Welcome our
new ES5 Overlords
Hello
A good time to talk about ES5
A good time to talk about ES5

Every current browser supports ES5
A good time to talk about ES5

Every current browser supports ES5

Every previous generation browser supports ES5
A good time to talk about ES5

Every current browser supports ES5

Every previous generation browser supports ES5

In a few weeks every browser before that will support ES5
A good time to talk about ES5

Every current browser supports ES5
Every previous generation browser supports ES5
In a few weeks every browser before that will support ES5
Node is ES5
A good time to talk about ES5

Every current browser supports ES5
Every previous generation browser supports ES5
In a few weeks every browser before that will support ES5
Node is ES5
IE8 may not be a requirement in your next project
ES5

Some new methods
ES5

Some new methods

Which are convenient
ES5

Some new methods
Which are convenient
That we ignore
There is magic in ES5
Some JS

A list of German bands

Clicking the button should show the band name

```javascript
var bands = ['Apparat', 'Boy', 'Kraftklub'];
for (var i = 0; i < bands.length; i++) {
  var band = bands[i],
  button = document.createElement('button');
  button.appendChild(document.createTextNode(band));
  button.addEventListener('click', function() {
    alert(band);
  });
  document.body.appendChild(button);
}
```
This code has two problems
This code has two problems

Valeska from 'Boy' is actually Swiss
This code has two problems

Valeska from 'Boy' is actually Swiss

When the loop is finished, 'band' has the last value, and that's what the inner function 'sees'.
This means we can only see KraftKlub
This means we can only see KraftKlub

We don't want to that
This means we can only see KraftKlub

We don't want to that

Felix is a poser
Some basic JS (fixed)

```javascript
var bands = ['Apparat', 'Boy', 'Kraftklub'];
for (var i = 0; i < bands.length; i++) {
    var band = bands[i],
    button = document.createElement('button');
    button.appendChild(document.createTextNode(band));
    (function(band){
        button.addEventListener('click', function(){
            alert(band);
        });
    })(band);
    document.body.appendChild(button);
}
```
Same thing in ES5

`['Apparat', 'Boy', 'Kraftklub'].forEach(function(band) {
    var button = document.createElement('button');
    button.appendChild(document.createTextNode(band));
    button.addEventListener('click', function() {
        alert(band);
    });
    document.body.appendChild(button);
})`
1/3rd less code
1/3rd less code

Looks nice
1/3rd less code
Looks nice
Functions are more 'natural' fit for JS than older 'for' loops
Let's do it for more things!
Safe Extension of Inbuilt Methods
No, really
Quick History* Lesson

*History may be more recent than expected
ES3: Non-native methods appear during iteration

```javascript
Object.prototype.oldStyleMethod = function oldStyleMethod (){}
var someObject = {};
for (var key in someObject) {
  console.log(key);
}
```
ES3: But native methods don't

This is why toString() doesn't appear in 'for' loops.

```javascript
console.log(Object.prototype.toString);
function toString() { [native code] };
```
Added methods are always enumerable in ES3

So they always appear in 'for' loops
Added methods are always enumerable in ES3

So they always appear in 'for' loops

Extending prototypes in ES3 can work if the entire universe changes their 'for' loops
Added methods are *always* enumerable in ES3

So they *always appear* in 'for' loops

Extending prototypes in ES3 can work if the *entire universe changes* their 'for' loops

Surprisingly this *not happen*
Added methods are **always** enumerable in ES3

So they *always appear* in 'for' loops

Extending prototypes in ES3 can work if the **entire universe changes** their 'for' loops

Surprisingly this not happen

So extending prototypes in ES3 is risky
ES5: Non enumerable methods can be added

Requires native ES5 (not shimmable)

```javascript
Object.defineProperty(Object.prototype, "newStyleMethod", {
  value: function newStyleMethod(){}
  enumerable: false
});
for (var key in someObject) { console.log(key) }
```
That's not the only problem
Generic names

Past conflicts:

String.prototype.namespace()
Generic names

Past conflicts:

String.prototype.namespace()

Array.prototype.find()
Prefixing
Set a sensible prefix
<table>
<thead>
<tr>
<th>Feature</th>
<th>Underscore</th>
<th>Sugar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Regular 'for' loops</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Conflict-free</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Underscore</td>
<td>Agave (ES5 only)</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Methods</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Regular 'for' loops</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Conflict-free</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Using ES5 defineProperty() and prefixing, Agave.JS has had no conflicts since it was created (early 2012).
Other reasons:

"You can do that. You should do that."

Brendan Eich, JQuery UK, 19 Apr 2013
Other reasons:

"You can do that. You should do that."

Brendan Eich, JQuery UK, 19 Apr 2013

Ember does it.

Ember JS Prototype Extensions
Other reasons:

"You can do that. You should do that."

Brendan Eich, JQuery UK, 19 Apr 2013

Ember does it.

Ember JS Prototype Extensions

I just gave you a happy hippo and now we are friends.
More magic: *Live Binding*
An experiment in two parts

1. A data → DOM binding (I like mustache, so I use Ractive).
An experiment in two parts

1. A data → DOM binding (I like mustache, so I use Ractive).
2. Data changes applied to binding live via object.defineProperty()
Live binding with defineProperty

```javascript
var livebind = function(object, binding, properties){
    properties.forEach(function(property){
        var hiddenProperty = '_' + property
        Object.defineProperty(object, property, {
            get: function(){ return testData[hiddenProperty]; },
            set: function(newValue){
                testData[hiddenProperty] = newValue;
                binding.set(property, newValue)
            },
            enumerable: true,
            configurable: true
        });
    });
}
```
Note

1. This is an experiment
Note

1. This is an experiment

2. We also use prototype chain injection (see links) for array.length magic
ES5-only is coming

For many, it's already here
ES5-only is coming

For many, it's already here
Use ES5 methods directly
ES5-only is coming

For many, it's already here
Use ES5 methods directly
Don't be scared to extend native prototypes
ES5-only is coming

For many, it’s already here

Use ES5 methods directly

Don’t be scared to extend native prototypes

Experiment
Thanks.

@mikemaccana

Enjoy the week