API Elements (JS) Documentation

Release 1.0

Apiary

May 29, 2018

Contents

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);
```

CHAPTER 3

API Reference

3.1 API Reference

Contents

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**()

Register a new element class for an element.

Arguments

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

Namespace.**serialiser**

Convinience 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**()

Recusively 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**()
Recusively 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 begining of the slice

class ObjectSlice()

ObjectSlice.**keys**()

Returns **array** –

ObjectSlice.**values**()

Returns **array** –

CHAPTER 4

Indices and tables

- genindex
- search

Index

H

J

L

M

N

O

P

R

S

T