

---

# API Elements (JS) Documentation

*Release 1.0*

**Apiary**

**Jan 24, 2019**



---

## Contents

---

<b>1</b>	<b>Installation</b>	<b>1</b>
<b>2</b>	<b>Usage</b>	<b>3</b>
<b>3</b>	<b>API Reference</b>	<b>5</b>
3.1	API Reference . . . . .	5
3.1.1	Namespace . . . . .	6
3.1.2	Elements . . . . .	7
3.1.3	Slice . . . . .	19
<b>4</b>	<b>Indices and tables</b>	<b>23</b>



# CHAPTER 1

---

## Installation

---

```
$ npm install api-elements
```



## CHAPTER 2

---

### Usage

---

```
const apiElements = require('api-elements');
const namespace = new apiElements.Namespace();

// Parsing a JSON Representation of API Elements tree
const parseResult = namespace.serialiser.deserialise({
  element: 'parseResult',
  content: []
});

console.log(parseResult);

// Creating API Elements directly
const parseResult = new namespace.elements.ParseResult();
console.log(parseResult);
```





### 3.1 API Reference

#### Contents

- *API Reference*
  - *Namespace*
    - \* *Serialiser*
  - *Elements*
    - \* *Element*
    - \* *Primitives*
      - *String*
      - *Number*
      - *Boolean*
      - *Null*
    - \* *Collections*
      - *Array*
      - *Object*
      - *Member*
    - \* *Profiles*
    - \* *Referencing*
    - \* *Parse Result*

```
· Annotation
· SourceMap
* API Description
· Category
· Copy
· Data Structure
· Resource
· HTTP Transaction

– Slice
```

### 3.1.1 Namespace

**class Namespace ()**

A refract element implementation with an extensible namespace, able to load other namespaces into it.

The namespace allows you to register your own classes to be instantiated when a particular refract element is encountered, and allows you to specify which elements get instantiated for existing Javascript objects.

`Namespace.register (name, elementClass)`

Register a new element class for an element.

#### Arguments

- **name** (*string*) –
- **elementClass** –

`Namespace.serializer`

**type:** JSONSerialiser

Convenience method for getting a JSON Serialiser configured with the current namespace

`Namespace.unregister (name)`

Unregister a previously registered class for an element.

#### Arguments

- **name** (*string*) –

`Namespace.use (plugin)`

Use a namespace plugin or load a generic plugin.

#### Arguments

- **plugin** –

### Serialiser

**class JSONSerialiser (namespace)**

#### Arguments

- **namespace** (*Namespace*) –
- **namespace** –

`JSONSerialiser.deserialize(value)`

**Arguments**

- **value** (*object*) –

**Returns** *Element* –

`JSONSerialiser.serialize(element)`

**Arguments**

- **element** (*Element*) –

**Returns** *object* –

## 3.1.2 Elements

### Element

**class** `Element` (*content, meta, attributes*)

**Arguments**

- **content** –
- **meta** –
- **attributes** –
- **element** (*string*) –

`Element.attributes`

The attributes property defines attributes about the given instance of the element, as specified by the element property.

`Element.children`

**type:** *ArraySlice*

Returns all of the children elements found within the element.

**See also:**

- *recursiveChildren*

`Element.classes`

**type:** *ArrayElement*

`Element.clone()`

Creates a deep clone of the instance

`Element.description`

**type:** *StringElement*

Human-readable description of element

`Element.element`

**type:** *String*

`Element.findRecursive(...names)`

Finds the given elements in the element tree. When providing multiple element names, you must first freeze the element.

**Arguments**

- **names** (*elementNames*) –

**Returns** `ArraySlice` –

`Element.freeze()`

Freezes the element to prevent any mutation. A frozen element will add *parent* property to every child element to allow traversing up the element tree.

`Element.id`

**type:** `StringElement`

Unique Identifier, MUST be unique throughout an entire element tree.

`Element.isFrozen`

**type:** `boolean`

Returns whether the element is frozen.

**See also:**

- *freeze*

`Element.links`

**type:** `ArrayElement`

`Element.meta`

`Element.parents`

**type:** `ArraySlice`

Returns all of the parent elements.

**See also:**

- *freeze*

`Element.recursiveChildren`

**type:** `ArraySlice`

Returns all of the children elements found within the element recursively.

**See also:**

- *children*

`Element.title`

**type:** `StringElement`

Human-readable title of element

`Element.toRef()`

Creates a reference pointing at the Element

**Returns** `RefElement` –

`Element.toValue()`

## Primitives

### String

**class** **StringElement** (*content, meta, attributes*)

#### Arguments

- **content** (*string*) –
- **meta** –
- **attributes** –

**StringElement.length**

**type:** number

The length of the string.

### Number

**class** **NumberElement** (*content, meta, attributes*)

#### Arguments

- **content** (*number*) –
- **meta** –
- **attributes** –

### Boolean

**class** **BooleanElement** (*content, meta, attributes*)

#### Arguments

- **content** (*boolean*) –
- **meta** –
- **attributes** –

### Null

**class** **NullElement** ()

## Collections

### Array

**class** **ArrayElement** (*content, meta, attributes*)

#### Arguments

- **content** (*Array.<Element>*) –
- **meta** –

- **attributes** –

`ArrayElement.add(value)`

**Arguments**

- **value** –

`ArrayElement.compactMap(transform, thisArg)`

Returns an array containing the truthy results of calling the given transformation with each element of this sequence

**Arguments**

- **transform** – A closure that accepts an element of this array as its argument and returns an optional value.
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** An array of the non-undefined results of calling transform with each element of the array

`ArrayElement.contains(value)`

Looks for matching children using deep equality

**Arguments**

- **value** –

**Returns** boolean –

`ArrayElement.filter(callback, thisArg)`

**Arguments**

- **callback** – Function to execute for each element
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** `ArraySlice` –

`ArrayElement.find(condition)`

Recursively search all descendents using a condition function.

**Arguments**

- **condition** –

**Returns** `ArraySlice` –

`ArrayElement.findByClass(className)`

**Arguments**

- **className** (*string*) –

**Returns** `ArraySlice` –

`ArrayElement.findByElement(element)`

**Arguments**

- **element** (*string*) –

**Returns** `ArraySlice` –

`ArrayElement.findElements()`

Recursively search all descendents using a condition function.

**Returns** `Array.<Element>` –

`ArrayElement.first`

**type:** `Element`

Return the first item in the collection

`ArrayElement.flatMap (callback, thisArg)`

Maps and then flattens the results.

**Arguments**

- **callback** – Function to execute for each element.
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** `array` –

`ArrayElement.forEach (callback, thisArg)`

**Arguments**

- **callback** (*forEachCallback*) – Function to execute for each element
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

`ArrayElement.get ()`

**Returns** `Element` –

`ArrayElement.getById (id)`

Search the tree recursively and find the element with the matching ID

**Arguments**

- **id** (*string*) –

**Returns** `Element` –

`ArrayElement.getIndex ()`

**Returns** `Element` –

`ArrayElement.getValue ()`

Helper for returning the value of an item This works for both `ArrayElement` and `ObjectElement` instances

`ArrayElement.isEmpty`

**type:** `boolean`

Returns whether the collection is empty

`ArrayElement.last`

**type:** `Element`

Return the last item in the collection

`ArrayElement.length`

**type:** `number`

Returns the length of the collection

`ArrayElement.map (callback, thisArg)`

**Arguments**

- **callback** – Function to execute for each element
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

`ArrayElement.push(value)`

### Arguments

- **value** –

`ArrayElement.reject(callback, thisArg)`

### Arguments

- **callback** – Function to execute for each element
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** `ArraySlice` –

`ArrayElement.remove()`

`ArrayElement.second`  
**type:** `Element`

Return the second item in the collection

`ArrayElement.set()`

`ArrayElement.shift()`

**Returns** `Element` –

`ArrayElement.unshift(value)`

### Arguments

- **value** –

## Object

**class** `ObjectElement` (*content, meta, attributes*)

### Arguments

- **content** –
- **meta** –
- **attributes** –

`ObjectElement.compactMap(transform, thisArg)`

Returns an array containing the truthy results of calling the given transformation with each element of this sequence

### Arguments

- **transform** – A closure that accepts the value, key and member element of this object as its argument and returns an optional value.
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** An array of the non-undefined results of calling transform with each element of the array

`ObjectElement.filter(callback, thisArg)`

### Arguments

- **callback** –



- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** `ObjectSlice` –

`ObjectElement.forEach(callback, thisArg)`

**Arguments**

- **callback** –
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

`ObjectElement.get(key)`

**Arguments**

- **key** –

**Returns** `Element` –

`ObjectElement.getKey(key)`

**Arguments**

- **key** –

**Returns** `Element` –

`ObjectElement.getMember(key)`

**Arguments**

- **key** –

**Returns** `MemberElement` –

`ObjectElement.hasKey()`

**Returns** `boolean` –

`ObjectElement.items()`

**Returns** `array` –

`ObjectElement.keys()`

`ObjectElement.map(callback, thisArg)`

**Arguments**

- **callback** –
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

`ObjectElement.reject(callback, thisArg)`

**Arguments**

- **callback** –
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** `ObjectSlice` –

`ObjectElement.remove(key)`

**Arguments**

- **key** –

`ObjectElement.set()`

Set allows either a key/value pair to be given or an object. If an object is given, each key is set to its respective value.

`ObjectElement.values()`

### Member

**class** `MemberElement` (*key, value, meta, attributes*)

#### Arguments

- **key** (`Element`) –
- **value** (`Element`) –
- **meta** –
- **attributes** –

`MemberElement.key`  
**type:** `Element`

`MemberElement.value`  
**type:** `Element`

### Profiles

**class** `LinkElement` (*content, meta, attributes*)

Hyperlinking MAY be used to link to other resources, provide links to instructions on how to process a given element (by way of a profile or other means), and may be used to provide meta data about the element in which it's found. The meaning and purpose of the hyperlink is defined by the link relation according to RFC 5988.

#### Arguments

- **content** –
- **meta** –
- **attributes** –

`LinkElement.href`  
**type:** `StringElement`

The URI for the given link.

`LinkElement.relation`  
**type:** `StringElement`

The relation identifier for the link, as defined in RFC 5988.

### Referencing

**class** `RefElement` (*content, meta, attributes*)

#### Arguments

- **content** –
- **meta** –
- **attributes** –

`RefElement.path`  
**type:** `StringElement`

Path of referenced element to transclude instead of element itself.

## Parse Result

**class** `ParseResult` (*content, meta, attributes*)

### Arguments

- **content** (*Array*) –
- **meta** –
- **attributes** –

`ParseResult.annotations`  
**type:** `ArraySlice`

`ParseResult.api`  
**type:** `Category`

`ParseResult.errors`  
**type:** `ArraySlice`

`ParseResult.sourceMapValue`  
**type:** `Array`

`ParseResult.warnings`  
**type:** `ArraySlice`

## Annotation

**class** `Annotation` (*content, meta, attributes*)

### Arguments

- **content** (*string*) –
- **meta** –
- **attributes** –

`Annotation.sourceMapValue`  
**type:** `Array`

## SourceMap

**class** `SourceMap` (*content, meta, attributes*)

### Arguments

- **content** (*Array*) –
- **meta** –
- **attributes** –

`SourceMap.sourceMapValue`  
**type:** `Array`

## API Description

### Category

**class** **Category** (*content, meta, attributes*)

#### Arguments

- **content** (*Array*) –
- **meta** –
- **attributes** –

**Category.authSchemeGroups**

**type:** *ArraySlice*

**Category.authSchemes**

**type:** *ArraySlice*

**Category.dataStructures**

**type:** *ArraySlice*

**Category.resourceGroups**

**type:** *ArraySlice*

**Category.resources**

**type:** *ArraySlice*

**Category.transitions**

**type:** *ArraySlice*

### Copy

**class** **Copy** (*content, meta, attributes*)

#### Arguments

- **content** (*string*) –
- **meta** –
- **attributes** –

**Copy.contentType**

**type:** *StringElement*

**Copy.copy**

**type:** *Copy*

### Data Structure

**class** **DataStructure** (*content, meta, attributes*)

#### Arguments

- **content** (*Element*) –
- **meta** –
- **attributes** –

**class** **Enum** (*content, meta, attributes*)

**Arguments**

- **content** ([Element](#)) –
- **meta** –
- **attributes** –

**Enum.enumerations**  
**type:** [ArrayElement](#)

## Resource

**class** **Resource** (*content, meta, attributes*)

**Arguments**

- **content** ([Array](#)) –
- **meta** –
- **attributes** –

**Resource.dataStructure**  
**type:** [DataStructure](#)

**Resource.href**  
**type:** [StringElement](#)

**Resource.hrefVariables**  
**type:** [HrefVariables](#)

**Resource.transitions**  
**type:** [ArraySlice](#)

**class** **Transition** (*content, meta, attributes*)

**Arguments**

- **content** ([Array](#)) –
- **meta** –
- **attributes** –

**Transition.contentTypes**  
**type:** [ArrayElement](#)

**Transition.data**  
**type:** [DataStructure](#)

**Transition.href**  
**type:** [StringElement](#)

**Transition.hrefVariables**  
**type:** [HrefVariables](#)

**Transition.method**  
**type:** [StringElement](#)

**Transition.relation**  
**type:** [StringElement](#)

`Transition.transactions`  
**type:** `ArraySlice`

## HTTP Transaction

**class** `HttpTransaction` (*content, meta, attributes*)

### Arguments

- **content** (*Array*) –
- **meta** –
- **attributes** –

`HttpTransaction.authSchemes`  
**type:** `ArrayElement`

`HttpTransaction.request`  
**type:** `HttpRequest`

`HttpTransaction.response`  
**type:** `HttpResponse`

**class** `HttpMessagePayload` (*content, meta, attributes*)

### Arguments

- **content** (*Array*) –
- **meta** –
- **attributes** –

`HttpMessagePayload.contentType`  
**type:** `StringElement`

`HttpMessagePayload.dataStructure`  
**type:** `Asset`

`HttpMessagePayload.headers`  
**type:** `HttpHeaders`

`HttpMessagePayload.messageBody`  
**type:** `Asset`

`HttpMessagePayload.messageBodySchema`  
**type:** `Asset`

**class** `HttpRequest` (*content, meta, attributes*)

### Arguments

- **content** –
- **meta** –
- **attributes** –

`HttpRequest.href`  
**type:** `StringElement`

`HttpRequest.method`  
**type:** `StringElement`

**class** **HttpResponse** (*content, meta, attributes*)

**Arguments**

- **content** –
- **meta** –
- **attributes** –

`HttpResponse.statusCode`

**type:** NumberElement

**class** **HttpHeaders** (*content, meta, attributes*)

**Arguments**

- **content** (*Array*) –
- **meta** –
- **attributes** –

**class** **Asset** (*content, meta, attributes*)

**Arguments**

- **content** (*string*) –
- **meta** –
- **attributes** –

`Asset.contentType`

**type:** StringElement

`Asset.href`

**type:** StringElement

**class** **HrefVariables** (*content, meta, attributes*)

**Arguments**

- **content** (*Array*) –
- **meta** –
- **attributes** –

**class** **AuthScheme** (*content, meta, attributes*)

**Arguments**

- **content** (*Array*) –
- **meta** –
- **attributes** –

### 3.1.3 Slice

**class** **ArraySlice** (*elements*)

**Arguments**

- **elements** (*Array.<Element>*) –
- **elements** –

`ArraySlice.add()`

`ArraySlice.compactMap` (*transform*, *thisArg*)

Returns an array containing the truthy results of calling the given transformation with each element of this sequence

**Arguments**

- **transform** – A closure that accepts an element of this array as its argument and returns an optional value.
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** An array of the non-undefined results of calling transform with each element of the array

`ArraySlice.filter` (*callback*, *thisArg*)

**Arguments**

- **callback** – Function to execute for each element. This may be a callback, an element name or an element class.
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** `ArraySlice` –

`ArraySlice.find` (*callback*, *thisArg*)

Returns the first element in the array that satisfies the given value

**Arguments**

- **callback** – Function to execute for each element. This may be a callback, an element name or an element class.
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** `Element` –

`ArraySlice.first`

**type:** `Element`

Returns the first element in the slice or undefined if the slice is empty

`ArraySlice.flatMap` (*callback*, *thisArg*)

Maps and then flattens the results.

**Arguments**

- **callback** – Function to execute for each element.
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** `array` –

`ArraySlice.forEach` (*callback*, *thisArg*)

**Arguments**

- **callback** – Function to execute for each element
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

`ArraySlice.get()`

**Returns** `Element` –

`ArraySlice.getValue()`



`ArraySlice.includes (value)`

**Arguments**

- **value** –

**Returns** `boolean` –

`ArraySlice.isEmpty`

**type:** `boolean`

Returns whether the slice is empty

`ArraySlice.length`

**type:** `number`

Returns the number of elements in the slice

`ArraySlice.map (callback, thisArg)`

**Arguments**

- **callback** – Function to execute for each element
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** `array` – A new array with each element being the result of the callback function

`ArraySlice.push ()`

Adds the given element to the end of the slice

`ArraySlice.reduce (callback, initialValue)`

**Arguments**

- **callback** – Function to execute for each element
- **initialValue** –

`ArraySlice.reject (callback, thisArg)`

**Arguments**

- **callback** – Function to execute for each element. This may be a callback, an element name or an element class.
- **thisArg** – Value to use as this (i.e the reference Object) when executing callback

**Returns** `ArraySlice` –

`ArraySlice.shift ()`

Removes the first element from the slice

**Returns** `Element` – The removed element or undefined if the slice is empty

`ArraySlice.toValue ()`

**Returns** `Array` –

`ArraySlice.unshift ()`

Adds the given element to the beginning of the slice

**class** `ObjectSlice ()`

`ObjectSlice.keys ()`

**Returns** `array` –

`ObjectSlice.values()`

**Returns** array –

## CHAPTER 4

---

### Indices and tables

---

- `genindex`
- `search`



## A

[Annotation\(\) \(class\), 15](#)  
[Annotation.sourceMapValue \(Annotation attribute\), 15](#)  
[ArrayElement\(\) \(class\), 9](#)  
[ArrayElement.add\(\) \(ArrayElement method\), 10](#)  
[ArrayElement.compactMap\(\) \(ArrayElement method\), 10](#)  
[ArrayElement.contains\(\) \(ArrayElement method\), 10](#)  
[ArrayElement.filter\(\) \(ArrayElement method\), 10](#)  
[ArrayElement.find\(\) \(ArrayElement method\), 10](#)  
[ArrayElement.findByClass\(\) \(ArrayElement method\), 10](#)  
[ArrayElement.findByElement\(\) \(ArrayElement method\), 10](#)  
[ArrayElement.findElements\(\) \(ArrayElement method\), 10](#)  
[ArrayElement.first \(ArrayElement attribute\), 11](#)  
[ArrayElement.flatMap\(\) \(ArrayElement method\), 11](#)  
[ArrayElement.forEach\(\) \(ArrayElement method\), 11](#)  
[ArrayElement.get\(\) \(ArrayElement method\), 11](#)  
[ArrayElement.getId\(\) \(ArrayElement method\), 11](#)  
[ArrayElement.getIndex\(\) \(ArrayElement method\), 11](#)  
[ArrayElement.getValue\(\) \(ArrayElement method\), 11](#)  
[ArrayElement.isEmpty \(ArrayElement attribute\), 11](#)  
[ArrayElement.last \(ArrayElement attribute\), 11](#)  
[ArrayElement.length \(ArrayElement attribute\), 11](#)  
[ArrayElement.map\(\) \(ArrayElement method\), 11](#)  
[ArrayElement.push\(\) \(ArrayElement method\), 11](#)  
[ArrayElement.reject\(\) \(ArrayElement method\), 12](#)

[ArrayElement.remove\(\) \(ArrayElement method\), 12](#)  
[ArrayElement.second \(ArrayElement attribute\), 12](#)  
[ArrayElement.set\(\) \(ArrayElement method\), 12](#)  
[ArrayElement.shift\(\) \(ArrayElement method\), 12](#)  
[ArrayElement.unshift\(\) \(ArrayElement method\), 12](#)  
[ArraySlice\(\) \(class\), 19](#)  
[ArraySlice.add\(\) \(ArraySlice method\), 19](#)  
[ArraySlice.compactMap\(\) \(ArraySlice method\), 20](#)  
[ArraySlice.filter\(\) \(ArraySlice method\), 20](#)  
[ArraySlice.find\(\) \(ArraySlice method\), 20](#)  
[ArraySlice.first \(ArraySlice attribute\), 20](#)  
[ArraySlice.flatMap\(\) \(ArraySlice method\), 20](#)  
[ArraySlice.forEach\(\) \(ArraySlice method\), 20](#)  
[ArraySlice.get\(\) \(ArraySlice method\), 20](#)  
[ArraySlice.getValue\(\) \(ArraySlice method\), 20](#)  
[ArraySlice.includes\(\) \(ArraySlice method\), 20](#)  
[ArraySlice.isEmpty \(ArraySlice attribute\), 21](#)  
[ArraySlice.length \(ArraySlice attribute\), 21](#)  
[ArraySlice.map\(\) \(ArraySlice method\), 21](#)  
[ArraySlice.push\(\) \(ArraySlice method\), 21](#)  
[ArraySlice.reduce\(\) \(ArraySlice method\), 21](#)  
[ArraySlice.reject\(\) \(ArraySlice method\), 21](#)  
[ArraySlice.shift\(\) \(ArraySlice method\), 21](#)  
[ArraySlice.toValue\(\) \(ArraySlice method\), 21](#)  
[ArraySlice.unshift\(\) \(ArraySlice method\), 21](#)  
[Asset\(\) \(class\), 19](#)  
[Asset.contentType \(Asset attribute\), 19](#)  
[Asset.href \(Asset attribute\), 19](#)  
[AuthScheme\(\) \(class\), 19](#)

## B

[BooleanElement\(\) \(class\), 9](#)

## C

[Category\(\) \(class\), 16](#)

`Category.authSchemeGroups` (*Category attribute*), 16  
`Category.authSchemes` (*Category attribute*), 16  
`Category.dataStructures` (*Category attribute*), 16  
`Category.resourceGroups` (*Category attribute*), 16  
`Category.resources` (*Category attribute*), 16  
`Category.transitions` (*Category attribute*), 16  
`Copy()` (*class*), 16  
`Copy.contentType` (*Copy attribute*), 16  
`Copy.copy` (*Copy attribute*), 16

## D

`DataStructure()` (*class*), 16

## E

`Element()` (*class*), 7  
`Element.attributes` (*Element attribute*), 7  
`Element.children` (*Element attribute*), 7  
`Element.classes` (*Element attribute*), 7  
`Element.clone()` (*Element method*), 7  
`Element.description` (*Element attribute*), 7  
`Element.element` (*Element attribute*), 7  
`Element.findRecursive()` (*Element method*), 7  
`Element.freeze()` (*Element method*), 8  
`Element.id` (*Element attribute*), 8  
`Element.isFrozen` (*Element attribute*), 8  
`Element.links` (*Element attribute*), 8  
`Element.meta` (*Element attribute*), 8  
`Element.parents` (*Element attribute*), 8  
`Element.recursiveChildren` (*Element attribute*), 8  
`Element.title` (*Element attribute*), 8  
`Element.toRef()` (*Element method*), 8  
`Element.toValue()` (*Element method*), 8  
`Enum()` (*class*), 16  
`Enum.enumerations` (*Enum attribute*), 17

## H

`HrefVariables()` (*class*), 19  
`HttpHeaders()` (*class*), 19  
`HttpMessagePayload()` (*class*), 18  
`HttpMessagePayload.contentType` (*HttpMessagePayload attribute*), 18  
`HttpMessagePayload.dataStructure` (*HttpMessagePayload attribute*), 18  
`HttpMessagePayload.headers` (*HttpMessagePayload attribute*), 18  
`HttpMessagePayload.messageBody` (*HttpMessagePayload attribute*), 18  
`HttpMessagePayload.messageBodySchema` (*HttpMessagePayload attribute*), 18  
`HttpRequest()` (*class*), 18

`HttpRequest.href` (*HttpRequest attribute*), 18  
`HttpRequest.method` (*HttpRequest attribute*), 18  
`HttpResponse()` (*class*), 18  
`HttpResponse.statusCode` (*HttpResponse attribute*), 19  
`HttpTransaction()` (*class*), 18  
`HttpTransaction.authSchemes` (*HttpTransaction attribute*), 18  
`HttpTransaction.request` (*HttpTransaction attribute*), 18  
`HttpTransaction.response` (*HttpTransaction attribute*), 18

## J

`JSONSerialiser()` (*class*), 6  
`JSONSerialiser.deserialise()` (*JSONSerialiser method*), 6  
`JSONSerialiser.serialise()` (*JSONSerialiser method*), 7

## L

`LinkElement()` (*class*), 14  
`LinkElement.href` (*LinkElement attribute*), 14  
`LinkElement.relation` (*LinkElement attribute*), 14

## M

`MemberElement()` (*class*), 14  
`MemberElement.key` (*MemberElement attribute*), 14  
`MemberElement.value` (*MemberElement attribute*), 14

## N

`Namespace()` (*class*), 6  
`Namespace.register()` (*Namespace method*), 6  
`Namespace.serialiser` (*Namespace attribute*), 6  
`Namespace.unregister()` (*Namespace method*), 6  
`Namespace.use()` (*Namespace method*), 6  
`NullElement()` (*class*), 9  
`NumberElement()` (*class*), 9

## O

`ObjectElement()` (*class*), 12  
`ObjectElement.compactMap()` (*ObjectElement method*), 12  
`ObjectElement.filter()` (*ObjectElement method*), 12  
`ObjectElement.forEach()` (*ObjectElement method*), 13  
`ObjectElement.get()` (*ObjectElement method*), 13  
`ObjectElement.getKey()` (*ObjectElement method*), 13  
`ObjectElement.getMember()` (*ObjectElement method*), 13

[ObjectElement.hasKey\(\)](#) (*ObjectElement method*), [13](#)  
[ObjectElement.items\(\)](#) (*ObjectElement method*), [13](#)  
[ObjectElement.keys\(\)](#) (*ObjectElement method*), [13](#)  
[ObjectElement.map\(\)](#) (*ObjectElement method*), [13](#)  
[ObjectElement.reject\(\)](#) (*ObjectElement method*), [13](#)  
[ObjectElement.remove\(\)](#) (*ObjectElement method*), [13](#)  
[ObjectElement.set\(\)](#) (*ObjectElement method*), [13](#)  
[ObjectElement.values\(\)](#) (*ObjectElement method*), [14](#)  
[ObjectSlice\(\)](#) (*class*), [21](#)  
[ObjectSlice.keys\(\)](#) (*ObjectSlice method*), [21](#)  
[ObjectSlice.values\(\)](#) (*ObjectSlice method*), [21](#)

## P

[ParseResult\(\)](#) (*class*), [15](#)  
[ParseResult.annotations](#) (*ParseResult attribute*), [15](#)  
[ParseResult.api](#) (*ParseResult attribute*), [15](#)  
[ParseResult.errors](#) (*ParseResult attribute*), [15](#)  
[ParseResult.sourceMapValue](#) (*ParseResult attribute*), [15](#)  
[ParseResult.warnings](#) (*ParseResult attribute*), [15](#)

## R

[RefElement\(\)](#) (*class*), [14](#)  
[RefElement.path](#) (*RefElement attribute*), [15](#)  
[Resource\(\)](#) (*class*), [17](#)  
[Resource.dataStructure](#) (*Resource attribute*), [17](#)  
[Resource.href](#) (*Resource attribute*), [17](#)  
[Resource.hrefVariables](#) (*Resource attribute*), [17](#)  
[Resource.transitions](#) (*Resource attribute*), [17](#)

## S

[SourceMap\(\)](#) (*class*), [15](#)  
[SourceMap.sourceMapValue](#) (*SourceMap attribute*), [15](#)  
[StringElement\(\)](#) (*class*), [9](#)  
[StringElement.length](#) (*StringElement attribute*), [9](#)

## T

[Transition\(\)](#) (*class*), [17](#)  
[Transition.contentType](#) (*Transition attribute*), [17](#)  
[Transition.data](#) (*Transition attribute*), [17](#)  
[Transition.href](#) (*Transition attribute*), [17](#)  
[Transition.hrefVariables](#) (*Transition attribute*), [17](#)  
[Transition.method](#) (*Transition attribute*), [17](#)