flyweight

    NHL franchise object.

    This is the detailed docstring.

```

classmethod `from_key` (*id*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod `has_key` (*id*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

id: int

The NHL statsapi universal franchise ID

Type int

name: str

Franchise name

Type str

class `nhl.Game` (*args, **kwargs)

Bases: `nhl.flyweight.Flyweight`

NHL game object.

This is the detailed docstring.

classmethod `from_key` (*id*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod `has_key` (*id*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

away: `nhl.team.Team`

Game away

Type *Team*

property `defensemen`

events: `nhl.list.List`

List:

property `forwards`

property `goalies`

home: `nhl.team.Team`

Game home

Type *Team*

info: `nhl.gameinfo.GameInfo`

Game info

Type *GameInfo*

players: `nhl.list.List`

List:

property skaters

```
class nhl.GameInfo(id: int, season_id: int, type: str, start: datetime.datetime, end: datetime.datetime, venue: nhl.venue.Venue, home_team: nhl.team.Team, away_team: nhl.team.Team, score: tuple, end_type: str, end_time: nhl.gametime.Gametime, referees: nhl.list.List, linesmen: nhl.list.List)
```

Bases: object

NHL game info.

This is the detailed docstring.

classmethod from_key (*id*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or *None* if key not found

Return type *nhl.Game* or None

classmethod has_key (*id*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

property away_score

away_team: `nhl.team.Team`

property date

property description

end: `datetime.datetime`

end_time: `nhl.gametime.Gametime`

end_type: `str`

property home_score

home_team: `nhl.team.Team`

id: `int`

linesmen: `nhl.list.List`

property playoff_game

property playoff_round

property playoff_series

referees: `nhl.list.List`

score: `tuple`

property season

season_id: `int`

property season_type

start: `datetime.datetime`

type: `str`

venue: `nhl.venue.Venue`

class `nhl.Gametime` (**args, **kwargs*)
Bases: `nhl.flyweight.Flyweight`

NHL gametime object.

This object represents a unique time of the game. There are convenience properties to convert the gametime into convenient formats.

Parameters

- **period** (*int*) – The period of the game.
- **period_sec** (*int*) – The number of elapsed seconds in the period.

classmethod `from_key` (*period, period_sec*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type `cls` or None

classmethod `has_key` (*period, period_sec*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type `bool`

property `min_sec`

Elapsed minutes and seconds of the game.

Type (`int, int`)

period: `int`

Game period. 1-3 for regulation. 4+ for overtime.

Type `int`

property `period_min_sec`

Elapsed minutes and seconds of the period.

Type (`int, int`)

period_sec: `int`

Elapsed seconds of the period.

Type `int`

property `period_str`

Period number as string (i.e. “2nd”)

Type `str`

property `sec`

Elapsed seconds of the game.

Type `int`

class `nhl.List` (*iterable=(, /)*)

Bases: `list`

Searchable, sortable, and filter-able `list` subclass

filter (*attr, value, compare='=='*)

Filter list by a comparison of a given attribute (or chain or attributes).

Parameters

- **attr** (*str*) – attributes to be compared (with initial . omitted)
- **value** (*any*) – value to compare attr against
- **compare** (*str*) – comparison type “==”, “=”, “eq”: *attr == value* “!=”, “ne”: *attr != value* “>”, “gt”: *attr > value* “>=”, “ge”: *attr >= value* “<”, “lt”: *attr < value* “<=”, “le”: *attr <= value* “in”, “contains”: *value in attr*

Returns reduced list with items that satisfy filter criterion

Return type *nhl.List*

select (*attr, default=None*)

Select a given attribute (or chain or attributes) from the objects within the list.

Parameters

- **attr** (*str*) – attributes to be selected (with initial . omitted)
- **default** (*any*) – value to return if given element in list doesn’t contain desired attribute

Returns list of selected attribute values

Return type *nhl.List*

sort (*key, reverse=False*)

Sort the list by *key* either ascending or descending.

Parameters

- **key** (*str*) – attribute to sort by (with initial . omitted)
- **reverse** (*bool, optional*) – reverse the direction of sort to descending, (default False, ascending)

Returns sorted list

Return type *nhl.List*

unique ()

Reduce the list to unique elements.

Returns reduced list

Return type *nhl.List*

property len

Helper property to return length of list

Returns length of list *len(self)*

Return type int

class `nhl.Location` (**args, **kwargs*)

Bases: `nhl.flyweight.Flyweight`

NHL location object.

This is the detailed docstring.

distance (*other*)

Measure distance between current location and another on-ice location.

Parameters **other** (`Location`) – location to measure distance from

Returns distance (ft)

Return type float

classmethod from_key (*x*, *y*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key (*x*, *y*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

x: int

Rink position in ft along x-axis (length) [-100, 100]

Type int

y: int

Rink position in ft along y-axis (breadth) [-42, 42]

Type int

class nhl.Official (**args*, ***kwargs*)

Bases: nhl.flyweight.Flyweight

NHL official object.

This is the detailed docstring.

classmethod from_key (*id*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or *None* if key not found

Return type nhl.Official or None

classmethod has_key (*id*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

property first_name

Official's first name

Type str

id: int

The NHL statsapi universal official ID

Type int

property last_name

Official's last name

Type str

name: str

Official's full name

Type str

```
class nhl.Player (*args, **kwargs)
    Bases: nhl.flyweight.Flyweight
    NHL player object.
    This is the detailed docstring.
classmethod from_key (id)
    Return flyweight object with specified key, if it has already been created.
        Returns Previously constructed flyweight object with given key or None if key not found
        Return type cls or None
classmethod has_key (id)
    Check whether flyweight object with specified key has already been created.
        Returns True if already created, False if not
        Return type bool
property age
    Current age in years
        Type int
birth_city: str
    Player's birth city
        Type str
birth_country: str
    Player's birth country
        Type str
birth_date: datetime.date
    Player's birth date
        Type datetime.date
property first_name
    Player's first name
        Type str
height: int
    Player's height in total inches
        Type int
property height_ft_in
    Height in feet and inches (height // 12, height % 12)
        Type int
id: int
    The NHL statsapi universal player ID
        Type int
property last_name
    Player's last name
        Type str
```

name: str
Player's full name
Type str

number: int
Player's primary number
Type int

position: str
Player's primary position ("LW", "C", "RW", "D", "G")
Type str

shoots_catches: str
Indication of whether the player shoots (skater)/catches (goalie) "L" or "R"
Type str

weight: int
Player's weight in lbs
Type int

class nhl.**PlayerStats** (*player: nhl.player.Player, shifts: nhl.list.List, events: nhl.list.List*)
Bases: object

NHL player statistics.

This is the detailed docstring.

events: nhl.list.List
Player's events
Type *List[Event]*

player: nhl.player.Player
Player object
Type *Player*

shifts: nhl.list.List
Player's shifts
Type *List[Shift]*

class nhl.**Shift** (**args, **kwargs*)
Bases: nhl.flyweight.Flyweight

NHL shift object.

This is the detailed docstring.

classmethod **from_key** (*game_id, player_id, shift_id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod **has_key** (*game_id, player_id, shift_id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

game_id: int
NHL statsapi unique game ID
Type int

property length

off: int
Shift end gametime
Type *Gametime*

on: int
Shift start gametime
Type *Gametime*

property player

player_id: int
NHL statsapi unique player ID
Type int

shift_id: int
Shift number for specified game
Type int

class nhl.**Team**(*args, **kwargs)
Bases: nhl.flyweight.Flyweight
NHL team object.
This is the detailed docstring.

classmethod **from_key**(id)
Return flyweight object with specified key, if it has already been created.
Returns Previously constructed flyweight object with given key or None if key not found
Return type cls or None

classmethod **has_key**(id)
Check whether flyweight object with specified key has already been created.
Returns True if already created, False if not
Return type bool

abbreviation: str
Team's name abbreviated
Type int

conference: nhl.conference.Conference
Conference:

division: nhl.division.Division
Division:

first_year: int
First year of play
Type int

franchise: nhl.franchise.Franchise

Franchise:

property full_name

Team's full name

Type str

id: int

The NHL statsapi universal team ID

Type int

location: str

Team's location

Type str

name: str

Team's name

Type str

class nhl.Venue (*args, **kwargs)

Bases: nhl.flyweight.Flyweight

NHL venue object.

This is the detailed docstring.

classmethod from_key (id)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key (id)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

id: int

The NHL statsapi universal venue ID

Type int

name: str

Venue name

Type str

1.2 nhl.rink

Module Attributes

<i>LENGTH</i>	Rink length (ft)
<i>WIDTH</i>	Rink width (ft)
<i>DX</i>	Delta length from center ice (ft)
<i>DY</i>	Delta width from center ice (ft)
<i>BLUE_LINE_X</i>	Blue line x-position from center ice (ft)
<i>GOAL_LINE_X</i>	Goal line x-position from center ice (ft)
<i>NZ_FACEOFF_DOT_X</i>	Neutral zone faceoff dot x-position from center ice (ft)
<i>OZ_FACEOFF_DOT_X</i>	Offensive zone faceoff dot x-position from center ice (ft)
<i>FACEOFF_DOT_Y</i>	All faceoff dot y-position from center ice (ft)
<i>FACEOFF_DOTS</i>	Tuple of faceoff dots (x, y)

Classes

<i>Location</i> (*args, **kwargs)	NHL location object.
-----------------------------------	----------------------

class nhl.rink.**Location** (*args, **kwargs)

Bases: nhl.flyweight.Flyweight

NHL location object.

This is the detailed docstring.

distance (*other*)

Measure distance between current location and another on-ice location.

Parameters *other* (*Location*) – location to measure distance from

Returns distance (ft)

Return type float

classmethod **from_key** (*x*, *y*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod **has_key** (*x*, *y*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

x: **int**

Rink position in ft along x-axis (length) [-100, 100]

Type int

y: **int**

Rink position in ft along y-axis (breadth) [-42, 42]

Type int

`nhl.rink.BLUE_LINE_X = 25`

Blue line x-position from center ice (ft)

`nhl.rink.DX = 100.0`

Delta length from center ice (ft)

`nhl.rink.DY = 42.5`

Delta width from center ice (ft)

`nhl.rink.FACEOFF_DOTS = ((0, 0), (20, 22), (20, -22), (-20, 22), (-20, -22), (69, 22), (69, -22))`

Tuple of faceoff dots (x, y)

`nhl.rink.FACEOFF_DOT_Y = 22`

All faceoff dot y-position from center ice (ft)

`nhl.rink.GOAL_LINE_X = 89`

Goal line x-position from center ice (ft)

`nhl.rink.LENGTH = 200`

Rink length (ft)

`nhl.rink.NZ_FACEOFF_DOT_X = 20`

Neutral zone faceoff dot x-position from center ice (ft)

`nhl.rink.OZ_FACEOFF_DOT_X = 69`

Offensive zone faceoff dot x-position from center ice (ft)

`nhl.rink.WIDTH = 85`

Rink width (ft)

1.3 nhl.statsapi

Functions

conference(id)

conferences()

division(id)

divisions()

franchise(id)

franchises()

game(id)

parse_conference(json)

parse_conferences(json)

parse_division(json)

parse_events(json, info, home_score, ...)

parse_franchise(json)

parse_game(json)

parse_location(json, gametime, flip_sides)

parse_official(json)

parse_player(json)

parse_shifts(game_id, team_id, player_id, ...)

parse_team(json)

parse_teams(json)

continues on next page

Table 5 – continued from previous page

<code>parse_venue(json)</code>
<code>player(id)</code>
<code>players(ids)</code>
<code>team(id)</code>
<code>teams([ids])</code>
<code>venue(id)</code>
<code>venues()</code>

Classes

<code>BeautifulSoup([markup, features, builder, ...])</code>	A data structure representing a parsed HTML or XML document.
<code>Conference(*args, **kwargs)</code>	NHL conference object.
<code>Division(*args, **kwargs)</code>	NHL division object.
<code>Event(*args, **kwargs)</code>	NHL event object.
<code>Franchise(*args, **kwargs)</code>	NHL franchise object.
<code>Game(*args, **kwargs)</code>	NHL game object.
<code>GameInfo(id, season_id, type, start, end, ...)</code>	NHL game info.
<code>Gametime(*args, **kwargs)</code>	NHL gametime object.
<code>List([iterable])</code>	Searchable, sortable, and filter-able <code>list</code> subclass
<code>Location(*args, **kwargs)</code>	NHL location object.
<code>Official(*args, **kwargs)</code>	NHL official object.
<code>Player(*args, **kwargs)</code>	NHL player object.
<code>PlayerStats(player, shifts, events)</code>	NHL player statistics.
<code>Shift(*args, **kwargs)</code>	NHL shift object.
<code>Team(*args, **kwargs)</code>	NHL team object.
<code>Venue(*args, **kwargs)</code>	NHL venue object.

```
class nhl.statsapi.BeautifulSoup (markup='', features=None, builder=None, parse_only=None,
                                from_encoding=None, exclude_encodings=None, element_classes=None, **kwargs)
```

Bases: `bs4.element.Tag`

A data structure representing a parsed HTML or XML document.

Most of the methods you'll call on a BeautifulSoup object are inherited from PageElement or Tag.

Internally, this class defines the basic interface called by the tree builders when converting an HTML/XML document into a data structure. The interface abstracts away the differences between parsers. To write a new tree builder, you'll need to understand these methods as a whole.

These methods will be called by the BeautifulSoup constructor:

- `reset()`
- `feed(markup)`

The tree builder may call these methods from its feed() implementation:

- `handle_starttag(name, attrs)` # See note about return value
- `handle_endtag(name)`
- `handle_data(data)` # Appends to the current data node
- `endData(containerClass)` # Ends the current data node

No matter how complicated the underlying parser is, you should be able to build a tree using ‘start tag’ events, ‘end tag’ events, ‘data’ events, and “done with data” events.

If you encounter an empty-element tag (aka a self-closing tag, like HTML’s `
` tag), call `handle_starttag` and then `handle_endtag`.

decode (*pretty_print=False, eventual_encoding='utf-8', formatter='minimal'*)

Returns a string or Unicode representation of the parse tree as an HTML or XML document.

Parameters

- **pretty_print** – If this is True, indentation will be used to make the document more readable.
- **eventual_encoding** – The encoding of the final document. If this is None, the document will be a Unicode string.

endData (*containerClass=None*)

Method called by the TreeBuilder when the end of a data segment occurs.

handle_data (*data*)

Called by the tree builder when a chunk of textual data is encountered.

handle_endtag (*name, nsprefix=None*)

Called by the tree builder when an ending tag is encountered.

Parameters

- **name** – Name of the tag.
- **nsprefix** – Namespace prefix for the tag.

handle_starttag (*name, namespace, nsprefix, attrs, sourceline=None, sourcepos=None*)

Called by the tree builder when a new tag is encountered.

Parameters

- **name** – Name of the tag.
- **nsprefix** – Namespace prefix for the tag.
- **attrs** – A dictionary of attribute values.
- **sourceline** – The line number where this tag was found in its source document.
- **sourcepos** – The character position within *sourceline* where this tag was found.

If this method returns None, the tag was rejected by an active SoupStrainer. You should proceed as if the tag had not occurred in the document. For instance, if this was a self-closing tag, don’t call `handle_endtag`.

insert_after (*successor*)

This method is part of the PageElement API, but *BeautifulSoup* doesn’t implement it because there is nothing before or after it in the parse tree.

insert_before (*successor*)

This method is part of the PageElement API, but *BeautifulSoup* doesn’t implement it because there is nothing before or after it in the parse tree.

new_string (*s, subclass=None*)

Create a new NavigableString associated with this BeautifulSoup object.

new_tag (*name, namespace=None, nsprefix=None, attrs={}, sourceline=None, sourcepos=None, **kwattrs*)

Create a new Tag associated with this BeautifulSoup object.

Parameters

- **name** – The name of the new Tag.
- **namespace** – The URI of the new Tag’s XML namespace, if any.
- **prefix** – The prefix for the new Tag’s XML namespace, if any.
- **attrs** – A dictionary of this Tag’s attribute values; can be used instead of *kwattrs* for attributes like ‘class’ that are reserved words in Python.
- **sourceline** – The line number where this tag was (purportedly) found in its source document.
- **sourcepos** – The character position within *sourceline* where this tag was (purportedly) found.
- **kwattrs** – Keyword arguments for the new Tag’s attribute values.

object_was_parsed (*o*, *parent=None*, *most_recent_element=None*)

Method called by the TreeBuilder to integrate an object into the parse tree.

popTag ()

Internal method called by *_popToTag* when a tag is closed.

pushTag (*tag*)

Internal method called by *handle_starttag* when a tag is opened.

reset ()

Reset this object to a state as though it had never parsed any markup.

string_container (*base_class=None*)

ASCII_SPACES = ' \n\t\x0c\r'

DEFAULT_BUILDER_FEATURES = ['html', 'fast']

NO_PARSER_SPECIFIED_WARNING = 'No parser was explicitly specified, so I\'m using the b

ROOT_TAG_NAME = '[document]'

class nhl.statsapi.Conference (*args, **kwargs)

Bases: nhl.flyweight.Flyweight

NHL conference object.

This is the detailed docstring.

classmethod from_key (*id*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key (*id*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

abbreviation: str

Conference abbreviation

Type str

id: int
The NHL statsapi universal conference ID

Type int

name: str
Conference name

Type str

name_short: str
Conference name shortened

Type str

class nhl.statsapi.**Division** (*args, **kwargs)

Bases: nhl.flyweight.Flyweight

NHL division object.

This is the detailed docstring.

classmethod from_key (id)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key (id)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

abbreviation: str

Division abbreviated name

Type str

id: int

The NHL statsapi universal division ID

Type int

name: str

Division name

Type str

name_short: str

Division short name

Type int

class nhl.statsapi.**Event** (*args, **kwargs)

Bases: nhl.flyweight.Flyweight

NHL event object.

This is the detailed docstring.

classmethod from_key (game_id, id)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod `has_key` (*game_id*, *id*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

by_player: nhl.player.Player

by_players_on_ice: nhl.list.List

property `by_strength`

by_team: nhl.team.Team

game_id: int

gametime: nhl.gametime.Gametime

id: int

property `lead`

location: nhl.location.Location

property `name`

on_player: nhl.player.Player

on_players_on_ice: nhl.list.List

property `on_strength`

on_team: nhl.team.Team

score: tuple

property `strength`

property `subname`

subtype: str

type: str

value: float

property `valuenam`

class nhl.statsapi.Franchise (*args, **kwargs)

Bases: nhl.flyweight.Flyweight

NHL franchise object.

This is the detailed docstring.

classmethod `from_key` (*id*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod `has_key` (*id*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

id: int

The NHL statsapi universal franchise ID

Type int

name: str

Franchise name

Type str

class nhl.statsapi.Game(*args, **kwargs)

Bases: nhl.flyweight.Flyweight

NHL game object.

This is the detailed docstring.

classmethod from_key(id)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(id)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

away: nhl.team.Team

Game away

Type Team

property defensemen

events: nhl.list.List

List:

property forwards

property goalies

home: nhl.team.Team

Game home

Type Team

info: nhl.gameinfo.GameInfo

Game info

Type GameInfo

players: nhl.list.List

List:

property skaters

```
class nhl.statsapi.GameInfo(id: int, season_id: int, type: str, start: datetime.datetime,
                             end: datetime.datetime, venue: nhl.venue.Venue, home_team:
                             nhl.team.Team, away_team: nhl.team.Team, score: tuple, end_type:
                             str, end_time: nhl.gametime.Gametime, referees: nhl.list.List, lines-
                             men: nhl.list.List)
```

Bases: object

NHL game info.

This is the detailed docstring.

```
classmethod from_key(id)
```

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or *None* if key not found

Return type *nhl.Game* or None

```
classmethod has_key(id)
```

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

```
property away_score
```

```
away_team: nhl.team.Team
```

```
property date
```

```
property description
```

```
end: datetime.datetime
```

```
end_time: nhl.gametime.Gametime
```

```
end_type: str
```

```
property home_score
```

```
home_team: nhl.team.Team
```

```
id: int
```

```
linesmen: nhl.list.List
```

```
property playoff_game
```

```
property playoff_round
```

```
property playoff_series
```

```
referees: nhl.list.List
```

```
score: tuple
```

```
property season
```

```
season_id: int
```

```
property season_type
```

```
start: datetime.datetime
```

```
type: str
```

```
venue: nhl.venue.Venue
```

class nhl.statsapi.**Gametime** (*args, **kwargs)

Bases: nhl.flyweight.Flyweight

NHL gametime object.

This object represents a unique time of the game. There are convenience properties to convert the gametime into convenient formats.

Parameters

- **period** (*int*) – The period of the game.
- **period_sec** (*int*) – The number of elapsed seconds in the period.

classmethod **from_key** (*period, period_sec*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod **has_key** (*period, period_sec*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

property **min_sec**

Elapsed minutes and seconds of the game.

Type (int, int)

period: int

Game period. 1-3 for regulation. 4+ for overtime.

Type int

property **period_min_sec**

Elapsed minutes and seconds of the period.

Type (int, int)

period_sec: int

Elapsed seconds of the period.

Type int

property **period_str**

Period number as string (i.e. “2nd”)

Type str

property **sec**

Elapsed seconds of the game.

Type int

class nhl.statsapi.**List** (*iterable=(), /*)

Bases: list

Searchable, sortable, and filter-able list subclass

filter (*attr, value, compare='=='*)

Filter list by a comparison of a given attribute (or chain or attributes).

Parameters

- **attr** (*str*) – attributes to be compared (with initial . omitted)
- **value** (*any*) – value to compare attr against
- **compare** (*str*) – comparison type “==”, “=”, “eq”: *attr == value* “!=”, “ne”: *attr != value* “>”, “gt”: *attr > value* “>=”, “ge”: *attr >= value* “<”, “lt”: *attr < value* “<=”, “le”: *attr <= value* “in”, “contains”: *value in attr*

Returns reduced list with items that satisfy filter criterion

Return type *nhl.List*

select (*attr, default=None*)

Select a given attribute (or chain or attributes) from the objects within the list.

Parameters

- **attr** (*str*) – attributes to be selected (with initial . omitted)
- **default** (*any*) – value to return if given element in list doesn’t contain desired attribute

Returns list of selected attribute values

Return type *nhl.List*

sort (*key, reverse=False*)

Sort the list by *key* either ascending or descending.

Parameters

- **key** (*str*) – attribute to sort by (with initial . omitted)
- **reverse** (*bool, optional*) – reverse the direction of sort to descending. (default False, ascending)

Returns sorted list

Return type *nhl.List*

unique ()

Reduce the list to unique elements.

Returns reduced list

Return type *nhl.List*

property len

Helper property to return length of list

Returns length of list *len(self)*

Return type int

class *nhl.statsapi.Location* (**args, **kwargs*)

Bases: *nhl.flyweight.Flyweight*

NHL location object.

This is the detailed docstring.

distance (*other*)

Measure distance between current location and another on-ice location.

Parameters **other** (*Location*) – location to measure distance from

Returns distance (ft)

Return type float

classmethod `from_key(x, y)`

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or `None` if key not found

Return type `cls` or `None`

classmethod `has_key(x, y)`

Check whether flyweight object with specified key has already been created.

Returns `True` if already created, `False` if not

Return type `bool`

x: int

Rink position in ft along x-axis (length) [-100, 100]

Type `int`

y: int

Rink position in ft along y-axis (breadth) [-42, 42]

Type `int`

class `nhl.statsapi.Official(*args, **kwargs)`

Bases: `nhl.flyweight.Flyweight`

NHL official object.

This is the detailed docstring.

classmethod `from_key(id)`

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or `None` if key not found

Return type `nhl.Official` or `None`

classmethod `has_key(id)`

Check whether flyweight object with specified key has already been created.

Returns `True` if already created, `False` if not

Return type `bool`

property `first_name`

Official's first name

Type `str`

id: int

The NHL statsapi universal official ID

Type `int`

property `last_name`

Official's last name

Type `str`

name: str

Official's full name

Type `str`

```

class nhl.statsapi.Player(*args, **kwargs)
    Bases: nhl.flyweight.Flyweight

    NHL player object.

    This is the detailed docstring.

    classmethod from_key(id)
        Return flyweight object with specified key, if it has already been created.

        Returns Previously constructed flyweight object with given key or None if key not found

        Return type cls or None

    classmethod has_key(id)
        Check whether flyweight object with specified key has already been created.

        Returns True if already created, False if not

        Return type bool

    property age
        Current age in years

        Type int

    birth_city: str
        Player's birth city

        Type str

    birth_country: str
        Player's birth country

        Type str

    birth_date: datetime.date
        Player's birth date

        Type datetime.date

    property first_name
        Player's first name

        Type str

    height: int
        Player's height in total inches

        Type int

    property height_ft_in
        Height in feet and inches (height // 12, height % 12)

        Type int

    id: int
        The NHL statsapi universal player ID

        Type int

    property last_name
        Player's last name

        Type str

```

name: str

Player's full name

Type str

number: int

Player's primary number

Type int

position: str

Player's primary position ("LW", "C", "RW", "D", "G")

Type str

shoots_catches: str

Indication of whether the player shoots (skater)/catches (goalie) "L" or "R"

Type str

weight: int

Player's weight in lbs

Type int

class nhl.statsapi.**PlayerStats** (*player: nhl.player.Player, shifts: nhl.list.List, events: nhl.list.List*)

Bases: object

NHL player statistics.

This is the detailed docstring.

events: nhl.list.List

Player's events

Type List[Event]

player: nhl.player.Player

Player object

Type Player

shifts: nhl.list.List

Player's shifts

Type List[Shift]

class nhl.statsapi.**Shift** (**args, **kwargs*)

Bases: nhl.flyweight.Flyweight

NHL shift object.

This is the detailed docstring.

classmethod from_key (*game_id, player_id, shift_id*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key (*game_id, player_id, shift_id*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

game_id: int
NHL statsapi unique game ID

Type int

property length

off: int
Shift end gametime

Type *Gametime*

on: int
Shift start gametime

Type *Gametime*

property player

player_id: int
NHL statsapi unique player ID

Type int

shift_id: int
Shift number for specified game

Type int

class nhl.statsapi.**Team**(*args, **kwargs)

Bases: nhl.flyweight.Flyweight

NHL team object.

This is the detailed docstring.

classmethod **from_key**(id)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod **has_key**(id)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

abbreviation: str

Team's name abbreviated

Type int

conference: nhl.conference.Conference

Conference:

division: nhl.division.Division

Division:

first_year: int

First year of play

Type int

franchise: `nhl.franchise.Franchise`

Franchise:

property full_name

Team's full name

Type `str`

id: `int`

The NHL statsapi universal team ID

Type `int`

location: `str`

Team's location

Type `str`

name: `str`

Team's name

Type `str`

class `nhl.statsapi.Venue(*args, **kwargs)`

Bases: `nhl.flyweight.Flyweight`

NHL venue object.

This is the detailed docstring.

classmethod `from_key(id)`

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type `cls` or `None`

classmethod `has_key(id)`

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type `bool`

id: `int`

The NHL statsapi universal venue ID

Type `int`

name: `str`

Venue name

Type `str`

`nhl.statsapi.conference(id)`

`nhl.statsapi.conferences()`

`nhl.statsapi.division(id)`

`nhl.statsapi.divisions()`

`nhl.statsapi.franchise(id)`

`nhl.statsapi.franchises()`

`nhl.statsapi.game(id)`

`nhl.statsapi.parse_conference` (*json*)
`nhl.statsapi.parse_conferences` (*json*)
`nhl.statsapi.parse_division` (*json*)
`nhl.statsapi.parse_events` (*json*, *info*, *home_score*, *away_score*, *home_shifts*, *away_shifts*,
flip_sides)
`nhl.statsapi.parse_franchise` (*json*)
`nhl.statsapi.parse_game` (*json*)
`nhl.statsapi.parse_location` (*json*, *gametime*, *flip_sides*)
`nhl.statsapi.parse_official` (*json*)
`nhl.statsapi.parse_player` (*json*)
`nhl.statsapi.parse_shifts` (*game_id*, *team_id*, *player_id*, *player_number*, *html*)
`nhl.statsapi.parse_team` (*json*)
`nhl.statsapi.parse_teams` (*json*)
`nhl.statsapi.parse_venue` (*json*)
`nhl.statsapi.player` (*id*)
`nhl.statsapi.players` (*ids*)
`nhl.statsapi.team` (*id*)
`nhl.statsapi.teams` (*ids=None*)
`nhl.statsapi.venue` (*id*)
`nhl.statsapi.venues` ()

EXAMPLES

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

`nhl`, 1

`nhl.rink`, 13

`nhl.statsapi`, 14

A

abbreviation (*nhl.Conference* attribute), 2
 abbreviation (*nhl.Division* attribute), 2
 abbreviation (*nhl.statsapi.Conference* attribute), 17
 abbreviation (*nhl.statsapi.Division* attribute), 18
 abbreviation (*nhl.statsapi.Team* attribute), 27
 abbreviation (*nhl.Team* attribute), 11
 age () (*nhl.Player* property), 9
 age () (*nhl.statsapi.Player* property), 25
 ASCII_SPACES (*nhl.statsapi.BeautifulSoup* attribute), 17
 away (*nhl.Game* attribute), 4
 away (*nhl.statsapi.Game* attribute), 20
 away_score () (*nhl.GameInfo* property), 5
 away_score () (*nhl.statsapi.GameInfo* property), 21
 away_team (*nhl.GameInfo* attribute), 5
 away_team (*nhl.statsapi.GameInfo* attribute), 21

B

BeautifulSoup (*class in nhl.statsapi*), 15
 birth_city (*nhl.Player* attribute), 9
 birth_city (*nhl.statsapi.Player* attribute), 25
 birth_country (*nhl.Player* attribute), 9
 birth_country (*nhl.statsapi.Player* attribute), 25
 birth_date (*nhl.Player* attribute), 9
 birth_date (*nhl.statsapi.Player* attribute), 25
 BLUE_LINE_X (*in module nhl.rink*), 14
 by_player (*nhl.Event* attribute), 3
 by_player (*nhl.statsapi.Event* attribute), 19
 by_players_on_ice (*nhl.Event* attribute), 3
 by_players_on_ice (*nhl.statsapi.Event* attribute), 19
 by_strength () (*nhl.Event* property), 3
 by_strength () (*nhl.statsapi.Event* property), 19
 by_team (*nhl.Event* attribute), 3
 by_team (*nhl.statsapi.Event* attribute), 19

C

Conference (*class in nhl*), 1
 Conference (*class in nhl.statsapi*), 17
 conference (*nhl.statsapi.Team* attribute), 27
 conference (*nhl.Team* attribute), 11

conference () (*in module nhl.statsapi*), 28
 conferences () (*in module nhl.statsapi*), 28

D

date () (*nhl.GameInfo* property), 5
 date () (*nhl.statsapi.GameInfo* property), 21
 decode () (*nhl.statsapi.BeautifulSoup* method), 16
 DEFAULT_BUILDER_FEATURES
 (*nhl.statsapi.BeautifulSoup* attribute), 17
 defensemen () (*nhl.Game* property), 4
 defensemen () (*nhl.statsapi.Game* property), 20
 description () (*nhl.GameInfo* property), 5
 description () (*nhl.statsapi.GameInfo* property), 21
 distance () (*nhl.Location* method), 7
 distance () (*nhl.rink.Location* method), 13
 distance () (*nhl.statsapi.Location* method), 23
 Division (*class in nhl*), 2
 Division (*class in nhl.statsapi*), 18
 division (*nhl.statsapi.Team* attribute), 27
 division (*nhl.Team* attribute), 11
 division () (*in module nhl.statsapi*), 28
 divisions () (*in module nhl.statsapi*), 28
 DX (*in module nhl.rink*), 14
 DY (*in module nhl.rink*), 14

E

end (*nhl.GameInfo* attribute), 5
 end (*nhl.statsapi.GameInfo* attribute), 21
 end_time (*nhl.GameInfo* attribute), 5
 end_time (*nhl.statsapi.GameInfo* attribute), 21
 end_type (*nhl.GameInfo* attribute), 5
 end_type (*nhl.statsapi.GameInfo* attribute), 21
 endData () (*nhl.statsapi.BeautifulSoup* method), 16
 Event (*class in nhl*), 2
 Event (*class in nhl.statsapi*), 18
 events (*nhl.Game* attribute), 4
 events (*nhl.PlayerStats* attribute), 10
 events (*nhl.statsapi.Game* attribute), 20
 events (*nhl.statsapi.PlayerStats* attribute), 26

F

FACEOFF_DOT_Y (*in module nhl.rink*), 14

FACEOFF_DOTS (in module *nhl.rink*), 14
 filter() (*nhl.List* method), 6
 filter() (*nhl.statsapi.List* method), 22
 first_name() (*nhl.Official* property), 8
 first_name() (*nhl.Player* property), 9
 first_name() (*nhl.statsapi.Official* property), 24
 first_name() (*nhl.statsapi.Player* property), 25
 first_year (*nhl.statsapi.Team* attribute), 27
 first_year (*nhl.Team* attribute), 11
 forwards() (*nhl.Game* property), 4
 forwards() (*nhl.statsapi.Game* property), 20
 Franchise (class in *nhl*), 3
 Franchise (class in *nhl.statsapi*), 19
 franchise (*nhl.statsapi.Team* attribute), 27
 franchise (*nhl.Team* attribute), 11
 franchise() (in module *nhl.statsapi*), 28
 franchises() (in module *nhl.statsapi*), 28
 from_key() (*nhl.Conference* class method), 1
 from_key() (*nhl.Division* class method), 2
 from_key() (*nhl.Event* class method), 3
 from_key() (*nhl.Franchise* class method), 3
 from_key() (*nhl.Game* class method), 4
 from_key() (*nhl.GameInfo* class method), 5
 from_key() (*nhl.Gametime* class method), 6
 from_key() (*nhl.Location* class method), 8
 from_key() (*nhl.Official* class method), 8
 from_key() (*nhl.Player* class method), 9
 from_key() (*nhl.rink.Location* class method), 13
 from_key() (*nhl.Shift* class method), 10
 from_key() (*nhl.statsapi.Conference* class method), 17
 from_key() (*nhl.statsapi.Division* class method), 18
 from_key() (*nhl.statsapi.Event* class method), 18
 from_key() (*nhl.statsapi.Franchise* class method), 19
 from_key() (*nhl.statsapi.Game* class method), 20
 from_key() (*nhl.statsapi.GameInfo* class method), 21
 from_key() (*nhl.statsapi.Gametime* class method), 22
 from_key() (*nhl.statsapi.Location* class method), 23
 from_key() (*nhl.statsapi.Official* class method), 24
 from_key() (*nhl.statsapi.Player* class method), 25
 from_key() (*nhl.statsapi.Shift* class method), 26
 from_key() (*nhl.statsapi.Team* class method), 27
 from_key() (*nhl.statsapi.Venue* class method), 28
 from_key() (*nhl.Team* class method), 11
 from_key() (*nhl.Venue* class method), 12
 full_name() (*nhl.statsapi.Team* property), 28
 full_name() (*nhl.Team* property), 12

G

Game (class in *nhl*), 4
 Game (class in *nhl.statsapi*), 20
 game() (in module *nhl.statsapi*), 28
 game_id (*nhl.Event* attribute), 3
 game_id (*nhl.Shift* attribute), 10

game_id (*nhl.statsapi.Event* attribute), 19
 game_id (*nhl.statsapi.Shift* attribute), 27
 GameInfo (class in *nhl*), 5
 GameInfo (class in *nhl.statsapi*), 20
 Gametime (class in *nhl*), 6
 Gametime (class in *nhl.statsapi*), 21
 gametime (*nhl.Event* attribute), 3
 gametime (*nhl.statsapi.Event* attribute), 19
 GOAL_LINE_X (in module *nhl.rink*), 14
 goalies() (*nhl.Game* property), 4
 goalies() (*nhl.statsapi.Game* property), 20

H

handle_data() (*nhl.statsapi.BeautifulSoup* method), 16
 handle_endtag() (*nhl.statsapi.BeautifulSoup* method), 16
 handle_starttag() (*nhl.statsapi.BeautifulSoup* method), 16
 has_key() (*nhl.Conference* class method), 1
 has_key() (*nhl.Division* class method), 2
 has_key() (*nhl.Event* class method), 3
 has_key() (*nhl.Franchise* class method), 4
 has_key() (*nhl.Game* class method), 4
 has_key() (*nhl.GameInfo* class method), 5
 has_key() (*nhl.Gametime* class method), 6
 has_key() (*nhl.Location* class method), 8
 has_key() (*nhl.Official* class method), 8
 has_key() (*nhl.Player* class method), 9
 has_key() (*nhl.rink.Location* class method), 13
 has_key() (*nhl.Shift* class method), 10
 has_key() (*nhl.statsapi.Conference* class method), 17
 has_key() (*nhl.statsapi.Division* class method), 18
 has_key() (*nhl.statsapi.Event* class method), 19
 has_key() (*nhl.statsapi.Franchise* class method), 19
 has_key() (*nhl.statsapi.Game* class method), 20
 has_key() (*nhl.statsapi.GameInfo* class method), 21
 has_key() (*nhl.statsapi.Gametime* class method), 22
 has_key() (*nhl.statsapi.Location* class method), 24
 has_key() (*nhl.statsapi.Official* class method), 24
 has_key() (*nhl.statsapi.Player* class method), 25
 has_key() (*nhl.statsapi.Shift* class method), 26
 has_key() (*nhl.statsapi.Team* class method), 27
 has_key() (*nhl.statsapi.Venue* class method), 28
 has_key() (*nhl.Team* class method), 11
 has_key() (*nhl.Venue* class method), 12
 height (*nhl.Player* attribute), 9
 height (*nhl.statsapi.Player* attribute), 25
 height_ft_in() (*nhl.Player* property), 9
 height_ft_in() (*nhl.statsapi.Player* property), 25
 home (*nhl.Game* attribute), 4
 home (*nhl.statsapi.Game* attribute), 20
 home_score() (*nhl.GameInfo* property), 5
 home_score() (*nhl.statsapi.GameInfo* property), 21

home_team (*nhl.GameInfo* attribute), 5
 home_team (*nhl.statsapi.GameInfo* attribute), 21

I

id (*nhl.Conference* attribute), 2
 id (*nhl.Division* attribute), 2
 id (*nhl.Event* attribute), 3
 id (*nhl.Franchise* attribute), 4
 id (*nhl.GameInfo* attribute), 5
 id (*nhl.Official* attribute), 8
 id (*nhl.Player* attribute), 9
 id (*nhl.statsapi.Conference* attribute), 17
 id (*nhl.statsapi.Division* attribute), 18
 id (*nhl.statsapi.Event* attribute), 19
 id (*nhl.statsapi.Franchise* attribute), 20
 id (*nhl.statsapi.GameInfo* attribute), 21
 id (*nhl.statsapi.Official* attribute), 24
 id (*nhl.statsapi.Player* attribute), 25
 id (*nhl.statsapi.Team* attribute), 28
 id (*nhl.statsapi.Venue* attribute), 28
 id (*nhl.Team* attribute), 12
 id (*nhl.Venue* attribute), 12
 info (*nhl.Game* attribute), 4
 info (*nhl.statsapi.Game* attribute), 20
 insert_after() (*nhl.statsapi.BeautifulSoup*
 method), 16
 insert_before() (*nhl.statsapi.BeautifulSoup*
 method), 16

L

last_name() (*nhl.Official* property), 8
 last_name() (*nhl.Player* property), 9
 last_name() (*nhl.statsapi.Official* property), 24
 last_name() (*nhl.statsapi.Player* property), 25
 lead() (*nhl.Event* property), 3
 lead() (*nhl.statsapi.Event* property), 19
 len() (*nhl.List* property), 7
 len() (*nhl.statsapi.List* property), 23
 LENGTH (in module *nhl.rink*), 14
 length() (*nhl.Shift* property), 11
 length() (*nhl.statsapi.Shift* property), 27
 linesmen (*nhl.GameInfo* attribute), 5
 linesmen (*nhl.statsapi.GameInfo* attribute), 21
 List (class in *nhl*), 6
 List (class in *nhl.statsapi*), 22
 Location (class in *nhl*), 7
 Location (class in *nhl.rink*), 13
 Location (class in *nhl.statsapi*), 23
 location (*nhl.Event* attribute), 3
 location (*nhl.statsapi.Event* attribute), 19
 location (*nhl.statsapi.Team* attribute), 28
 location (*nhl.Team* attribute), 12

M

min_sec() (*nhl.Gametime* property), 6
 min_sec() (*nhl.statsapi.Gametime* property), 22
 module
 nhl, 1
 nhl.rink, 13
 nhl.statsapi, 14

N

name (*nhl.Conference* attribute), 2
 name (*nhl.Division* attribute), 2
 name (*nhl.Franchise* attribute), 4
 name (*nhl.Official* attribute), 8
 name (*nhl.Player* attribute), 9
 name (*nhl.statsapi.Conference* attribute), 18
 name (*nhl.statsapi.Division* attribute), 18
 name (*nhl.statsapi.Franchise* attribute), 20
 name (*nhl.statsapi.Official* attribute), 24
 name (*nhl.statsapi.Player* attribute), 25
 name (*nhl.statsapi.Team* attribute), 28
 name (*nhl.statsapi.Venue* attribute), 28
 name (*nhl.Team* attribute), 12
 name (*nhl.Venue* attribute), 12
 name() (*nhl.Event* property), 3
 name() (*nhl.statsapi.Event* property), 19
 name_short (*nhl.Conference* attribute), 2
 name_short (*nhl.Division* attribute), 2
 name_short (*nhl.statsapi.Conference* attribute), 18
 name_short (*nhl.statsapi.Division* attribute), 18
 new_string() (*nhl.statsapi.BeautifulSoup* method),
 16
 new_tag() (*nhl.statsapi.BeautifulSoup* method), 16
 nhl
 module, 1
 nhl.rink
 module, 13
 nhl.statsapi
 module, 14
 NO_PARSER_SPECIFIED_WARNING
 (*nhl.statsapi.BeautifulSoup* attribute), 17
 number (*nhl.Player* attribute), 10
 number (*nhl.statsapi.Player* attribute), 26
 NZ_FACEOFF_DOT_X (in module *nhl.rink*), 14

O

object_was_parsed() (*nhl.statsapi.BeautifulSoup*
 method), 17
 off (*nhl.Shift* attribute), 11
 off (*nhl.statsapi.Shift* attribute), 27
 Official (class in *nhl*), 8
 Official (class in *nhl.statsapi*), 24
 on (*nhl.Shift* attribute), 11
 on (*nhl.statsapi.Shift* attribute), 27
 on_player (*nhl.Event* attribute), 3

on_player (*nhl.statsapi.Event attribute*), 19
 on_players_on_ice (*nhl.Event attribute*), 3
 on_players_on_ice (*nhl.statsapi.Event attribute*), 19
 on_strength () (*nhl.Event property*), 3
 on_strength () (*nhl.statsapi.Event property*), 19
 on_team (*nhl.Event attribute*), 3
 on_team (*nhl.statsapi.Event attribute*), 19
 OZ_FACEOFF_DOT_X (*in module nhl.rink*), 14

P

parse_conference () (*in module nhl.statsapi*), 28
 parse_conferences () (*in module nhl.statsapi*), 29
 parse_division () (*in module nhl.statsapi*), 29
 parse_events () (*in module nhl.statsapi*), 29
 parse_franchise () (*in module nhl.statsapi*), 29
 parse_game () (*in module nhl.statsapi*), 29
 parse_location () (*in module nhl.statsapi*), 29
 parse_official () (*in module nhl.statsapi*), 29
 parse_player () (*in module nhl.statsapi*), 29
 parse_shifts () (*in module nhl.statsapi*), 29
 parse_team () (*in module nhl.statsapi*), 29
 parse_teams () (*in module nhl.statsapi*), 29
 parse_venue () (*in module nhl.statsapi*), 29
 period (*nhl.Gametime attribute*), 6
 period (*nhl.statsapi.Gametime attribute*), 22
 period_min_sec () (*nhl.Gametime property*), 6
 period_min_sec () (*nhl.statsapi.Gametime property*), 22
 period_sec (*nhl.Gametime attribute*), 6
 period_sec (*nhl.statsapi.Gametime attribute*), 22
 period_str () (*nhl.Gametime property*), 6
 period_str () (*nhl.statsapi.Gametime property*), 22
 Player (*class in nhl*), 8
 Player (*class in nhl.statsapi*), 24
 player (*nhl.PlayerStats attribute*), 10
 player (*nhl.statsapi.PlayerStats attribute*), 26
 player () (*in module nhl.statsapi*), 29
 player () (*nhl.Shift property*), 11
 player () (*nhl.statsapi.Shift property*), 27
 player_id (*nhl.Shift attribute*), 11
 player_id (*nhl.statsapi.Shift attribute*), 27
 players (*nhl.Game attribute*), 4
 players (*nhl.statsapi.Game attribute*), 20
 players () (*in module nhl.statsapi*), 29
 PlayerStats (*class in nhl*), 10
 PlayerStats (*class in nhl.statsapi*), 26
 playoff_game () (*nhl.GameInfo property*), 5
 playoff_game () (*nhl.statsapi.GameInfo property*), 21
 playoff_round () (*nhl.GameInfo property*), 5
 playoff_round () (*nhl.statsapi.GameInfo property*), 21
 playoff_series () (*nhl.GameInfo property*), 5

playoff_series () (*nhl.statsapi.GameInfo property*), 21
 popTag () (*nhl.statsapi.BeautifulSoup method*), 17
 position (*nhl.Player attribute*), 10
 position (*nhl.statsapi.Player attribute*), 26
 pushTag () (*nhl.statsapi.BeautifulSoup method*), 17

R

referees (*nhl.GameInfo attribute*), 5
 referees (*nhl.statsapi.GameInfo attribute*), 21
 reset () (*nhl.statsapi.BeautifulSoup method*), 17
 ROOT_TAG_NAME (*nhl.statsapi.BeautifulSoup attribute*), 17

S

score (*nhl.Event attribute*), 3
 score (*nhl.GameInfo attribute*), 5
 score (*nhl.statsapi.Event attribute*), 19
 score (*nhl.statsapi.GameInfo attribute*), 21
 season () (*nhl.GameInfo property*), 5
 season () (*nhl.statsapi.GameInfo property*), 21
 season_id (*nhl.GameInfo attribute*), 5
 season_id (*nhl.statsapi.GameInfo attribute*), 21
 season_type () (*nhl.GameInfo property*), 5
 season_type () (*nhl.statsapi.GameInfo property*), 21
 sec () (*nhl.Gametime property*), 6
 sec () (*nhl.statsapi.Gametime property*), 22
 select () (*nhl.List method*), 7
 select () (*nhl.statsapi.List method*), 23
 Shift (*class in nhl*), 10
 Shift (*class in nhl.statsapi*), 26
 shift_id (*nhl.Shift attribute*), 11
 shift_id (*nhl.statsapi.Shift attribute*), 27
 shifts (*nhl.PlayerStats attribute*), 10
 shifts (*nhl.statsapi.PlayerStats attribute*), 26
 shoots_catches (*nhl.Player attribute*), 10
 shoots_catches (*nhl.statsapi.Player attribute*), 26
 skaters () (*nhl.Game property*), 5
 skaters () (*nhl.statsapi.Game property*), 20
 sort () (*nhl.List method*), 7
 sort () (*nhl.statsapi.List method*), 23
 start (*nhl.GameInfo attribute*), 5
 start (*nhl.statsapi.GameInfo attribute*), 21
 strength () (*nhl.Event property*), 3
 strength () (*nhl.statsapi.Event property*), 19
 string_container () (*nhl.statsapi.BeautifulSoup method*), 17
 subname () (*nhl.Event property*), 3
 subname () (*nhl.statsapi.Event property*), 19
 subtype (*nhl.Event attribute*), 3
 subtype (*nhl.statsapi.Event attribute*), 19

T

Team (*class in nhl*), 11

Team (*class in nhl.statsapi*), 27
team () (*in module nhl.statsapi*), 29
teams () (*in module nhl.statsapi*), 29
type (*nhl.Event attribute*), 3
type (*nhl.GameInfo attribute*), 5
type (*nhl.statsapi.Event attribute*), 19
type (*nhl.statsapi.GameInfo attribute*), 21

U

unique () (*nhl.List method*), 7
unique () (*nhl.statsapi.List method*), 23

V

value (*nhl.Event attribute*), 3
value (*nhl.statsapi.Event attribute*), 19
valuenam () (*nhl.Event property*), 3
valuenam () (*nhl.statsapi.Event property*), 19
Venue (*class in nhl*), 12
Venue (*class in nhl.statsapi*), 28
venue (*nhl.GameInfo attribute*), 6
venue (*nhl.statsapi.GameInfo attribute*), 21
venue () (*in module nhl.statsapi*), 29
venues () (*in module nhl.statsapi*), 29

W

weight (*nhl.Player attribute*), 10
weight (*nhl.statsapi.Player attribute*), 26
WIDTH (*in module nhl.rink*), 14

X

x (*nhl.Location attribute*), 8
x (*nhl.rink.Location attribute*), 13
x (*nhl.statsapi.Location attribute*), 24

Y

y (*nhl.Location attribute*), 8
y (*nhl.rink.Location attribute*), 13
y (*nhl.statsapi.Location attribute*), 24