ES6 draft status

November 2013

Status

- Spreadsheet https://skydrive.live.com/view.aspx?resid=704A682DC00D8AAD!59602&app=Excel&authkey=!AAMixsO0TuyPYwc

- http://wiki.ecmascript.org/doku.php?id=harmony:specification_drafts
- Parameterized Grammar productions
- Unreserve 'let' in sloppy mode, let[x] = expr; is a declaration not an assignment.
- Unicode RegExp
- Webv Reality RegExp Annex
- Module Syntax and static semantics
- Specified how to determine if a call is in tail position
- Eliminated the [[Invoke]] MOP operation
- Spread now requires an Iterable rather than an array-like
- Replaced JSON grammar with normative reference to ECMA-404
- Updated toLocale*String methods with reference Ecma-402

- Currently having major Word issues.

Strict Formal Parameters

- Decouple strict formal parameters and eval/ arguments declaration restrictions
- strict mode code
  - can't creating bindings for 'eval or 'argumentys'
- StrictFormalParameter and any parameter list with new parameter syntax
  - can't have duplicated param names, get's an array as its arguments object

- Do arrow functions have an arguments object?

[Computed Property Keys]

- No dynamic checking for duplicate computed property names in object literals or classes:

```
{[expr1]: 1
[[expr2]: 2
}  //does not check if expr1 == expr2
```

Class/ optional yield arg ambiguity

```
functtion *g() {
    class foo extends yield { }  //are those braces the class body?
    { }
}
```

- Proposed solutions

1) disallow trailing yield in extends clause
   - requires an extra parameter to Expression and AssignmentExpression

2) extends LeftHandSideExpression
   - would be only place an expression isn't explicitly Expression or AssignmentExpression

Cross-Realm Symbol registration

- https://mail.mozilla.org/pipermail/es-discuss/2013-September/033799.html
- https://mail.mozilla.org/pipermail/es-discuss/2013-September/033801.html

Symbol.for(aString) ==> aSymbol   //creates a new Symbol if key is not registered.

Symbol.keyFor(aSymbol) ==> aString

- where for all strings S:
  Symbol.keyFor(Symbol.for(S)) === S

- the use case for Symbol.keyFor is serialization

Introduce a prototype object to contains sloppy arguments object @@iterator function?

Conventions for ignore over-ride of @@iterator, etc.

- Property whose value is undefined. Should null value also mean not available.

- or just ToBoolean??

(function Foo() {}).bind(x).name ??

- treat name for bound functions like anonymous functions or try to compute a new name derived from target function. eg,

```
name: 'bound ' + this.[[target]].name

get name() {return 'bound '+this.[[target]].name}
```

- should avoid unnecessary extra computation when binding a function

time to obsolete statement about native objects??

- The map function is intentionally generic; it does not require that its this value be an Array object. Therefore it can be transferred to other kinds of objects for use as a method. *Whether the map function can be applied successfully to an* ***exotic*** **[native?]** *object that is not an Array is implementation-dependent.*

super and object literals

- Issue: How do you mixin some methods that reference super?
- Object.mixin(obj, ???);

Super is currently explicitly illegal within an object literal

```
Object.mixin(obj, {
    toString() {
        return `mixed(${super.toString()})`
    }
});
```

Static Error

Current Workarounds

```
Object.mixin(obj, class {
  toString() {
    return `mixed(${super.toString()})`
  }.prototype
});
```

```
Object.mixin(obj, class {
  static toString() {
    return `mixed(${super.toString()})`
  }
});
```

Fix?

- Remove restriction on super in object literals

- Concerns
  - The reason for the restriction was that some of us were worried about the foot-gun potential of super in objet literals.