Globalization

In software development, globalization is commonly understood to be the combination of internationalization and localization.

The Globalization Object

The Globalization object is a single object that has some named properties, all of which are constructors.

The value of the [[Prototype]] internal property of the Globalization object is the built-in Object prototype object specified by the ECMAScript Language Specification. The value of the [[Extensible]] internal property is false.

NOTE The [[Extensible]] internal property is set to false for compatibility with the future module system in the ECMAScript Language Specification, 6 edition.

The Globalization object does not have a [[Construct]] internal property; it is not possible to use the Globalization object as a constructor with the new operator.

The Globalization object does not have a [[Call]] internal property; it is not possible to invoke the Globalization object as a function.

Constructor Properties of the Globalization Object

Each of the properties of the Globalization object is a constructor. The common behavior of these constructors is specified in this section; all remaining aspects are specified in the following clauses: LocaleList, Collator, NumberFormat, and DateTimeFormat.

Properties of the Constructors and Their Prototypes

When Globalization.LocaleList is called with a this value that is not an object whose constructor property is Globalization.LocaleList itself, it creates and initializes a new LocaleList object. Thus the function call Globalization.LocaleList(...) is equivalent to the object creation expression new Globalization.LocaleList(...) with the same arguments.

The LocaleList Constructor

When Globalization.LocaleList is called with a this value that is an object whose constructor property is Globalization.LocaleList itself, it acts as a constructor: it initializes the object.
new Globalization.LocaleList (locales)

When the LocaleList constructor is called with one argument, it interprets the locales argument as an array and copies its elements into the newly constructed object, validating the elements as well-formed language tags using the abstract operation IsWellFormedLanguageTag (6.2.1), and omitting duplicates.

Let *obj* be the **this** value.

Let *seen* be the result of creating a new object as if by the expression `new Object()` where `Object` is the standard built-in constructor with that name.

Let *cb* be a function that takes

`seen` with argument `tag`.
If *duplicate* is true, then return.
Call the `[[Put]]` internal method of `seen` `tag`, **true**, and **true**.

Let *desc* be the result of creating a new object as if by the expression `new Object()` where `Object` is the standard built-in constructor with that name.
Call the `[[Put]]` internal method of `desc` with the arguments "value", `tag`, and **true**.
Call the `[[Put]]` internal method of `desc` with the arguments "enumerable", **true**, and **true**.
Call the `Object.defineProperty` function with arguments `obj`,

new Globalization.LocaleList ()

When the LocaleList constructor is called with no argument, it behaves as if it had received the array [locale] as the first argument, where locale is the value of the `[currentHostLocale`

In the following descriptions of functions that are properties of the LocaleList prototype object, the phrase “this LocaleList object” refers to the object that is the this value for the invocation of the function.

Globalization

1.1.1
The [[IndexOfMatchFor]] internal method compares the provided argument `locale`, which it expects to be a String value with a well-formed and canonicalized BCP 47 language tag, against the locales in this locale list and returns the index of the best available match.

It uses the fallback mechanism of RFC 4647, section 3.4. The following steps are taken:

```
Repeat while the value of the length property of `locale` is greater than 0:
    Let `index` be the result of applying the Array.prototype.indexOf method to this LocaleList object with the argument list `[locale]`.
    If `index` does not equal -1, then return `index`.
    Let `pos` be the result of calling the lastIndexOf method of `locale` with the argument "-".
    If `pos` does not equal -1, then
        If `pos` is greater or equal to 2 and the character at index `pos`-2 of `locale` equals "-",
        then decrease `pos` by 2.
        Let `locale` be the result of calling the substring method of `locale` with arguments 0 and `pos`.
    Else let `locale` be "".
    Return -1.
```

The [[LookupMatch]] internal method compares `requestedLocales`, which it expects to be a LocaleList object representing a BCP 47 language priority list, against the set of locales in this LocaleList object, and determines the best available language to meet the request. The algorithm is based on the Lookup algorithm described in RFC 4647 section 3.4, but options specified through Unicode extension sequences are ignored in the lookup.

Information about such subsequences is returned separately.
Let `extensionMatch` be null.
Let `i` be 0.

Repeat while `i` is less than the value of the length property of `requestedLocales` and `availableIndex` is -1:
- Let `locale` be the element at index `i` of `requestedLocales`.
- Let `extensionMatch` be the result of calling the match method of `locale` with the argument `/-u(-([a-z0-9]{2,8}))+/`.
  - If `extensionMatch` is not null, then:
    - Let `extension` be the value of the element at index 0 of `extensionMatch`.
    - Let `extensionIndex` be the value of the index property of `extensionMatch`.
    - Let `locale` be the result of calling the replace method of `locale` with arguments `extension` and "".
    - Let `availableIndex` be the result of calling the `[[IndexOfMatchFor]]` method of this LocaleList object, passing `locale` as the argument.
- If `availableIndex` does not equal -1, then:
  - Call the `[[Put]]` internal method of `result` with the arguments "locale", the element at index `availableIndex` of this LocaleList object, and true.
  - If `extensionMatch` is not null, then:
    - Call the `[[Put]]` internal method of `result` with the arguments "extension", `extension`, and true.

Call the `[[Put]]` internal method of `result` with the arguments "extensionIndex", `extensionIndex`, and true

Call the `[[Put]]` internal method of `result` with the arguments "locale", the value of the `[[currentHostLocale]]` internal property of the Globalization object, and true.

Return `result`.

`[[BestFitMatch]]` (`requestedLocales`) 1.5

The `[[BestFitMatch]]` internal method compares `requestedLocales`, which it expects to be a LocaleList object representing a BCP 47 language priority list, against the set of locales in this LocaleList object, and determines the best available language to meet the request. The algorithm is implementation dependent, but should produce results that a typical user of the requested locales would perceive as at least as good as those produced by the `[[LookupMatch]]` internal method. Options specified through Unicode

The internal method returns an object with a locale property, whose value is the language tag of the selected locale, which must be an element of this LocaleList object. If the language tag of the request locale that led to the selected locale contained a Unicode extension subsequence, then the returned object also contains an extension property whose value is the Unicode extension
subsequence (starting with ",-u"), and an extensionIndex property whose value is the index of the Unicode extension subsequence within the request locale language tag.

[[LookupSupportedLocalesOf]] (requestedLocales)

The [[LookupSupportedLocalesOf]] internal method returns the subset of the provided BCP 47 language priority list for which this LocaleList object has a matching locale when using the BCP 47 Lookup algorithm. Locales appear in the same order in the returned list as in the input list.

The following steps are taken:

1. If the constructor of requestedLocales is not Globalization.LocaleList, then replace requestedLocales with a new LocaleList object as if by the expression new Globalization.LocaleList(requestedLocales), where Globalization.LocaleList is the standard built-in constructor with that name.

2. Let callback be a function that takes the argument locale and performs the following steps:
   - Let locale be the result of calling the replace method of locale with the arguments /-u(-([a-zA-Z0-9]{2,8})+)/ and ".".
   - Let index be the result of calling the [[IndexOfMatchFor]] internal method of this LocaleList object, passing locale as the argument.
   - If index does not equal -1, then return true, otherwise return false.

3. Let subset be the result of applying the Array.prototype.filter method to requestedLocales, passing the argument list [callback, this].

4. Return the result of creating a new object as if by the expression new Globalization.LocaleList(subset), where Globalization.LocaleList is the standard built-in constructor with that name.

[[BestFitSupportedLocalesOf]] (requestedLocales)

The [[BestFitSupportedLocalesOf]] internal method returns the subset of the provided BCP 47 language priority list for which this LocaleList object has a matching locale when using the Best Fit Match algorithm. Locales appear in the same order in the returned list as in the input list.

The steps taken are implementation dependent.

[[SupportedLocalesOf]] (requestedLocales, options)

The [[SupportedLocalesOf]] internal method returns the subset of the provided BCP 47 language priority list for which this LocaleList object has a matching locale when using the supported BCP 47 algorithm. Locales appear in the same order in the returned list as in the input list.
Two algorithms are available to match the locales: the Lookup algorithm described in RFC 4647 section 3.4, and an implementation dependent best-fit algorithm. Locales appear in the same order in the returned list as in

The following steps are taken:

- If `options` is provided and not `undefined`, then
  - Let `matcher` be the result of calling the `[[Get]]` internal method of `options` with argument "localeMatcher".
  - If `matcher` is not `undefined`, then
    - Let `matcher` be `ToString(matcher)`.
    - If `matcher` is not equal to "lookup" or "best fit", then throw a `RangeError` exception.
  - If `matcher` is `undefined` or equals "best fit" then
    - Return the result of calling the `[[BestFitSupportedLocalesOf]]` internal method of this LocaleList object with argument `requestedLocales`.

- Return the result of calling the `[[LookupSupportedLocalesOf]]` internal method of this LocaleList object with argument `requestedLocales`.

A numeric value that is one greater than the name
If the element at index \( i \) of \( \text{extensionSubtags} \) does not equal less the element at index \( keyPos + 1 \) of \( \text{extensionSubtags} \).
When Globalization.

1.10.1

Thus the function call `Globalization.Collator(...)` is equivalent to the object creation expression `new Globalization.Collator(...)` with the same arguments.

**The Collator Constructor**

When `Globalization.Collator` is called with a this value that is an object whose constructor property is `Globalization.Collator` itself, it acts as a constructor: it initializes the object.

```
new Globalization.Collator ([localeList [, options]])
```

When the Collator constructor is called with two arguments, it computes its effective locale and its collation options from these arguments.

The compare method itself is not directly suitable as an argument to `Array.prototype.sort` because it must be invoked as the method of a Collator object.

NOTE 2

Thus the function call `Globalization.NumberFormat(...)` is equivalent to the object creation expression `new Globalization.NumberFormat(...)` with the same arguments.

**The NumberFormat Constructor**

When `Globalization.NumberFormat` is called with a this value that is an object whose constructor property is `Globalization.NumberFormat` itself, it acts as a constructor: it initializes the object.

```
new Globalization.NumberFormat ([localeList [, options]])
```

When the NumberFormat constructor is called with arguments `localeList` and `options`, it computes its effective locale and its formatting options from these arguments.

Function

When Globalization.

1.10.2
Thus the function call `Globalization.DateTimeFormat(...)` is equivalent to the object creation expression `new Globalization.DateTimeFormat(...)` with the same arguments.

**The DateTimeFormat Constructor**

When `Globalization.DateTimeFormat` is called with a `this` value that is an object whose constructor property is `Globalization.DateTimeFormat` itself, it acts as a constructor: it initializes the object.

```javascript
new Globalization.DateTimeFormat ([localeList [, options]])
```

When the `DateTimeFormat` constructor is called with arguments `localeList` and `options`, it computes its effective locale and its formatting options from these arguments.