"DISCLAIMER

english (uk)

english (uk)

english (uk)

english (uk)

english (uk)

that comment on or otherwise explain it or assist in its implementation

english (uk)

english (uk)

english (uk)

english (uk)

M4, No widow/orphan control
LocaleList Objects

LocaleList objects represent lists of language tags identifying locales. They can be used in two ways:

To represent a language priority list, as described in RFC 4647, section 2.3, or successor. Algorithms interpreting a LocaleList object in this sense treat the list as ordered in descending order of priority.

To represent a set of locales, such as those supported by an application or by the implementation of an object described in this specification. Algorithms interpreting a LocaleList object in this sense treat the list as unordered.

LocaleList objects have the properties of a generic array-like object: A length property and other properties whose names are array indices, as defined in ES5, 15.4. The value of the length property is numerically greater than the name of every property inserted during construction whose name is an array index. As LocaleList objects are extensible, this invariant is not guaranteed to be maintained after construction.

The LocaleList constructor is a property of the Intl object.

The Intl.LocaleList Constructor

Initializing an Object as a LocaleList

The abstract operation InitializeLocaleList accepts the argument `localeList`, which must be an object, and the optional argument `locales`. It initializes `localeList` as a LocaleList object by taking the following steps:

If `locales` is not provided or is `undefined`, then

Let `seen` be a new List containing the String returned by the DefaultLocale abstract operation.

Else

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC 5646, Tags for Identifying Languages, or successor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RFC 4647, Matching of Language Tags, or successor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>IETF RFC 6067, BCP 47 Extension U, or successor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
If the result of calling the abstract operation `IsStructurallyValidLanguageTag`, passing `tag` as the argument, is `false`, then throw a `RangeError` exception.

Let `tag` be the result of calling the abstract operation `CanonicalizeLanguageTag`, passing `tag` as the argument.

---

**NOTE**

Non-normative summary: The function interprets the `locales` argument as an array and copies its elements into `localeList`, validating the elements as structurally valid language tags and canonicalizing them, and omitting duplicates. It tags this object as a non-initialized locale list.

**NOTE**

If an object has been previously initialized using the Intl.LocaleList constructor, and the canonicalization of the previously used `locales` differs from the canonicalization of the `locales` used now, then either step 4.a or step 5 will fail.

---

### The Intl.LocaleList Constructor Called as a Function

When Intl.LocaleList is called as a function rather than as a constructor, it accepts the optional argument `locales` and takes the following steps:

If `this` is the Intl object or `undefined`, then

- Return the result of creating a new object as if by the expression new
  `Intl.LocaleList(locales)`, where `Intl.LocaleList` is the standard built-in constructor defined in 9.1.3.

---

**NOTE**

Non-normative summary: The function interprets the `locales` argument as an array and copies its elements into `localeList`, validating the elements as structurally valid language tags and canonicalizing them, and omitting duplicates. It tags this object as a non-initialized locale list.

**NOTE**

If an object has been previously initialized using the Intl.LocaleList constructor, and the canonicalization of the previously used `locales` differs from the canonicalization of the `locales` used now, then either step 4.a or step 5 will fail.

---

### The Intl.LocaleList Constructor Used in a new Expression

When `Intl.LocaleList` is called as part of a `new` expression, it is a constructor. It accepts the optional argument `locales`, and initializes the properties of the newly constructed object by calling the `InitializeLocaleList` abstract operation (9.1.1), passing the newly constructed object and `locales` as arguments.

The `[[Prototype]]` internal property of the newly constructed object is set to the original Intl.LocaleList prototype object, the one that is the value of Intl.LocaleList.prototype (9.2.1).

---

### Properties of the Intl.LocaleList Constructor

Besides the internal properties and the length property (whose value is 1), the Intl.LocaleList constructor has the following properties:
Intl.LocaleList.prototype

The value of Intl.LocaleList.prototype is the built-in Intl.LocaleList prototype object (9.3).

Properties of the Intl.LocaleList Prototype Object

The Intl.LocaleList prototype object is itself an Intl.LocaleList instance as specified in 9.4, whose properties are set as if it had been constructed by the expression `new Intl.LocaleList([])`.

Intl.LocaleList.prototype.constructor

The initial value of Intl.LocaleList.prototype.constructor is the built-in Intl.LocaleList constructor.

Properties of Intl.LocaleList Instances

Intl.LocaleList instances inherit properties from the Intl.LocaleList prototype object.

Intl.LocaleList instances and other objects that have been successfully initialized as a LocaleList have an `[[initializedLocaleList]]` internal property whose value is `true`.

Intl.LocaleList instances and other objects that have been successfully initialized as a LocaleList also have the following properties.

length

The length property of an Intl.LocaleList object is a data property whose value is the number of array index properties added to the object by the InitializeLocaleList abstract operation.

The length property has the attributes `{ [[Writable]]: false, [[Enumerable]]: false, [[Configurable]]: false }.

Properties With Array Index Names

A LocaleList object has properties whose names are array indices from 0 to (length - 1). The value of each of these properties is a String value representing a structurally valid language tag. The values are unique within a LocaleList object.

These properties initially have the attributes `{ [[Writable]]: false, [[Enumerable]]: true, [[Configurable]]: false }.

If `index` ≠ -1

availableIndex ≠ -1, then

Let

If `requestedLocales` is `undefined`, then

Let `requestedLocales` be the result of creating a new LocaleList object as if by the expression `new Intl.LocaleList()` where `Intl.LocaleList` is the standard built-in constructor defined in 9.1.3.
Else
Let `requestedLocales` be `ToObject(requestedLocales)`.

If `requestedLocales` does not have an `[[initializedLocaleList]]` internal property with value `true`, then

Let `requestedLocales` be the result of creating a new `LocaleList` object as if by the expression `new Intl.LocaleList(requestedLocales)`, where

`Intl.LocaleList` is the standard built-in constructor defined in 9.1.3.

Algorithm variable, English (UK)

When the `GetGetOption` abstract operation is called with argument `options`, the following steps are taken.
Let `getOption` be a function which, when called with arguments `property`, `type`, `values`, and `fallback`, takes the following steps:

Return `getOption`.

### GetGetNumberOption

When the `GetGetNumberOption` abstract operation is called with argument `options`, the following steps are taken:

Let `getNumberOption` be a function which, when called with arguments `property`, `minimum`, `maximum`, and `fallback`, takes the following steps:
1.2 The Intl.Collator Constructor

Let `getOption` be the result of calling the `GetGetOption` abstract operation with argument `options`.

If $u$
If `u` is defined:

```javascript
getOption()
```

properties of the newly constructed object by calling the InitializeCollator abstract operation (11.1.1), passing the newly constructed object, `localeList`, and `options` as arguments.

The `[[Extensible]]` internal property of the newly constructed object is set to `true`.

If this Collator does not have an `[[initializedCollator]]` internal property with value `true`, then throw a `TypeError` exception.

If the `[[boundCompare]]` internal property of this Collator is `undefined`, then:

Let `that` be `this`.
Let \( bc \) be a function that takes the arguments \( x \) and \( y \) and performs the following steps:

- Return the result of calling the Compare abstract operation with arguments \( that, x, \) and \( y \).
- Set the \([\text{boundCompare}]\) internal property of this Collator to \( bc \).
- Return the value of the \([\text{boundCompare}]\) internal property of this Collator.

The \([\text{Set}]\) attribute is undefined.
The Intl.NumberFormat Constructor

Let \textit{getOption} be the result of calling the \textit{GetGetOption} abstract operation with argument \textit{options}.
Let $\text{getNumberOption}$ be the result of calling the GetGetNumberOption abstract operation with argument $\text{options}$.
If this NumberFormat object does not have an [[initializedNumberFormat]] internal property with value `true`, then throw a `TypeError` exception.

Let `x` be `ToNumber(value)`.

result of calling the `indexOf` method of `m` with argument `"."`.

If `period > 0`
Returns a String value representing the result of calling ToNumber(date)

1. does not have an [[initializedDateTimeFormat]] internal property with value true

2. does not have an [[initializedDateTimeFormat]] internal property with value true

omitted

omitted

omitted
Let \( \text{format} \) be the standard built-in function object defined in 12.3.2.
value of the [[Get]] attribute is a

an argument list containing the single item this.

References

ECMA-262, ECMAScript Language Specification
http://www.ecma-international.org/publications/standards/Ecma-262.htm

ISO 4217:2008, Codes for the representation of currencies and funds


The Unicode Standard
http://www.unicode.org/versions/latest

Unicode Technical Standard 10, Unicode Collation Algorithm
http://unicode.org/reports/tr10/
Unicode Technical Standard 35, Unicode Locale Data Markup Language
http://unicode.org/reports/tr35/

Unicode Common Locale Data Repository
http://cldr.unicode.org/

IETF BCP 47:
RFC 5646, Tags for Identifying Languages
RFC 4647, Matching of Language Tags

IETF RFC 6067, BCP 47 Extension U

IANA Time Zone Database
http://www.iana.org/time-zones