JSCA 1.0 Specification

Overview

This is a brief description of the JSCA file format. This JSON-based format is used to define content assist metadata for the Javascript Content Assist (CA) system. The structure of JSCA is described by another JSON-based format called JFF (JSON File Format).

Please note, that in general, properties on a JSON object may occur in any order; however, in order to simplify processing of JSCA files, it is required that any JSON objects having a "name" property must list that property before all others.

Short Example

```
{
  "types": [
    {
      "name": "Titanium.Android.NotificationManager",
      "examples": [],
      "functions": [
        {
          "name": "addEventListener",
          "parameters": [
            {
              "name": "name",
              "usage": "",
              "type": "String",
              "description": "name of the event"
            },
            {
              "name": "callback",
              "usage": "",
              "type": "Function",
              "description": "callback function to invoke when the event is fired"
            }
          ],
          "userAgents": [
            {
              "platform": "android"
            }
          ],
          "since": [
            {
              "version": "1.5",
              "name": "Titanium Mobile SDK"
            }
```

```
          ],
          "isConstructor": false,
          "isClassProperty": false,
          "examples": [],
          "isInternal": false,
          "exceptions": [],
          "references": [],
          "isMethod": true,
          "isInstanceProperty": true,
          "description": "add an event listener for the instance to receive view
triggered events"
        }
      ],
      "events": [],
      "userAgents": [
        {
          "platform": "android"
        }
      ],
      "remarks": [
        "<p>The constants above are taken from other docs</p>"
      ],
      "deprecated": false,
      "since": [
        {
          "version": "1.5",
          "name": "Titanium Mobile SDK"
        }
      ],
      "properties": [
        {
          "name": "DEFAULT_ALL",
          "isInternal": false,
          "isInstanceProperty": false,
          "since": [
            {
              "version": "1.5",
              "name": "Titanium Mobile SDK"
            }
          ],
          "examples": [],
          "userAgents": [
            {
              "platform": "android"
            }
          ],
          "type": "Number",
          "isClassProperty": true,
          "description": "<p>Use all default values (where applicable).</p>"
        }
      ],
      "description": "<p>Module to notify users of events that happen.</p>"
    }
  ],
  "aliases": [
    {
      "type": "Titanium",
      "name": "Ti"
    }
```

```
  ],
  "version": "1.0"
```

```
  "build": "1.7.0-201103300900"
}
```

Top-Level Properties

The JSCA file format consists of a single top-level object with the following properties: version, build, aliases, and types. Below is a description of each of these properties

```
{
  "version": "http://url-to-version-spec",
  "aliases": [],
  "types": []
}
```

Version Property

The version property indicates the file format version number of the JSCA file. The current version number is 1.0. It is expected that this value will change as new JSCA structures are defined.

```
"version": "1.0"
```

Build Property

The build property indicates the current build of the file

```
"build": "1.7.0-201103300900"
```

Aliases Property

The aliases property allows for the definition of shortcuts to deeply nested types. This property consists of an array of Alias types. This property is optional.

```
"aliases": [
    {
      "type": "Titanium",
      "name": "Ti"
    }
  ]
```

Types Property

This is perhaps the most important top-level property. This property contains a list of Type types used to defined the types in the JSCA file. This property is optional.

```
{
  "types": [
    {
      "name": "Titanium.Android.NotificationManager",
      "examples": [],
      "functions": [],
      "events": [],
      "userAgents": [],
      "remarks": [],
      "deprecated": false,
      "since": [],
      "properties": [],
      "description": "<p>Module to notify users of events that happen.</p>"
    },
    ...
  ],
  ...
}
```

Types

Below is a list of types declared in the JSCA JFF file. These types are what are used to describe the JS types contained within a JSCA file.

Alias Type

A type used to provide shortcuts to deeply nested types.

Name Property

The shortcut name to be used as an alias to a specific type. This property is required.

Description Property

A description of the alias. This property is optional. This property is optional.

Type Property

The type that this alias references via its shortcut. This property is required.

Type Type

A type is a named collection of properties and functions. A Type contains the following properties described below: name, description, deprecated, userAgents, since, inherits, properties, functions, events, remarks, and examples.

Name Property

This property is a string value that gives a unique name to the Type. It is expected that the value is a valid Javascript identifier. This property is required.

Description Property

This property gives a brief description of the type. Currently, this is not displayed in content assist, but future versions of Studio will present this information, so it is a good idea to include this property. This property is optional.

Deprecated Property

This property is a boolean flag indicating, when true, that this type should not be used. This property is optional.

UserAgents Property

This property is a list of UserAgent types and lists in which user agents this type exists. This property is optional.

Since Property

This property is a list of Since types and lists from which framework and version this type has been available. This property is optional.

Inherits Property

This property describes in a comma-delimited list the super-types for this type. If this property is not defined or is empty, then Object is assumed to be the type's super-type. This property is optional.

Properties Property

This property is a list of Property types, each describing the non-function members of this type. This property is optional.

Functions Property

This property is a list of Function type, each describing the function members (aka. methods) of this type. This property is optional.

Events Property

This property is a list of Event types, each describing an event and the values of the event object passed into the events callback. This property is optional.

Remarks Property

This property is a list of strings, each providing additional information about the this type. This property is optional.

Examples Property

This property is a list of Example types, each providing code examples of the type in use. This property is optional.

IsInternal Property

This property indicates if this type is internal only. Internal types do not appear in content assist but the properties of the type will still appear on items of the specified type. This property is optional.

UserAgent Type

A UserAgent type is used to capture information about runtime environments where types and their members are available. Studio also provides an extension point where icons for each user agent can be specified. UserAgents in conjunction with the user agent extension point allows developers to view via icons in which environments types and their members are available. A UserAgent consists of the following properties: platform, version, os, osVersion, and description.

Platform Property

This property indicates on which platform the user agent exists. This property is required.

Version Property

This property indicates the platform version. This property is optional.

OS Property

This property indicates a specific operating system on which the user agent runs. This property is optional.

OSVersion Property

This property indicates the OS version. This property is optional.

Description Property.

This property provides a description of this user agent. This property is optional.

Since Type

A Since type is used to indicate in which version a type or one of its members was added to a specific framework. A Since type consists of the following properties: name, version, and property.

Name Property

This property indicates the name of the framework or specification. This property is required.

Version Property

This property indicates the framework or specification number. This property is required.

Property Type

The Property type captures information about all non-function members for a given type. The Property type consists of the following properties: name, description, userAgents, since, isInstanceProperty, isClassProperty, isInternal, type, and examples.

Name Property

This property is the name of the Property instance. This property is required.

Description Property

This property provides a description of the Property instance. This property is optional.

UserAgents Property

This property lists all user agents in which this Property exists. This property is optional.

Since Property

This property lists all framework and specification in which this Property is defined. This property is optional.

IsInstanceProperty Property

This property indicates if this Property is a member of instances of its owning type. This property is optional.

IsClassProperty Property

This property indicates if this Property is a member of the owning type's type. This property is optional.

IsInternal Property

This property indicates if this property is internal only. Internal properties do not appear in content assist. This property is optional.

Type Property

This property indicates the type of the Property instance. This property is required.

Examples Property

This property allows examples to be created showing the Property in use. This property is optional.

Permission Property

This property indicates if the property is "read-only", "write-only" or "read-write" (default). This property is optional.

Availability Property

This property indicates if the property is "creation" (only invoked when creating the object), "not-creation" (only invoked on the object) or "always" (default). This property is optional.

Constants Property

This property provides an array of strings that represent the possible Titanium constants that the property can be assigned. This property is optional.

Function Type

The Function type captures information about all function members for a given type. The Function type consists of the following properties: name, description, userAgents, since, isInstanceProperty, isClassProperty, isInternal, examples, references, returnTypes, isConstructor, and isMethod.

Name Property

This property is the name of the Function instance. This property is required.

Description Property

This property provides a description of the function. This property is optional.

UserAgents Property

This property lists all user agents in which the function exists. This property is optional.

Since Property

This property lists all framework and specification in which the function is defined. This property is optional.

IsInstanceProperty Property

This property indicates if the function is a member of instances of its owning type. This property is optional.

IsClassProperty Property

This property indicates if the function is a member of the owning type's type. This property is optional.

IsInternal Property

This property indicates if the function is internal only. Internal properties do not appear in content assist. This property is optional.

Examples Property

This property is a list of examples, showing the function in use. This property is optional.

Parameters Property

This property is a list of parameters used when invoking the function. This property is optional.

References Property

This property is a list of references where you can find additional and/or related information on the function. This property is optional.

Exceptions Property

This property is a list of exceptions that may occur when invoking the function. This property is optional.

ReturnTypes Property

This property is a list of return values that the function can return. This property is optional.

IsConstructor Property

This property indicates if the function can be used as a constructor of type instances. This property is optional.

IsMethod Property

This property indicates if the function is a method on a type instance. This property is optional.

Event Type

The Event type captures information about the events that a given type listens to. The Event type consists of the following properties: name, description, and properties.

Name Property

This property gives a name to the event. This is the same name used when registering the event via addEventListener. This property is required.

Description Property

This property provides a description of the event. This property is optional.

Properties Property

This property provides a list of EventProperty instances which describe the properties of the event object passed into the event's callback/listener. This property is optional.

EventProperty Type

The EventProperty type captures the types of properties contained in the event object passed to an event's callback. The EventProperty type consists of the following properties: name, description, and type.

Name Property

This property gives a name to the event property. This property is required.

Description Property

This property provides a description of the event property. This property is optional.

Type Property

This property indicates the type of the event property. This property is required.

Constants Property

This property provides an array of strings that represent the possible Titanium constants that the property can be assigned. This property is optional.

ReturnType Type

The ReturnType type is used to capture information about function return types. The ReturnType type consists of the following properties: type and description.

Type Property

This property indicates the type of the return type. This property is required.

Description Property

This property provides a description of the return type. This property is optional.

Example Type

The Example type is used to provide examples of how a given type can be used. The Example type consists of the following properties: name and code.

Name Property

This property gives a title to the example. This property is required.

Code Property

This property provides a code sample for the example. This property is required.

Parameter Type

The Parameter type captures information about a function's parameters. The Parameter type consists of the following properties: name, type, usage, and description.

Name Property

This property gives a name to the parameter. This property is required.

Type Property

This property indicates the type of the parameter. This property is required.

Usage Property

This property indicates the usage of this parameter. Possible values are "required", "optional", "zero-or-more", and "one-or-more". If the property is not defined, then this defaults to "required". This property is optional.

Description Property

This property provides a description of the Parameter. This property is optional.

Constants Property

This property provides an array of strings that represent the possible Titanium constants that the parameter can take. This property is optional.

Exception Type

The Exception type is used to capture exception types that may be thrown by a function when it is invoked. The Exception type consists of the following properties: type and description.

Type Property

This property indicates the type of the Exception. This property is required.

Description Property

This property provides a description of the exception, possibly explaining in which situations it may occur. This property is optional.