Extensible, Durable JavaScript with Inheritance

Writing great, extensible code with that dirty, naughty little language.

Mike Hodnick
github.com/kindohm

"#tccc12, or, The TCCC Event When JavaScript Took Over"

https://twitter.com/#!/TCCodeCamp/status/175282282427453440

March 1, 2012

JavaScript is a **first-class** citizen

Just how dirty and naughty is JavaScript?

Extensibility
&
Durability

a bad example...

How do you do inheritance in JavaScript?

Wish List

- Inherit properties and methods
- Super methods
- Initialization logic
- Private state
- Efficient
- x instanceof X

Road Map

- The three basic Crockford approaches
- Super methods
- Private state
- Initialization
- 3rd party approaches

Classical inheritance.

dirty attempt at a class-based system.

Prototypal inheritance.

just the objects.

Functional inheritance.

pretty. expressive.

Super-methods

"...*I have never once found need to use an* `uber` *function.*"

Douglas Crockford
http://javascript.crockford.com/inheritance.html

private state

does it matter?

initialization

3rd-party libraries

backbone, prototype, base2, joose, jsclass...

pd

`npm install pd`

https://github.com/Raynos/pd

prototype.js

http://prototypejs.org

Recommendations:

1) Expose extensibility through an API

or

2) Require a 3rd party library/convention

Consider:

1) Do you care about private state?

2) Do you care about efficiency?

References

Douglas Crockford
http://www.amazon.com/JavaScript-Good-Parts-Douglas-Crockford/dp/0596517742
http://www.crockford.com/javascript/inheritance.html
http://javascript.crockford.com/prototypal.html

Jake Verbaten
http://raynos.org/blog
https://github.com/Raynos/pd

John Resig
http://ejohn.org/blog/simple-javascript-inheritance/

3rd party libs
http://jsclass.jcoglan.com
http://code.google.com/p/joose-js
http://code.google.com/p/base2
http://prototypejs.org/learn/class-inheritance

Other
http://www.adequatelygood.com/2010/3/JavaScript-Module-Pattern-In-Depth

github.com/kindohm