

## console.logによるログ出力

基本編のおまけでログ出力について紹介します。言語を学ぶときやデバッグ時には変数を出力することで状態を知ろうとするでしょう。その時に使えるのがConsole APIです。これらは特にTypeScript特有のものではありませんが、覚えておいて損はないので紹介します。

```
console.log("ログ出力");
```

これが一番使われるものだと思いますが、他にもいろいろあります。これを実装すべきといった仕様はないのですが、現状で各環境で実装されている仕様はLiving StandardとしてWHATWGにあります。しかし、どの環境でも使えるとは限りません。特に、TypeScript Playgroundで使えないものが多数ありますが、これらは何もしなだけでエラーになるわけではありません。

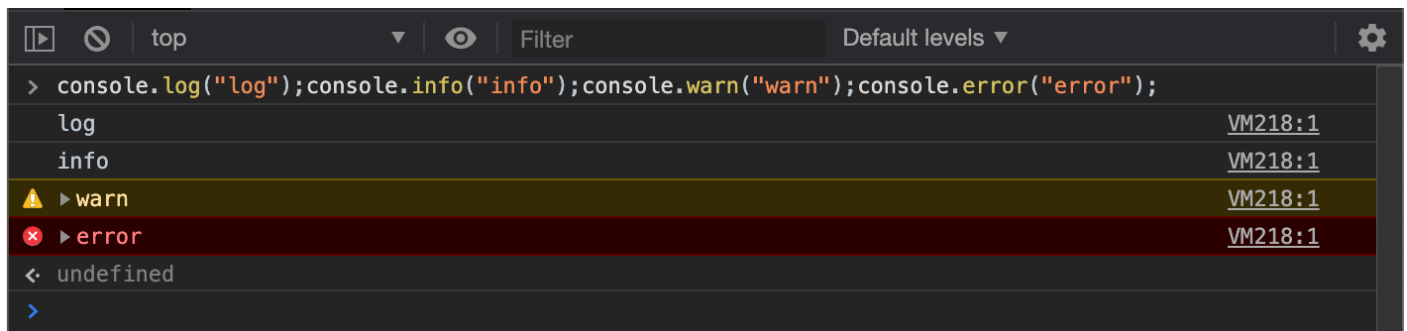
- <https://console.spec.whatwg.org/>

本章ではいくつかをピックアップして紹介します。

### console.log/info/warn/error()

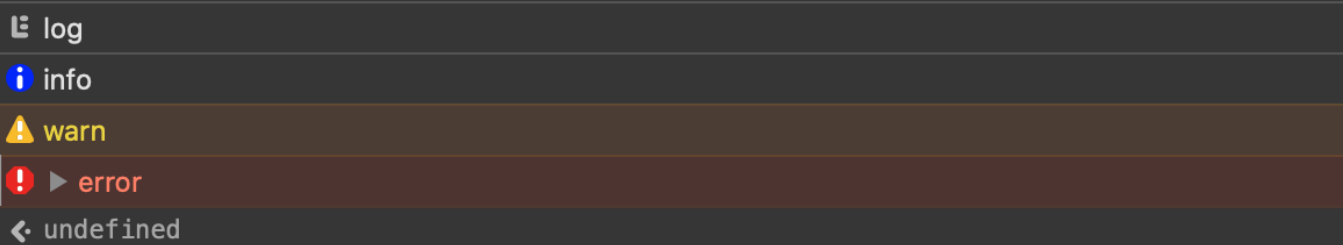
API	Chrome開発ツール	Safari開発ツール	Node.js	TS PG
<code>console.log()</code>	✓	✓	✓	✓
<code>console.info()</code>	<code>log()</code> と同じ	✓	<code>log()</code> と同じ	✓
<code>console.warn()</code>	✓	✓	<code>log()</code> と同じ	✓
<code>console.error()</code>	✓	✓	<code>log()</code> と同じ	✓

それぞれ、見た目が大きく異なります。



Chromeの開発ツールでの表示

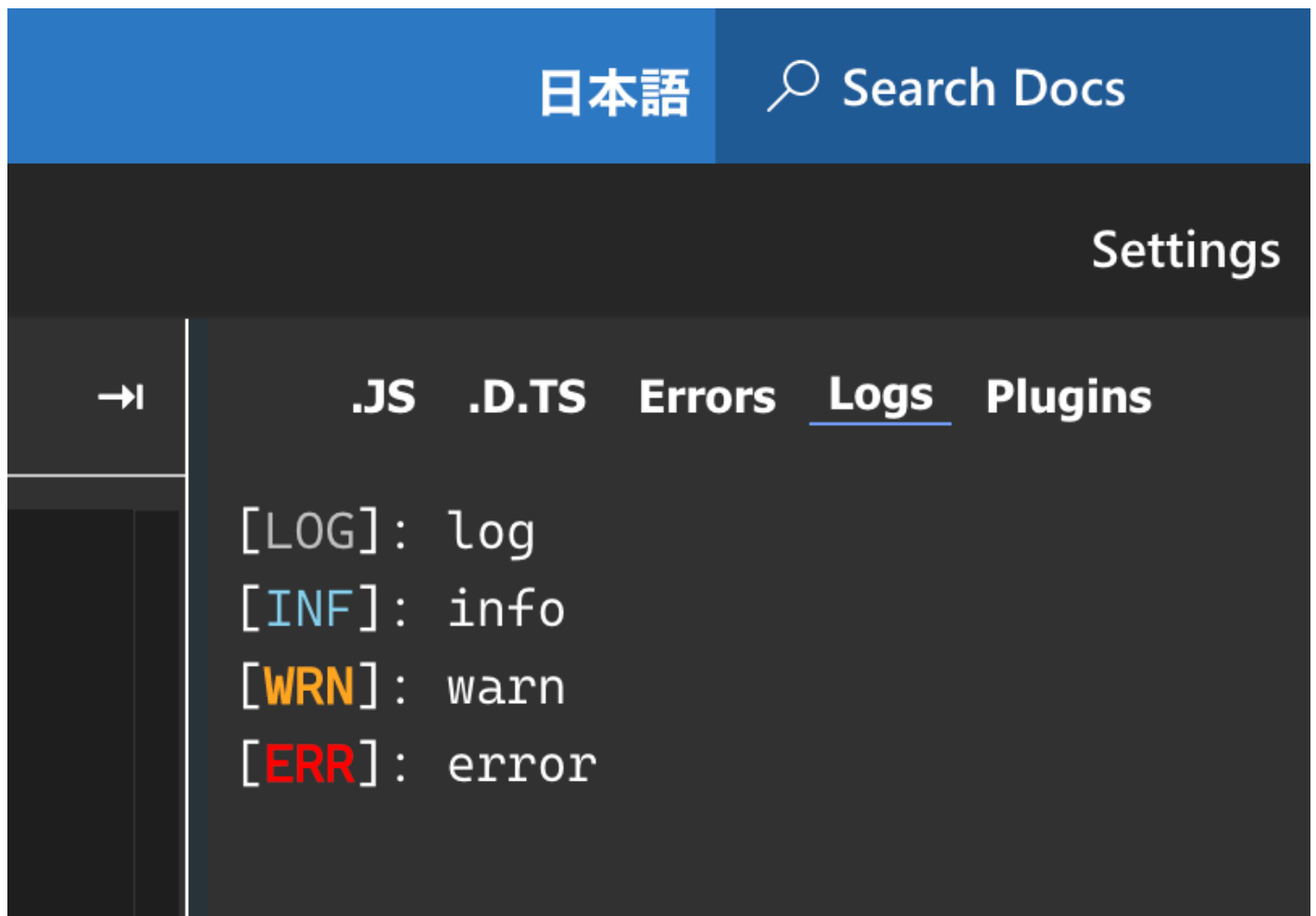
```
> console.log("log");console.info("info");console.warn("warn");console.error("error");
```



Safariの開発ツールでの表示

```
[> console.log("log");console.info("info");console.warn("warn");console.error("error");
log
info
warn
error
undefined
> ]
```

Node.jsでの表示



TypeScript Playgroundでの表示

どの環境でも使えるテクニックですが、何かしらの変数を出力したい場合、`{ }`でくくると、変数名と値がペアになって出力されます。これはオブジェクトの短縮記法によるもので、オブジェクトの中に変数だけを書くと、その変数と同じ名前のキーに、その変数の値が格納されます。オブジェクトはキーと値をセットで出力されるため、変数名と値が両方出力されます。

```
console.log({name});  
>>> { "name": "wozozo" }
```

注釈

2020年9月リリースのChrome 85から挙動が変わり、コンソールの左にあるログレベルを選択した場合、選択したものだけが表示されます。エラーがあった場合にもerrorを選択しなければ表示されません。Chrome 84以前、および2020年10月現在のSafariでは選択したログレベル以上のものが表示されますので、infoやwarnを選択した場合にもエラーが表示されます。

## console.table/dir()

API	Chrome開発ツール	Safari開発ツール	Node.js	TS PG
<code>console.table()</code>	✓	✓	✓	
<code>console.dir()</code>	✓	✓	<code>log()</code> と同じ	

複雑なオブジェクトを見やすく表示してくれる機能です。 `console.table()` は表形式で、 `console.dir()` はツリー形式でデータを表示します。

```
> a = {smile: "😊", cry: "😭", angry: "😡"};
< ▶ {smile: "😊", cry: "😭", angry: "😡"}
> console.table(a);
```

(index)	Value
smile	"😊"
cry	"😭"
angry	"😡"

```
▶ Object
< undefined
> console.dir(a);
```

▼ Object ⓘ

angry: "😡"

cry: "😭"

smile: "😊"

▶ \_\_proto\_\_: Object

```
< undefined
```

`console.table()` と `console.dir()`

ただし、この両方がフルでサポートされているのはブラウザの開発者ツールのみです。Node.js は `console.table()` はサポートしています。TypeScript Playgroundはエラーにはなりません、何も表示されません。

## その他のメソッド

他にもたくさん機能はありますが、特に使う必要がないと思われるものです。

`console.assert()` は、他の言語のライブラリや組み込み機能で提供されているのと同じく、条件式が開発者の想定通りかどうかを確認する関数です。最初の引数が `true` でなければエラーとしてログに出力されます。しかし、開発者ツールでもNode.jsでも、エラーが発生したことしかわかりません。実行時にエラーにするよりは、ユニットテストでカバーすべき内容でしょう。

```
console.assert(a === 1);
```

## まとめ

デバッグログで使える機能を紹介しました。いろいろ紹介しましたが、基本的にはテキスト出力がユーザーインタフェースの一部となっているNode.js以外は本番コードには残すべきものではありません。あくまでもデバッグの補助です。環境ごとに使えたり使えなかったり結果が異なる点にも注意が必要です。