
API Elements (JS) Documentation

Release 1.0

Apiary

May 29, 2018

Contents

1	Installation	1
2	Usage	3
3	API Reference	5
3.1	API Reference	5
3.1.1	Namespace	6
3.1.2	Elements	7
3.1.3	Slice	17
4	Indices and tables	19

CHAPTER 1

Installation

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



```
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*

· <i>Annotation</i>
· <i>SourceMap</i>
* <i>API Description</i>
· <i>Category</i>
· <i>Copy</i>
· <i>Data Structure</i>
· <i>Resource</i>
· <i>HTTP Transaction</i>
– <i>Slice</i>

3.1.1 Namespace

class Namespace ()

A refactor 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 refactor element is encountered, and allows you to specify which elements get instantiated for existing Javascript objects.

Namespace.**register** ()

Register a new element class for an element.

Arguments

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

Namespace.**serialiser**

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

Namespace.**unregister** ()

Unregister a previously registered class for an element.

Arguments

- **name** (*string*) –

Namespace.**use** ()

Use a namespace plugin or load a generic plugin.

Arguments

- **plugin** –

Serialiser

class JSONSerialiser ()

Arguments

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

JSONSerialiser.**deserialise** ()

Arguments

- **value** (*object*) –

Returns Element –

JSONSerialiser.**serialise**()

Arguments

- **element** (*Element*) –

Returns object –

3.1.2 Elements

Element

class Element ()

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

Returns all of the children elements found within the element.

Element.classes

Element.clone ()

Creates a deep clone of the instance

Element.description

Human-readable description of element

Element.element

Element.findRecursive ()

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

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

`Element.isFrozen`
Returns whether the element is frozen.

`Element.links`

`Element.meta`

`Element.parents`
Returns all of the parent elements.

`Element.recursiveChildren`
Returns all of the children elements found within the element recursively.

`Element.title`
Human-readable title of element

`Element.toRef()`
Creates a reference pointing at the Element
Returns RefElement –

`Element.toValue()`

Primitives

String

`class StringElement()`

Arguments

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

`StringElement.length`
The length of the string.

Number

`class NumberElement()`

Arguments

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

Boolean

`class BooleanElement()`

Arguments

- `content` (*boolean*) –
- `meta` –

- **attributes** –

Null

```
class NullElement ()
```

Collections

Array

```
class ArrayElement ()
```

Arguments

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

```
ArrayElement.add ()
```

Arguments

- **value** –

```
ArrayElement.contains ()
```

Looks for matching children using deep equality

Arguments

- **value** –

Returns boolean –

```
ArrayElement.filter ()
```

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 ()
```

Recursively search all descendents using a condition function.

Arguments

- **condition** –

Returns ArraySlice –

```
ArrayElement.findByClass ()
```

Arguments

- **className** (*string*) –

Returns ArraySlice –

```
ArrayElement.findByElement ()
```

Arguments

- **element** (*string*) –

Returns `ArraySlice` –

`ArrayElement.findElements()`

Recursively search all descendents using a condition function.

Returns `Array.<Element>` –

`ArrayElement.first`

Return the first item in the collection

`ArrayElement.forEach()`

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()`

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`

Returns whether the collection is empty

`ArrayElement.last`

Return the last item in the collection

`ArrayElement.length`

Returns the length of the collection

`ArrayElement.map()`

Arguments

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

`ArrayElement.push()`

Arguments

- **value** –

`ArrayElement.reject()`

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`

Return the second item in the collection

`ArrayElement.set()`

`ArrayElement.shift()`

Returns `Element` –

`ArrayElement.unshift()`

Arguments

- **value** –

Object

`class ObjectElement()`

Arguments

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

`ObjectElement.filter()`

Arguments

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

Returns `ObjectSlice` –

`ObjectElement.forEach()`

Arguments

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

`ObjectElement.get()`

Arguments

- **key** –

Returns `Element` –

`ObjectElement.getKey()`

Arguments

- **key** –

Returns `Element` –

`ObjectElement.getMember()`

Arguments

- **key** –

Returns MemberElement –

`ObjectElement.hasKey()`

Returns boolean –

`ObjectElement.items()`

Returns array –

`ObjectElement.keys()`

`ObjectElement.map()`

Arguments

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

`ObjectElement.reject()`

Arguments

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

Returns ObjectSlice –

`ObjectElement.remove()`

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 ()

Arguments

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

`MemberElement.key`

`MemberElement.value`

Profiles

class LinkElement ()

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`

The URI for the given link.

`LinkElement.relation`

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

Referencing

class RefElement ()

Arguments

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

`RefElement.path`

Path of referenced element to transclude instead of element itself.

Parse Result

class ParseResult ()

Arguments

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

`ParseResult.api`

`ParseResult.sourceMapValue`

Annotation

class Annotation ()

Arguments

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

- `attributes` –

`Annotation.sourceMapValue`

SourceMap

`class SourceMap()`

Arguments

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

`SourceMap.sourceMapValue`

API Description

Category

`class Category()`

Arguments

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

`Category.resourceGroups`

Copy

`class Copy()`

Arguments

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

`Copy.contentType`

`Copy.copy`

Data Structure

`class DataStructure()`

Arguments

- `content` (*Element*) –
- `meta` –

- **attributes** –

class Enum()

Arguments

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

Enum.enumerations

Resource

class Resource()

Arguments

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

Resource.href

class Transition()

Arguments

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

Transition.method

HTTP Transaction

class HttpTransaction()

Arguments

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

HttpTransaction.request

class HttpMessagePayload()

Arguments

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

HttpMessagePayload.headers

class `HttpRequest()`

Arguments

- `content` –
- `meta` –
- `attributes` –

`HttpRequest.method`

class `HttpResponse()`

Arguments

- `content` –
- `meta` –
- `attributes` –

`HttpResponse.statusCode`

class `HttpHeaders()`

Arguments

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

class `Asset()`

Arguments

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

`Asset.contentType`

class `HrefVariables()`

Arguments

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

class `AuthScheme()`

Arguments

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

3.1.3 Slice

class `ArraySlice()`

Arguments

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

`ArraySlice.add()`

`ArraySlice.filter()`

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()`

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`

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

`ArraySlice.flatMap()`

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 `Element` –

`ArraySlice.forEach()`

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()`

Arguments

- **value** –

Returns boolean –

`ArraySlice.isEmpty`

Returns whether the slice is empty

`ArraySlice.length`

Returns the number of elements in the slice

`ArraySlice.map()`

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()`

Arguments

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

`ArraySlice.reject()`

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), 13
Annotation.sourceMapValue (Annotation attribute), 14
ArrayElement() (class), 9
ArrayElement.add() (ArrayElement method), 9
ArrayElement.contains() (ArrayElement method), 9
ArrayElement.filter() (ArrayElement method), 9
ArrayElement.find() (ArrayElement method), 9
ArrayElement.findByClass() (ArrayElement method), 9
ArrayElement.findByElement() (ArrayElement method), 9
ArrayElement.findElements() (ArrayElement method), 10
ArrayElement.first (ArrayElement attribute), 10
ArrayElement.forEach() (ArrayElement method), 10
ArrayElement.get() (ArrayElement method), 10
ArrayElement.getId() (ArrayElement method), 10
ArrayElement.getIndex() (ArrayElement method), 10
ArrayElement.getValue() (ArrayElement method), 10
ArrayElement.isEmpty (ArrayElement attribute), 10
ArrayElement.last (ArrayElement attribute), 10
ArrayElement.length (ArrayElement attribute), 10
ArrayElement.map() (ArrayElement method), 10
ArrayElement.push() (ArrayElement method), 10
ArrayElement.reject() (ArrayElement method), 10
ArrayElement.remove() (ArrayElement method), 11
ArrayElement.second (ArrayElement attribute), 11
ArrayElement.set() (ArrayElement method), 11
ArrayElement.shift() (ArrayElement method), 11
ArrayElement.unshift() (ArrayElement method), 11
ArraySlice() (class), 17
ArraySlice.add() (ArraySlice method), 17
ArraySlice.filter() (ArraySlice method), 17
ArraySlice.find() (ArraySlice method), 17
ArraySlice.first (ArraySlice attribute), 17
ArraySlice.flatMap() (ArraySlice method), 17
ArraySlice.forEach() (ArraySlice method), 17
ArraySlice.get() (ArraySlice method), 17
ArraySlice.getValue() (ArraySlice method), 17

ArraySlice.includes() (ArraySlice method), 17
ArraySlice.isEmpty (ArraySlice attribute), 18
ArraySlice.length (ArraySlice attribute), 18
ArraySlice.map() (ArraySlice method), 18
ArraySlice.push() (ArraySlice method), 18
ArraySlice.reduce() (ArraySlice method), 18
ArraySlice.reject() (ArraySlice method), 18
ArraySlice.shift() (ArraySlice method), 18
ArraySlice.toValue() (ArraySlice method), 18
ArraySlice.unshift() (ArraySlice method), 18
Asset() (class), 16
Asset.contentType (Asset attribute), 16
AuthScheme() (class), 16

B

BooleanElement() (class), 8

C

Category() (class), 14
Category.resourceGroups (Category attribute), 14
Copy() (class), 14
Copy.contentType (Copy attribute), 14
Copy.copy (Copy attribute), 14

D

DataStructure() (class), 14

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), 7
Element.id (Element attribute), 7
Element.isFrozen (Element attribute), 7

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), 15
Enum.enumerations (Enum attribute), 15

H

HrefVariables() (class), 16
HttpHeaders() (class), 16
HttpMessagePayload() (class), 15
HttpMessagePayload.headers (HttpMessagePayload attribute), 15
HttpRequest() (class), 15
HttpRequest.method (HttpRequest attribute), 16
HttpResponse() (class), 16
HttpResponse.statusCode (HttpResponse attribute), 16
HttpTransaction() (class), 15
HttpTransaction.request (HttpTransaction attribute), 15

J

JSONSerialiser() (class), 6
JSONSerialiser.deserialize() (JSONSerialiser method), 6
JSONSerialiser.serialize() (JSONSerialiser method), 7

L

LinkElement() (class), 13
LinkElement.href (LinkElement attribute), 13
LinkElement.relation (LinkElement attribute), 13

M

MemberElement() (class), 12
MemberElement.key (MemberElement attribute), 12
MemberElement.value (MemberElement attribute), 12

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), 8

O

ObjectElement() (class), 11
ObjectElement.filter() (ObjectElement method), 11
ObjectElement.forEach() (ObjectElement method), 11
ObjectElement.get() (ObjectElement method), 11
ObjectElement.getKey() (ObjectElement method), 11

ObjectElement.getMember() (ObjectElement method), 11
ObjectElement.hasKey() (ObjectElement method), 12
ObjectElement.items() (ObjectElement method), 12
ObjectElement.keys() (ObjectElement method), 12
ObjectElement.map() (ObjectElement method), 12
ObjectElement.reject() (ObjectElement method), 12
ObjectElement.remove() (ObjectElement method), 12
ObjectElement.set() (ObjectElement method), 12
ObjectElement.values() (ObjectElement method), 12
ObjectSlice() (class), 18
ObjectSlice.keys() (ObjectSlice method), 18
ObjectSlice.values() (ObjectSlice method), 18

P

ParseResult() (class), 13
ParseResult.api (ParseResult attribute), 13
ParseResult.sourceMapValue (ParseResult attribute), 13

R

RefElement() (class), 13
RefElement.path (RefElement attribute), 13
Resource() (class), 15
Resource.href (Resource attribute), 15

S

SourceMap() (class), 14
SourceMap.sourceMapValue (SourceMap attribute), 14
StringElement() (class), 8
StringElement.length (StringElement attribute), 8

T

Transition() (class), 15
Transition.method (Transition attribute), 15