Hi Istvan,

Sorry for not sending you the notes earlier.

Dear TC39 members, feel free to point out any errors and clear omissions since these notes are very rough at best.

Opt-in versioning?

```html
<script type="application/ecmascript; version=5">"use strict";
...
</script>

<script type="application/javascript; version=1.9">let yield
...
</script>
```

version=preharmony

```
  ph1
  ph2
```

ephemeron/name/cacthall

Mark: Page level meta tag?
Brendan: We talked about this before

Mike Samuel: Inline code to make sure that using a script by URL/path works:

use version harmony

This is attractive

Allen: Do we need to add namespaces
Brendan: If we only add a dozen items then we might get away without
Mark: Modules
Cormac: Do we need a way to allow setting a limitation on version per page. The example is that newer versions might add catchalls which changes the security model.
Mark: Another way to tackle this is to document the constraints under which future versions and extensions may extend the language.
Brendan: Use lexical scope for example

Waldemar:

```java
{
  double a;
  double b;
```
... if (a < b) ... true
if (a < b) ... false
if (a < b) ... true
}

Brainstorm/discussion about host objects etc

Allen: As long as the ES5 spec is followed and no host objects or extensions are present reading a property is guaranteed to return the last value that was set.

8.9: Object model
Execution model

Mark: make functions be the link to the environment. THe objects are native but its methods might be host objects

Brendan: The idea of taming host objects is something we should pursue.

Thursday 2009-09-24

Promises in E:

def r := a.foo(b, c) // sync
def p := a<-foo(b, c) // async, eventually do
def x := when(x, q)->{
... x ... q ...
} catch(ex) { // optional
... ex ...
}

def p := race([a, b, c, ...])
def p := timeBomb(millis, ex)
def p := race(a<-foo(b, c), timeBomb(3000, 'oops'))

def p := when(timeOut(3000)) -> {
... 
}

Brendan: ES next 2-3 years June GA 2012. Feature freeze in May 2011 (20 months). Definitional interpreter

Mark: Ephemerons require new kernel state
Allen: Weak refs as well
Mark: As soon as we introduce visible collection we need to express that Waldemar/Allen/Mark: That can be done in prose

W: Grammar needs to be tightly integrated
W: Grammar needs to be tightly integrated
A: We need a mapping at least

Fresh let or not in for (let i = 0; ...; i++)? Consensus to not get a new var.

Mark:

```javascript
const a = [];
for (let i = 0; i < 3; i++) {
    a.push(function() { return i * i; });
}
a[0](); // 4
```

Brendan:

```javascript
for (let i in o) {
    a.push(...)
}
for (const i in o) {
    a.push(...)
}
```

(for (var i = E in o) {...} is valid in ES today)

No consensus after all?

Rob:

```javascript
for (let x = []; x.length < 3; x.push(42)) {
    ...
}
```

Specify iteration order for ESH

Mark: Generator and finally?
Brendan: This has been solved in Python and Spidermonkey

Mark: return to label?
Brendan: Not without lambdas
Brendan: Maciej objected on the mailing list. Probably due to implementation issues.
Allen: It is easy to implement

Brendan:

```javascript
function gen() {
    while (...) {
        try {
            yield x;
        } finally {
            ...
        }
    }
}
```

```javascript
g = gen();
```
g = gen();
g.next();
g.throw(e);

Let

let x;

x is undefined

redeclaration of let should be forbidden

for (var k in keys(o))
for (var v in values(o))
for each (var x in anIter)

Ephemerons

Allen: Adds overhead to the GC since the ephemeron has to be handled in a second pass