Instantiation Reform

TC39 July 2014
MM, DL, AWB, TVC -- AWB Champion
Thanks to Claude Pache

Goals

Subclass exotics

Avoid un (or partially) initialized exotics

Full ES5 compat

ES6 class compat, aside from @@create

Reliable test for am-i-called-as-constructor?

Support base-creates-proxy scenario

Simple Story

```
class Derived extends Base {
    constructor(...args) {
        // TDZ this, on “new Derived…” etc
        super(...otherArgs); // this = what super returns
        // this is initialized
}   }
function Base(...otherArgs)
    // implicit this = Object.create(mostDerived.prototype, {});
```

From Claude Pache

F.[[Construct]](args, rcvr)

Distinguish functions-which-call-super

Vanilla func at end of super-call-chain is base

- instantiation postponed to base-entry

From Claude Pache *with mods*

F.[[Construct]](args, rcvr)

mod: Only MOP signature change

Distinguish functions-which-call-super

mod: …-call-super-as-a-function

super(..), but not super.foo(..)

Vanilla func at end of super-call-chain is base

instantiation postponed to base-entry

[[Call]] traps

F(...args) → F.[[Call]](undefined, args)

Derived.[[Call]](const this, args)

super(...other) → super.special_name(...other)

[[Construct]] traps

new F(...args) → F.[[Construct]](args, F)

Base.[[Construct]](rcvr, args)

entry → const this = [[Create]](rcvr.prototype)

Derived.[[Construct]](args, rcvr)

entry → TDZ this

super(...other) → const this = super.[[Construct]](other, rcvr)

Remaining Requirements

Am I called as a constructor?

What is the original's constructor's .prototype

How do I provide alternate instance to subclasses?

Am I called as a Constructor?

```
F(...other) {
    let constructing = false;
    try { this; } catch(_) { constructing = true; }
    super(..);
    ..
}
```

Base instantiates proxy scenario

```
Base(...other) {
    return new Proxy(... this.__proto__ …);
}
```

Or, kill two birds with “new*”

```
function Date() {
    let now = $$GetSystemTime();
    if (new* === void 0) {
        let obj = Object.create(new*.prototype);
        // obj@now = now; // private “now” state
        return obj;
    } else {
        return ToTimeString(now);
}   }
```

Reflection and Proxies

Reflect.construct(F, args, rcvr) // throw on undefined

construct trap:

construct: function(target, args, rcvr)