
arg.js

Arg.js gives you quick and easy access to parameters in the URL.

- Installing
- Changes
- Usage
- License

Installing

- Download your own copy
- Package manager: NuGet, Bower
 - *we would love to include more here, please send us Pull Requests*

Changes

v1.3

- BUG: Empty arrays result in extra &&&&

v1.2

- Simplified project structure and file names
- Minified version included in /dist
- Resolves decoding URL issues #17
- Fix for `Arg.query()` in IE 8 #19
- Use `hasOwnProperty` when looping through arguments

v1.1

- Added `Arg(key)` shorter interface as well as `Arg.get(key)`.
- Ignores undefined/empty keys and values.
- Cleans up edge cases (i.e. where paths are present in `parse()` calls etc).
- Will now optionally coerce a native type out of value if possible (i.e. Number, Boolean, undefined, etc). To not coerce, set `Arg.coerceMode = false`
- Better handling of complex objects that have mixed nested objects/arrays. See new test case added to `test/spec/arg.js` for an example object that was failing and is no longer failing.

-
- Added support for anchors in `Arg.url(path, params, anchorString)` (i.e. no longer assumes they're variables if it's a string)

v1

- Launch

People who like arg.js, also like:

- over.js - Elegant function overloading in JavaScript

Usage

Getting stuff

The examples here assume this path:

```
1 page.html?name=Mat&address[0].city=London&address[0].country=UK&address[1].city=Boulder&address[1].country=US#?fromhash=true
```

Get a single value

```
1 Arg("name")
2 //="Mat"
```

It will get the value from both the query segment, and the hash segment.

```
1 Arg("fromhash")
2 //="true"
```

Get an array

```
1 Arg("address")
2 //=[
3 //  { city: "London", country: "UK" },
4 //  { city: "Boulder", country: "US" }
5 // ]
```

Get an object

```
1 Arg("address[0]")
2 //={ city: "London", country: "UK" }
```

Get a field from an object in an array

```
1 Arg("address[0].city")
2 // = "London"
```

Get with a default value

```
1 Arg("address[0].something", "Unknown")
2 // = "Unknown"
```

Getting everything

Everything with Arg.all()

```
1 Arg.all()
2 // = {
3 //   address: [
4 //     { city: "London", country: "UK" },
5 //     { city: "Boulder", country: "US" }
6 //   ],
7 //   fromhash: "true",
8 //   name: "Mat"
9 // }
```

- `Arg.all()` gets all parameters (from the query and hash segments) in one object. Optionally, you can use the `query` or `hash` methods to be specific.

Just the query segment with Arg.query() `Arg.query()` gets an object made up of all the values in the query segment of the URL. The query segment is everything following the initial `?`, but before the `#` (if there is one.)

```
1 Arg.query()
2 // = {
3 //   address: [
4 //     { city: "London", country: "UK" },
5 //     { city: "Boulder", country: "US" }
6 //   ],
7 //   name: "Mat"
8 // }
```

- Notice how the `fromhash` value is missing.

Just the hash segment with Arg.hash() `Arg.hash()` gets an object made up of all the values in the hash segment of the URL. The hash segment is anything following the `#`.

```
1 Arg.hash()
2 // = {
3 //   fromhash: "true"
```

```
4 // }
```

Parsing your own strings with `Arg.parse()` Instead of using the current URL, you can be explicit by using the `Arg.parse` method.

```
1 var myArgs = "name=Mat&company=Stretchr";
2 Arg.parse(myArgs);
3 // = {
4 //   name: "Mat",
5 //   company: "Stretchr"
6 // }
```

Building URLs and querystrings

`Arg.url()` helper

The `Arg.url()` function builds a URL, and has a few overloaded versions.

`Arg.url(params)` - just the params Passing just an object will generate a URL based on the current location, just changing the parameters.

```
1 Arg.url({name: "Mat", company: "Stretchr"});
2 // = "path/to/current/page?name=Mat&company=Stretchr"
```

If you set `Arg.urlUseHash = true`, then the parameters will be placed in the hash segment of the new URL following the `#?` separator:

```
1 Arg.urlUseHash = true;
2 Arg.url({name: "Mat", company: "Stretchr"});
3 // = "path/to/current/page#?name=Mat&company=Stretchr"
```

`Arg.url(path, params)` - explicit path Being explicit about a path in the first argument will use that location instead.

```
1 Arg.url("http://www.stretchr.com/", {name: "Mat", company: "Stretchr"})
2 // = "http://www.stretchr.com/?name=Mat&company=Stretchr"
```

`Arg.url(path, query, hash)` - explicit query and hash parameters in one URL If you want to use query and hash paremeters, pass a path and two objects.

```
1 Arg.url("http://www.stretchr.com/", {name: "Mat", company: "Stretchr"},
  {comment: 123});
2 //= "http://www.stretchr.com/?name=Mat&company=Stretchr#?comment=123";
```

Arg.stringify The `Arg.stringify` method lets you easily encode an object into a query string.

```
1 Arg.stringify({ name: "Mat" });
2 //= name=Mat
```

Encoding objects

```
1 Arg.stringify({ one: { two: { three: 3 }}});
2 //= one.two.three=3
```

Encoding arrays

```
1 Arg.stringify({list:["one","two","three"]});
2 //= list[0]=one&list[1]=two&list[2]=three
```

License

by Mat Ryer and Ryan Quinn Copyright (c) 2013 Stretchr, Inc.

Please consider promoting this project if you find it useful.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.