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 though 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:

```
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

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

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

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

Get an array

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

Get an object

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

Get a field from an object in an array

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

Get with a default value

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

Getting everything

Everything with `Arg.all()`

```
Arg.all()
//= {
//    address: [
//      { city: "London", country: "UK" },
//      { city: "Boulder", country: "US" }
//    ],
//    fromhash: "true",
//    name: "Mat"
//  }
```

- `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.)

```
Arg.query()
//= {
//    address: [
//      { city: "London", country: "UK" },
//      { city: "Boulder", country: "US" }
//    ],
//    name: "Mat"
//  }
```

- 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 #.

```
Arg.hash()
//= {
//    fromhash: "true"
//   }
```

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

```
var myArgs = "name=Mat&company=Stretchr";
Arg.parse(myArgs);
//= {
//     name: "Mat",
//     company: "Stretchr"
//   }
```

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.

```
Arg.url({name: "Mat", company: "Stretchr"});
//= "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 #? seperator:

```
Arg.urlUseHash = true;
Arg.url({name: "Mat", company: "Stretchr"});
//= "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.

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

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

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

Encoding objects

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

Encoding arrays

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

License