

JavaScript Standard Library

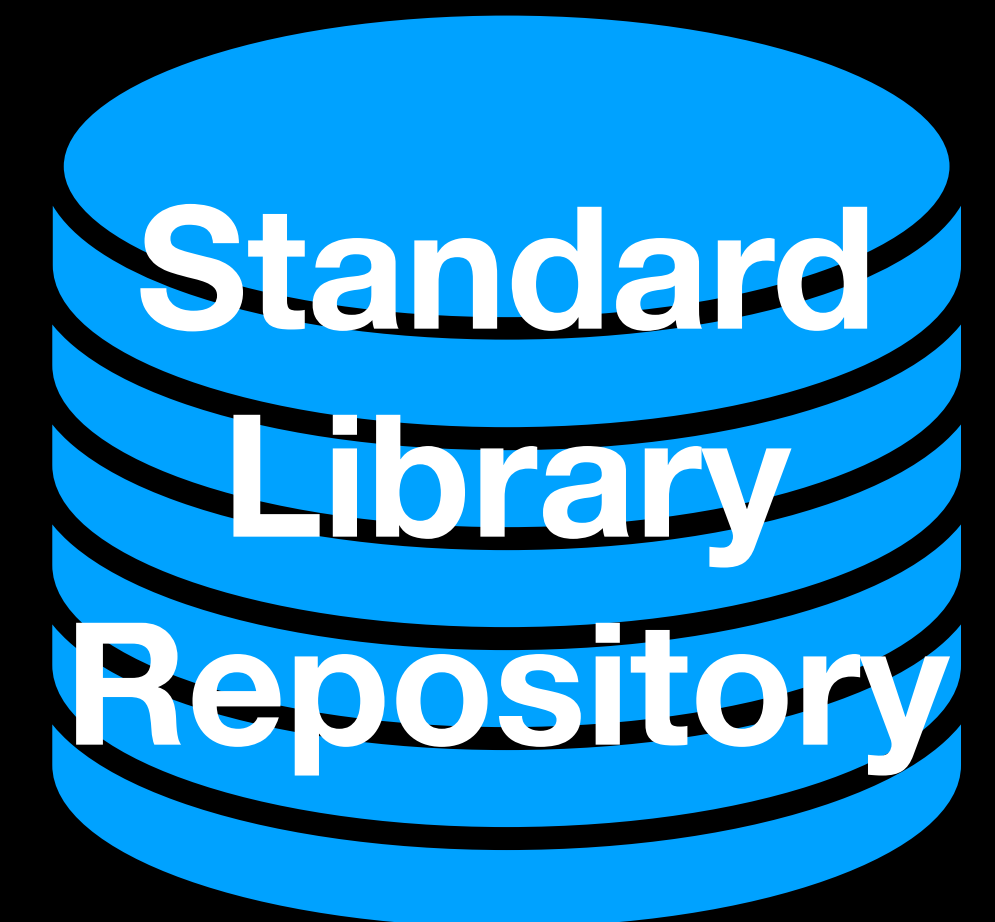
for stage 1

Nathan Hammond
Mattijs Hoitink
Michael Saboff

Goals

- Structure to deploy standardized JavaScript functionality
- Provide common functionality that doesn't need to be downloaded
- Help reduce global namespace bloat
- Enable safe extensibility of library features

Overview



JavaScript Engine
(Includes Built-ins)

Overview

foo.js

**Standard
Library
Repository**

JavaScript Engine
(Includes Built-ins)

Overview

foo.js

```
import { A } from "std:A";
```

std:A

...

```
exports = {  
  A: A;  
};
```



JavaScript Engine
(Includes Built-ins)

Overview

foo.js

```
import { A } from "std:A";  
  
let myA = new A();  
...
```

std:A

```
...  
  
exports = {  
  A: A;  
};
```



JavaScript Engine
(Includes Built-ins)

Overview

foo.js

```
import { A } from "std:A";  
  
let myA = new A();  
...
```

std:A

```
...  
  
exports = {  
  A: A;  
};
```

A stylized icon representing a database or repository, consisting of several horizontal blue cylinders stacked on top of each other. The text "Standard Library Repository" is written in white across the cylinders.

**Standard
Library
Repository**

JavaScript Engine
(Includes Built-ins)

Example

```
import { Statistics } from "std:Statistics";  
  
let stats = new Statistics([6, 1, 7, 3, 4, 9, 8]);  
  
let sum = stats.sum(); // 38;  
let median = stats.median(); // 6  
let geoMean = stats.geomean(); // 4.4812040415131  
let stdev = stats.stdev(); // 2.6649654437397
```


Generics

```
import{ len, map } from "std:builtins";
```

```
const ar = [1, 2, 3];  
const st = new Set([4, 5, 6, 7]);
```

```
len(ar); // => 3  
len(st); // => 4
```

```
map(ar, (val, idx) => val * 2); // => Array {2, 4, 6}  
map(st, (val, idx) => val * 2); // => Set {8, 10, 12, 14}
```


Someone proposes adding
new feature to an existing builtin ...

Someone proposes adding
new feature to an existing builtin ...

Let's add `Array.flatten()`

Someone proposes adding
new feature to an existing builtin ...

Let's add Array. **smooshed!!**



Safe Extensibility

- Imported objects are frozen
- Users extend via inheritance, wrapping, ...

Extending Library

```
import { Statistics } from "std:Statistics";
```

Extending Library

```
import { Statistics } from "std:Statistics";  
Statistics.prototype.quantiles = (q) => { ... };
```


Extending Library

```
import { Statistics } from "std:Statistics";  
Statistics.prototype.quantiles => { ... };
```

Throws readonly error

Extending Library

```
import { Statistics } from "std:Statistics";  
  
class QuantileStats extends Statistics  
{  
    constructor(q) { }  
    ...  
}
```

Other Questions to Consider

- What features should go into the standard library versus the core language?
- Is there a different stage process for library components?
- Can library components be spec'ed in Javascript?
- How should we collaborate with Nodejs and Web on Standard Libraries

Future Work

- Design polyfill fallback support
- Safe version updates to existing library modules

Questions