Generators in the JavaScript Specification : Icky Implementation Details

- JavaScript's poorly enforced pointer-naming conventions were already messy.
  - .prototype?
  - .constructor?
  - instanceof?
  - … forget certainty. Anything can mean anything.

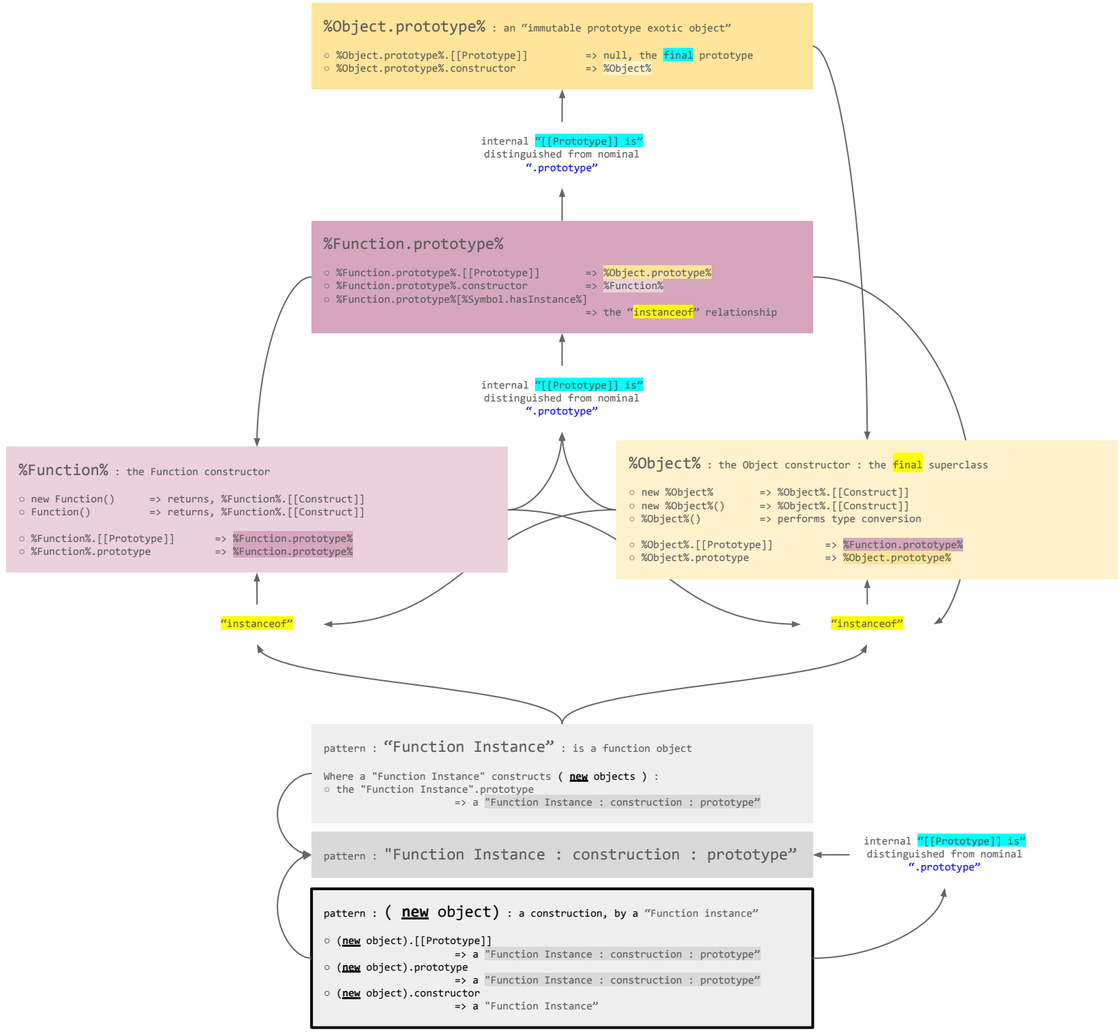
- The Generators design made the most of free-will. Designers appear to have named things based on convenience to each type of object, without regard for consistency about the environment as a whole. It is thus futile to try and present 4 to 5 dimensions of relationship in a single 2-dimensional chart, coherently.

- So, we will present a few charts in sequence. The objects in each chart will be the same, but each chart will prioritise a different lens / projection of the object hypercube.

2025-05-14 - jerng

Lens 0 : Object Fundamentals

- Do review this graph of JavaScript's internal patterns.
- "Everything is an object."
- %Object.prototype% is the "god object", having a null prototype.



Lens 1 : Practical View of Generators

- Refer to the general documentation at W3Schools and MDN.
- The **numbering** below roughly follows **chronological** concerns.
- The numbered objects are most superficial, which developers will interact with directly.
  - The other objects are “implementation details”, which are commonly ignored.

1. We can’t make Generators directly. We must first create GeneratorFunction objects, using the syntax : `a = function * () {}`
   a. This example is a trivial function body : again, refer to general documentation for more thorough examples in practice.

2. Upon creation of a GeneratorFunction, `a`, the constructor has actually made a second separate **new** object. You will find it at `a.prototype`, it is a “%GeneratorPrototype Instance%, based on the given function body.

3. The GeneratorFunction must be called/applied, as a function, whereby it will then return Generator objects.
   a. The use of Generator objects is beyond the scope of this deck; please refer to the general documentation at W3Schools and MDN.

4. Should you call helper functions such as Generator.map(), you will receive a “Iterator Helper Object”, which behaves as a simplified Generator.

1

%GeneratorFunction% : the GeneratorFunction constructor
Also see : %AsyncGeneratorFunction%

- new %GeneratorFunction% => %GeneratorFunction%.[[Construct]]
- %GeneratorFunction%() => %GeneratorFunction%.[[Construct]]
- function * () {} => %GeneratorFunction%.[[Construct]]

- %GeneratorFunction%.[[Prototype]] => %Function%
- %GeneratorFunction%.prototype => %GeneratorFunction.prototype%
- %GeneratorFunction%.constructor => %Function%

2

“GeneratorFunction Object” :
Also see : “AsyncGeneratorFunction Instance”

Where %GeneratorFunction% constructs a **new** “GeneratorFunction Object” :
- “GeneratorFunction Object”.[[Prototype]] => %GeneratorFunction.prototype% (normal)
- “GeneratorFunction Object”.prototype => a **new** “%GeneratorPrototype% Instance” (weird)
- “GeneratorFunction Object”.constructor => %GeneratorFunction% (normal)

Subsequently :
- “GeneratorFunction Object”() => %GeneratorFunction.prototype%.[[Construct]]
- (function * () {})() => %GeneratorFunction.prototype%.[[Construct]]
  constructs **new** “Generator Objects”
  based on “GeneratorFunction Object”.prototype

2A

%GeneratorFunction.prototype% : is **not** a function object
Also see : %AsyncGeneratorFunction.prototype%

- %GeneratorFunction.prototype%.[[Prototype]] => %Function.prototype%
- %GeneratorFunction.prototype%.prototype => %Generator.prototype%
- %GeneratorFunction.prototype%.constructor => %GeneratorFunction%
- %GeneratorFunction.prototype%[%Symbol.toStringTag%] => “GeneratorFunction”

2B

“%GeneratorPrototype% Instance”
Also see : “%AsyncGeneratorPrototype% Instance”

- “%GeneratorPrototype%$ Instance”.[[Prototype]] => %Generator.prototype%
- “%GeneratorPrototype%$ Instance”.prototype => undefined
- “%GeneratorPrototype%$ Instance”.constructor => %GeneratorFunction.prototype%-

%GeneratorPrototype% : used by %GeneratorFunction.prototype%
Not an instance of %Function%, not a constructor.
Also see : %AsyncGeneratorPrototype%

- %GeneratorPrototype%.[[Prototype]] => %Iterator.prototype%
- %GeneratorPrototype%.prototype => undefined
- %GeneratorPrototype%.constructor => %GeneratorFunction.prototype%
- %GeneratorPrototype%[Symbol.toStringTag] => “Generator”
- %GeneratorPrototype%.next()
- %GeneratorPrototype%.return()
- %GeneratorPrototype%.throw()

%Iterator% : the Iterator constructor : an abstract superclass

- new %Iterator% => error
- %Iterator%() => error

- %Iterator%.[[Prototype]] => %Function.prototype%
- %Iterator%.prototype => %Iterator.prototype%
- %Iterator%.constructor => %Function%

%Iterator.prototype%
Also see : %AsyncIteratorPrototype%

- %Iterator.prototype%.[[Prototype]] => %Object.prototype%
- %Iterator.prototype%.prototype => undefined
- %Iterator.prototype%.constructor => %Iterator%

3

“Generator Object”
Also see : “AsyncGenerator Instance”

- “Generator Object”.[[Prototype]] => “%GeneratorPrototype% Instance”
- “Generator Object”.prototype => undefined
- “Generator Object”.constructor => %GeneratorFunction.prototype%

There is no pointer back to the sibling “GeneratorFunction Object”

4

“Iterator Helper Object” : e.g. returned by Iterator.map()

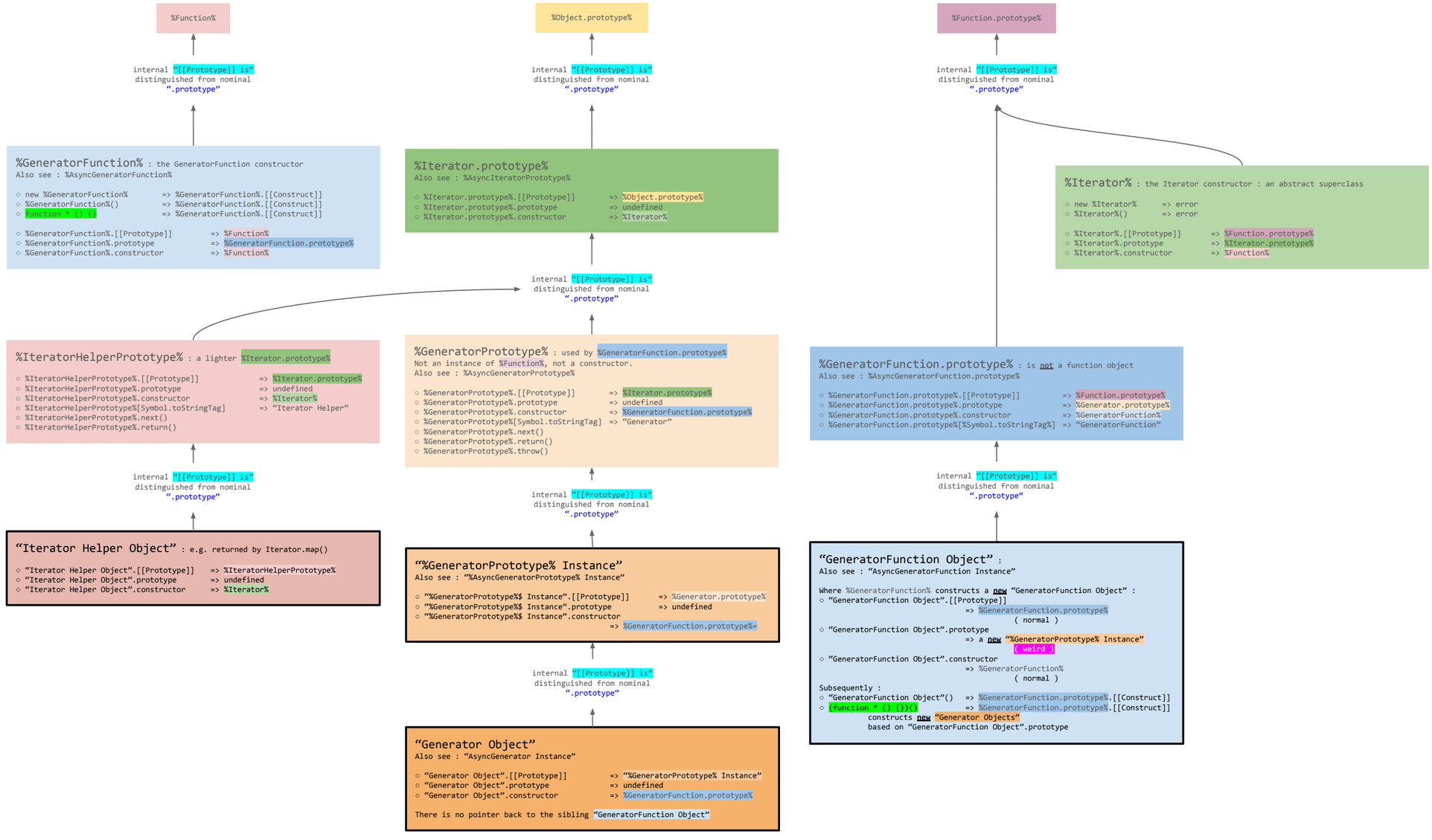
- “Iterator Helper Object”.[[Prototype]] => %IteratorHelperPrototype%
- “Iterator Helper Object”.prototype => undefined
- “Iterator Helper Object”.constructor => %Iterator%

%IteratorHelperPrototype% : a lighter %Iterator.prototype%

- %IteratorHelperPrototype%.[[Prototype]] => %Iterator.prototype%
- %IteratorHelperPrototype%.prototype => undefined
- %IteratorHelperPrototype%.constructor => %Iterator%
- %IteratorHelperPrototype%[Symbol.toStringTag] => “Iterator Helper”
- %IteratorHelperPrototype%.next()
- %IteratorHelperPrototype%.return()

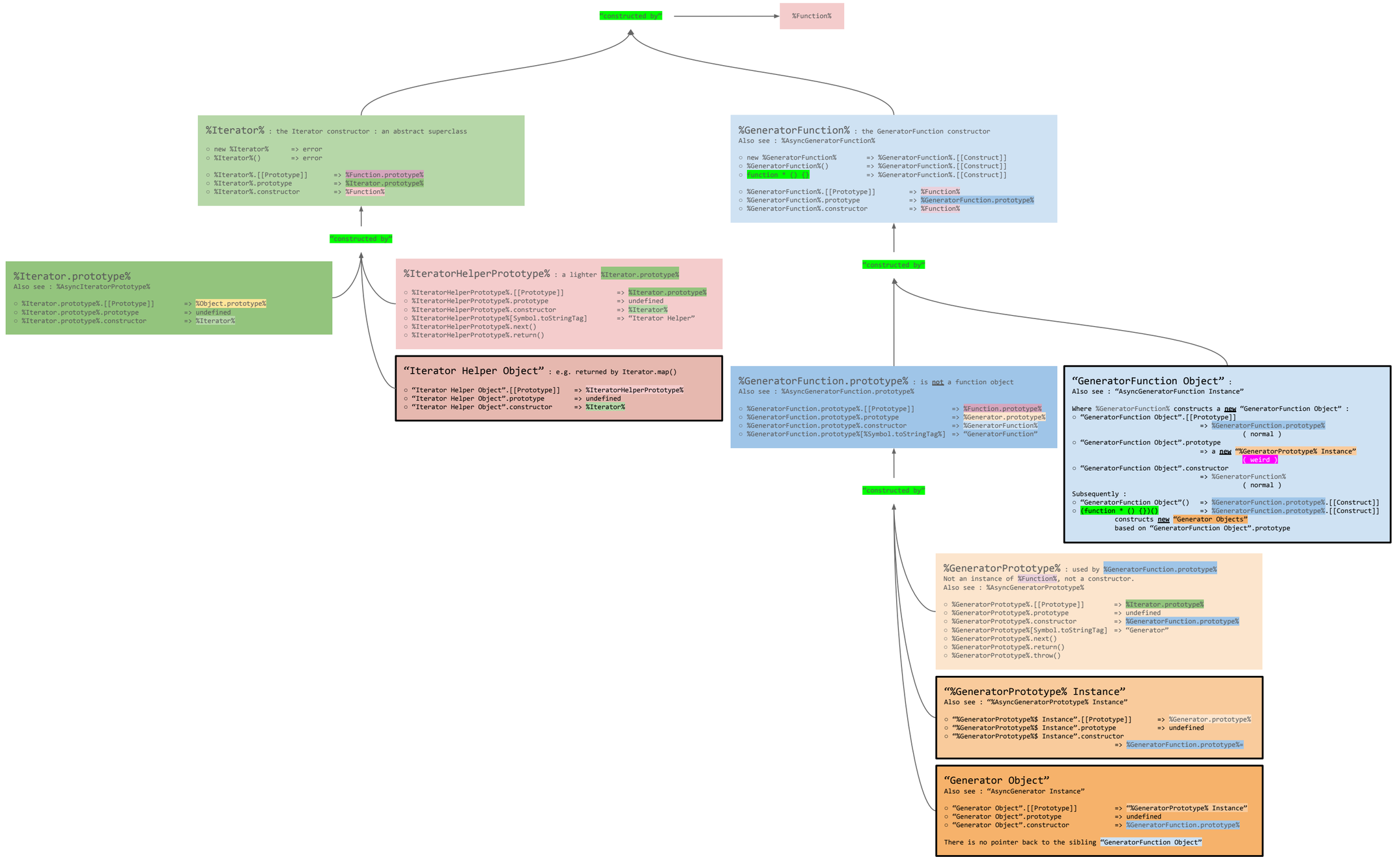
Lens 2 : [[Prototype]] chain

- Refer to the documentation for Object.getPrototypeOf(object)
- Note that this function may return a different value than what is found at object.prototype



Lens 3 : .constructor chain

- This seems rather unreliable as a source of meaning.
- But there is a semblance of chrono / logical order here.



Lens 4 : class hierarchy

- This seems rather unreliable as a source of meaning.
- But there is a semblance of chrono / logical order here.

%Function%

"instanceof"

%Object.prototype%

Everything else → "instanceof"

%GeneratorFunction% : the GeneratorFunction constructor

Also see : %AsyncGeneratorFunction%

- new %GeneratorFunction% => %GeneratorFunction%.[[Construct]]
- %GeneratorFunction%() => %GeneratorFunction%.[[Construct]]
- function * () {} => %GeneratorFunction%.[[Construct]]

- %GeneratorFunction%.[[Prototype]] => %Function%
- %GeneratorFunction%.prototype => %GeneratorFunction.prototype%
- %GeneratorFunction%.constructor => %Function%

%GeneratorFunction.prototype% : is not a function object

Also see : %AsyncGeneratorFunction.prototype%

- %GeneratorFunction.prototype%.[[Prototype]] => %Function.prototype%
- %GeneratorFunction.prototype%.prototype => %Generator.prototype%
- %GeneratorFunction.prototype%.constructor => %GeneratorFunction%
- %GeneratorFunction.prototype%[%Symbol.toStringTag%] => "GeneratorFunction"

"GeneratorFunction Object" :

Also see : "AsyncGeneratorFunction Instance"

Where %GeneratorFunction% constructs a new "GeneratorFunction Object" :

- "GeneratorFunction Object".[[Prototype]] => %GeneratorFunction.prototype% (normal)
- "GeneratorFunction Object".prototype => a new "%GeneratorPrototype% Instance" (weird)
- "GeneratorFunction Object".constructor => %GeneratorFunction% (normal)

Subsequently :

- "GeneratorFunction Object"() => %GeneratorFunction.prototype%.[[Construct]]
- (function * () {})() => %GeneratorFunction.prototype%.[[Construct]]

constructs new "Generator Objects" based on "GeneratorFunction Object".prototype

%Iterator% : the Iterator constructor : an abstract superclass

- new %Iterator% => error
- %Iterator%() => error

- %Iterator%.[[Prototype]] => %Function.prototype%
- %Iterator%.prototype => %Iterator.prototype%
- %Iterator%.constructor => %Function%

%Iterator.prototype%

Also see : %AsyncIteratorPrototype%

- %Iterator.prototype%.[[Prototype]] => %Object.prototype%
- %Iterator.prototype%.prototype => undefined
- %Iterator.prototype%.constructor => %Iterator%

"instanceof"

%IteratorHelperPrototype% : a lighter %Iterator.prototype%

- %IteratorHelperPrototype%.[[Prototype]] => %Iterator.prototype%
- %IteratorHelperPrototype%.prototype => undefined
- %IteratorHelperPrototype%.constructor => %Iterator%
- %IteratorHelperPrototype%[Symbol.toStringTag] => "Iterator Helper"
- %IteratorHelperPrototype%.next()
- %IteratorHelperPrototype%.return()

"Iterator Helper Object" : e.g. returned by Iterator.map()

- "Iterator Helper Object".[[Prototype]] => %IteratorHelperPrototype%
- "Iterator Helper Object".prototype => undefined
- "Iterator Helper Object".constructor => %Iterator%

%GeneratorPrototype% : used by %GeneratorFunction.prototype%

Not an instance of %Function%, not a constructor.

Also see : %AsyncGeneratorPrototype%

- %GeneratorPrototype%.[[Prototype]] => %Iterator.prototype%
- %GeneratorPrototype%.prototype => undefined
- %GeneratorPrototype%.constructor => %GeneratorFunction.prototype%
- %GeneratorPrototype%[Symbol.toStringTag] => "Generator"
- %GeneratorPrototype%.next()
- %GeneratorPrototype%.return()
- %GeneratorPrototype%.throw()

"%GeneratorPrototype% Instance"

Also see : "%AsyncGeneratorPrototype% Instance"

- "%GeneratorPrototype%$ Instance".[[Prototype]] => %Generator.prototype%
- "%GeneratorPrototype%$ Instance".prototype => undefined
- "%GeneratorPrototype%$ Instance".constructor => %GeneratorFunction.prototype%=

"Generator Object"

Also see : "AsyncGenerator Instance"

- "Generator Object".[[Prototype]] => "%GeneratorPrototype% Instance"
- "Generator Object".prototype => undefined
- "Generator Object".constructor => %GeneratorFunction.prototype%

There is no pointer back to the sibling "GeneratorFunction Object"

Lens 5 : hypercube : messy

- This seems rather unreliable as a source of meaning.
- But there is a semblance of chrono / logical order here.

"constructed by"

%Function%

%Function.prototype%

%Object.prototype%

internal "[[Prototype]] is" distinguished from nominal ".prototype"

internal "[[Prototype]] is" distinguished from nominal ".prototype"

internal "[[Prototype]] is" distinguished from nominal ".prototype"

%GeneratorFunction% : the GeneratorFunction constructor
Also see : %AsyncGeneratorFunction%

- new %GeneratorFunction% => %GeneratorFunction%.[[Construct]]
- %GeneratorFunction%() => %GeneratorFunction%.[[Construct]]
- function * () {} => %GeneratorFunction%.[[Construct]]

- %GeneratorFunction%.[[Prototype]] => %Function%
- %GeneratorFunction%.prototype => %GeneratorFunction.prototype%
- %GeneratorFunction%.constructor => %Function%

"constructed by"

%Iterator% : the Iterator constructor : an abstract superclass

- new %Iterator% => error
- %Iterator%() => error

- %Iterator%.[[Prototype]] => %Function.prototype%
- %Iterator%.prototype => %Iterator.prototype%
- %Iterator%.constructor => %Function%

%Iterator.prototype%
Also see : %AsyncIteratorPrototype%

- %Iterator.prototype%.[[Prototype]] => %Object.prototype%
- %Iterator.prototype%.prototype => undefined
- %Iterator.prototype%.constructor => %Iterator%

%GeneratorFunction.prototype% : is **not** a function object
Also see : %AsyncGeneratorFunction.prototype%

- %GeneratorFunction.prototype%.[[Prototype]] => %Function.prototype%
- %GeneratorFunction.prototype%.prototype => %Generator.prototype%
- %GeneratorFunction.prototype%.constructor => %GeneratorFunction%
- %GeneratorFunction.prototype%[%Symbol.toStringTag%] => "GeneratorFunction"

"constructed by"

"instanceof"

"constructed by"

"constructed by"

internal "[[Prototype]] is" distinguished from nominal ".prototype"

internal "[[Prototype]] is" distinguished from nominal ".prototype"

internal "[[Prototype]] is" distinguished from nominal ".prototype"

%GeneratorPrototype% : used by %GeneratorFunction.prototype%
Not an instance of %Function%, not a constructor.
Also see : %AsyncGeneratorPrototype%

- %GeneratorPrototype%.[[Prototype]] => %Iterator.prototype%
- %GeneratorPrototype%.prototype => undefined
- %GeneratorPrototype%.constructor => %GeneratorFunction.prototype%
- %GeneratorPrototype%[Symbol.toStringTag] => "Generator"
- %GeneratorPrototype%.next()
- %GeneratorPrototype%.return()
- %GeneratorPrototype%.throw()

%IteratorHelperPrototype% : a lighter %Iterator.prototype%

- %IteratorHelperPrototype%.[[Prototype]] => %Iterator.prototype%
- %IteratorHelperPrototype%.prototype => undefined
- %IteratorHelperPrototype%.constructor => %Iterator%
- %IteratorHelperPrototype%[Symbol.toStringTag] => "Iterator Helper"
- %IteratorHelperPrototype%.next()
- %IteratorHelperPrototype%.return()

internal "[[Prototype]] is" distinguished from nominal ".prototype"

internal "[[Prototype]] is" distinguished from nominal ".prototype"

"%GeneratorPrototype% Instance"
Also see : "%AsyncGeneratorPrototype% Instance"

- "%GeneratorPrototype%$ Instance".[[Prototype]] => %Generator.prototype%
- "%GeneratorPrototype%$ Instance".prototype => undefined
- "%GeneratorPrototype%$ Instance".constructor => %GeneratorFunction.prototype%-

"GeneratorFunction Object" :
Also see : "AsyncGeneratorFunction Instance"

Where %GeneratorFunction% constructs a **new** "GeneratorFunction Object" :
- "GeneratorFunction Object".[[Prototype]] => %GeneratorFunction.prototype% (normal)
- "GeneratorFunction Object".prototype => a **new** "%GeneratorPrototype% Instance" (weird)
- "GeneratorFunction Object".constructor => %GeneratorFunction% (normal)

Subsequently :
- "GeneratorFunction Object"() => %GeneratorFunction.prototype%.[[Construct]]
- (function * () {})() => %GeneratorFunction.prototype%.[[Construct]]

constructs **new** "Generator Objects" based on "GeneratorFunction Object".prototype

internal "[[Prototype]] is" distinguished from nominal ".prototype"

"Iterator Helper Object" : e.g. returned by Iterator.map()

- "Iterator Helper Object".[[Prototype]] => %IteratorHelperPrototype%
- "Iterator Helper Object".prototype => undefined
- "Iterator Helper Object".constructor => %Iterator%

"Generator Object"
Also see : "AsyncGenerator Instance"

- "Generator Object".[[Prototype]] => "%GeneratorPrototype% Instance"
- "Generator Object".prototype => undefined
- "Generator Object".constructor => %GeneratorFunction.prototype%

There is no pointer back to the sibling "GeneratorFunction Object"