内容来自于国外的《[30 seconds of code](https://www.30secondsofcode.org)》，收录了 700 多个实用的代码片段，除了代码有启发性之外，命名也具有参考价值。

目前只翻译了标题和简短描述，可对该系列做个整体了解，点击标题可进入详细内容，使用 Google 翻译就能大体理解其含义。

与 [Underscore.js](https://underscorejs.org/) 和 [Lodash.js](https://lodash.com/docs/) 不同，30 seconds of code 中的代码片段都是各自独立的，可直接引用，也更容易吸收学习。

## JavaScript

#### 1）URL

1. [URLJoin](https://www.30secondsofcode.org/js/s/url-join)（将所有给定的URL段连接在一起，然后规范化结果URL）
2. [slugify](https://www.30secondsofcode.org/js/s/slugify)（将字符串转换为URL友好的段）
3. [queryStringToObject](https://www.30secondsofcode.org/js/s/query-string-to-object)（从给定的查询字符串或URL生成对象）
4. [如何在不重新加载页面的情况下使用JavaScript修改URL？](https://www.30secondsofcode.org/blog/s/javascript-modify-url-without-reload)
5. [getURLParameters](https://www.30secondsofcode.org/js/s/get-url-parameters)（创建一个包含当前URL参数的对象）
6. [getBaseURL](https://www.30secondsofcode.org/js/s/get-base-url)（获取当前URL，不带任何参数或片段标识符）
7. [isAbsoluteURL](https://www.30secondsofcode.org/js/s/is-absolute-url)（检查给定的字符串是否是绝对URL）
8. [redirect](https://www.30secondsofcode.org/js/s/redirect)（重定向到指定的URL）
9. [currentURL](https://www.30secondsofcode.org/js/s/current-url)（返回当前URL）
10. [getProtocol](https://www.30secondsofcode.org/js/s/get-protocol)（获取当前页面上正在使用的协议）

#### 2）字符串

1. [如何将可迭代对象转换为数组？](https://www.30secondsofcode.org/blog/s/javascript-iterable-to-array)
2. [正则表达式备忘单](https://www.30secondsofcode.org/blog/s/regexp-cheatsheet)
3. [6种JavaScript正则表达式功能](https://www.30secondsofcode.org/blog/s/6-javascript-regexp-tricks)
4. [JSONtoCSV](https://www.30secondsofcode.org/js/s/jso-nto-csv)（将对象数组转换为仅包含指定列的逗号分隔值（CSV）字符串）
5. [CSVToJSON](https://www.30secondsofcode.org/js/s/csv-to-json)（将逗号分隔值（CSV）字符串转换为2D对象数组。字符串的第一行用作标题行）
6. [arrayToCSV](https://www.30secondsofcode.org/js/s/array-to-csv)（将2D数组转换为逗号分隔值（CSV）字符串）
7. [CSVToArray](https://www.30secondsofcode.org/js/s/csv-to-array)（将逗号分隔值（CSV）字符串转换为2D数组）
8. [toTitleCase](https://www.30secondsofcode.org/js/s/to-title-case)（将字符串转换为标题的大小写）
9. [toCamelCase](https://www.30secondsofcode.org/js/s/to-camel-case)（将字符串转换为驼峰式）
10. [toSnakeCase](https://www.30secondsofcode.org/js/s/to-snake-case)（将字符串转换为蛇形）
11. [toKebabCase](https://www.30secondsofcode.org/js/s/to-kebab-case)（将字符串转换为烤肉串式）
12. [toRomanNumeral](https://www.30secondsofcode.org/js/s/to-roman-numeral)（将整数转换为其罗马数字表示形式。接受介于1和3999之间的值（包括两者））
13. [toCharArray](https://www.30secondsofcode.org/js/s/to-char-array)（将字符串转换为字符数组）
14. [toCurrency](https://www.30secondsofcode.org/js/s/to-currency)（接受数字并以指定的货币格式返回）
15. [swapCase](https://www.30secondsofcode.org/js/s/swap-case)（创建一个字符串，将大写字符转换为小写，反之亦然）
16. [decapitalize](https://www.30secondsofcode.org/js/s/decapitalize)（将字符串的首字母大写）
17. [capitalize](https://www.30secondsofcode.org/js/s/capitalize)（大写字符串的第一个字母）
18. [capitalizeEveryWord](https://www.30secondsofcode.org/js/s/capitalize-every-word)（将字符串中每个单词的首字母大写）
19. [size](https://www.30secondsofcode.org/js/s/size)（获取数组，对象或字符串的大小）
20. [mapString](https://www.30secondsofcode.org/js/s/map-string)（创建一个新字符串，其结果是在给定字符串中的每个字符上调用提供的函数）
21. [stringPermutations](https://www.30secondsofcode.org/js/s/string-permutations)（生成字符串的所有排列（包含重复项））
22. [words](https://www.30secondsofcode.org/js/s/words)（将给定的字符串转换为单词数组）
23. [reverseNumber](https://www.30secondsofcode.org/js/s/reverse-number)（倒数）
24. [sortCharactersInString](https://www.30secondsofcode.org/js/s/sort-characters-in-string)（按字母顺序对字符串中的字符进行排序）
25. [reverseString](https://www.30secondsofcode.org/js/s/reverse-string)（反转字符串）
26. [pluralize](https://www.30secondsofcode.org/js/s/pluralize)（根据输入的数字返回单词的单数或复数形式，并使用可选的字典（如果提供））
27. [prettyBytes](https://www.30secondsofcode.org/js/s/pretty-bytes)（将以字节为单位的数字转换为人类可读的字符串）
28. [sample](https://www.30secondsofcode.org/js/s/sample)（从数组中获取随机元素）
29. [randomAlphaNumeric](https://www.30secondsofcode.org/js/s/random-alpha-numeric)（生成具有指定长度的随机字符串）
30. [removeNonASCII](https://www.30secondsofcode.org/js/s/remove-non-ascii)（删除不可打印的ASCII字符）
31. [wordWrap](https://www.30secondsofcode.org/js/s/word-wrap)（使用换行符将字符串包装为给定数量的字符）
32. [escapeRegExp](https://www.30secondsofcode.org/js/s/escape-reg-exp)（转义要在正则表达式中使用的字符串）
33. [normalizeLineEndings](https://www.30secondsofcode.org/js/s/normalize-line-endings)（规范化字符串中的行尾）
34. [removeAccents](https://www.30secondsofcode.org/js/s/remove-accents)（从字符串中删除重音）
35. [splitLines](https://www.30secondsofcode.org/js/s/split-lines)（将多行字符串拆分为行数组）
36. [mask](https://www.30secondsofcode.org/js/s/mask)（用指定的掩码字符替换除最后一个字符以外的所有字符）
37. [compactWhitespace](https://www.30secondsofcode.org/js/s/compact-whitespace)（压缩字符串中的空格）
38. [removeWhitespace](https://www.30secondsofcode.org/js/s/remove-whitespace)（返回删除了空格的字符串）
39. [expandTabs](https://www.30secondsofcode.org/js/s/expand-tabs)（将制表符转换为空格，其中每个制表符对应于计数空格）
40. [containsWhitespace](https://www.30secondsofcode.org/js/s/contains-whitespace)（检查给定的字符串是否包含任何空格字符）
41. [stripHTMLTags](https://www.30secondsofcode.org/js/s/strip-html-tags)（从字符串中删除 HTML / XML 标签）
42. [byteSize](https://www.30secondsofcode.org/js/s/byte-size)（返回字符串的长度（以字节为单位））
43. [fromCamelCase](https://www.30secondsofcode.org/js/s/from-camel-case)（从驼峰式转换字符串）
44. [pad](https://www.30secondsofcode.org/js/s/pad)（如果字符串短于指定的长度，则在两侧用指定的字符填充字符串）
45. [padNumber](https://www.30secondsofcode.org/js/s/pad-number)（将给定数字填充到指定长度）
46. [truncateString](https://www.30secondsofcode.org/js/s/truncate-string)（将字符串截断到指定长度）
47. [truncateStringAtWhitespace](https://www.30secondsofcode.org/js/s/truncate-string-at-whitespace)（将字符串截断到指定长度，并在可能的情况下使用空格）
48. [formatNumber](https://www.30secondsofcode.org/js/s/format-number)（使用本地号码格式顺序格式化号码）
49. [indentString](https://www.30secondsofcode.org/js/s/indent-string)（缩进提供的字符串中的每一行）
50. [isLowerCase](https://www.30secondsofcode.org/js/s/is-lower-case)（检查字符串是否为小写）
51. [isUpperCase](https://www.30secondsofcode.org/js/s/is-upper-case)（检查字符串是否为大写）
52. [isAnagram](https://www.30secondsofcode.org/js/s/is-anagram)（检查一个字符串是否是另一个字符串的字谜（不区分大小写，忽略空格，标点符号和特殊字符））
53. [isEmpty](https://www.30secondsofcode.org/js/s/is-empty)（检查 a 值是否为空对象/集合，没有可枚举的属性或任何不视为集合的类型）
54. [isAlpha](https://www.30secondsofcode.org/js/s/is-alpha)（检查字符串是否仅包含字母字符）
55. [isAlphaNumeric](https://www.30secondsofcode.org/js/s/is-alpha-numeric)（检查字符串是否仅包含字母数字字符）
56. [isString](https://www.30secondsofcode.org/js/s/is-string)（检查给定参数是否为字符串，仅适用于原始数据类型）

#### 3）对象

1. [代码剖析-循环，数组 Reduce 和链式](https://www.30secondsofcode.org/blog/s/code-anatomy-chaining-reduce-for-loop)
2. [什么是迭代器？在哪里可以使用它们？](https://www.30secondsofcode.org/blog/s/javascript-iterators)
3. [for ... in，for ... of和forEach之间有什么区别？](https://www.30secondsofcode.org/blog/s/javascript-for-in-for-of-foreach)
4. [forOwn](https://www.30secondsofcode.org/js/s/for-own)（遍历对象的所有自身属性，为每个对象运行一个回调）
5. [forOwnRight](https://www.30secondsofcode.org/js/s/for-own-right)（反向迭代对象的所有自身属性，为每个对象运行一个回调）
6. [如何克隆对象？](https://www.30secondsofcode.org/blog/s/javascript-shallow-deep-clone)（了解JavaScript如何处理可变数据（例如对象和数组），并了解浅层克隆和深层克隆的工作方式）
7. [shallowClone](https://www.30secondsofcode.org/js/s/shallow-clone)（创建对象的浅层克隆）
8. [deepClone](https://www.30secondsofcode.org/js/s/deep-clone)（创建对象的深层克隆。 克隆基元，数组和对象，不包括类实例）
9. [如何使用解构赋值语法？](https://www.30secondsofcode.org/blog/s/javascript-destructuring-assignment)
10. [如何深度冻结对象？](https://www.30secondsofcode.org/blog/s/javascript-deep-freeze-object)（了解可变性在JavaScript中的工作原理，其对对象的应用程序以及如何正确冻结它们以使其不可变）
11. [deepFreeze](https://www.30secondsofcode.org/js/s/deep-freeze)（深度冻结对象）
12. [get](https://www.30secondsofcode.org/js/s/get)（从对象检索给定选择器指示的一组属性）
13. [deepGet](https://www.30secondsofcode.org/js/s/deep-get)（根据 keys 数组获取嵌套 JSON 对象中的目标值）
14. [renameKeys](https://www.30secondsofcode.org/js/s/rename-keys)（用提供的值替换多个对象键的名称）
15. [compactObject](https://www.30secondsofcode.org/js/s/compact-object)（从对象或数组中深度删除所有假值）
16. [countBy](https://www.30secondsofcode.org/js/s/count-by)（根据给定的函数对数组的元素进行分组，并返回每组中元素的计数）
17. [groupBy](https://www.30secondsofcode.org/js/s/group-by)（根据给定函数对数组的元素进行分组）
18. [omitBy](https://www.30secondsofcode.org/js/s/omit-by)（省略与给定函数为其返回falsy的对象的键对应的键值对）
19. [pickBy](https://www.30secondsofcode.org/js/s/pick-by)（创建一个对象，该对象由属性组成，给定的函数返回 true）
20. [partitionBy](https://www.30secondsofcode.org/js/s/partition-by)（将 fn 应用于数组中的每个值，每次提供的函数返回新值时将其拆分）
21. [orderBy](https://www.30secondsofcode.org/js/s/order-by)（按属性和顺序对对象数组进行排序）
22. [orderWith](https://www.30secondsofcode.org/js/s/order-with)（根据提供的顺序数组，按属性排序对象的数组）
23. [deepMapKeys](https://www.30secondsofcode.org/js/s/deep-map-keys)（深度映射对象的键）
24. [mapKeys](https://www.30secondsofcode.org/js/s/map-keys)（使用提供的函数映射对象的键，生成一个新对象）
25. [mapObject](https://www.30secondsofcode.org/js/s/map-object)（使用函数将数组的值映射到对象）
26. [mapValues](https://www.30secondsofcode.org/js/s/map-values)（使用提供的函数映射对象的值，并使用相同的键生成一个新对象）
27. [flattenObject](https://www.30secondsofcode.org/js/s/flatten-object)（使用键的路径扁平化对象）
28. [unflattenObject](https://www.30secondsofcode.org/js/s/unflatten-object)（使用键的路径展开对象）
29. [formToObject](https://www.30secondsofcode.org/js/s/form-to-object)（将一组表单元素编码为一个对象）
30. [zipObject](https://www.30secondsofcode.org/js/s/zip-object)（将属性与值，给定的有效属性标识符数组和值数组相关联）
31. [invertKeyValues](https://www.30secondsofcode.org/js/s/invert-key-values)（反转对象的键值对，而无需对其进行突变）
32. [partition](https://www.30secondsofcode.org/js/s/partition)（将元素分为两个数组，具体取决于每个元素提供的函数的真实性）
33. [dig](https://www.30secondsofcode.org/js/s/dig)（根据给定的键获取嵌套 JSON 对象中的目标值）
34. [stringifyCircularJSON](https://www.30secondsofcode.org/js/s/stringify-circular-json)（将包含循环引用的 JSON 对象序列化为 JSON 格式）
35. [hasKey](https://www.30secondsofcode.org/js/s/has-key)（检查目标值是否存在于 JSON 对象中）
36. [frequencies](https://www.30secondsofcode.org/js/s/frequencies)（创建一个对象，将数组的唯一值作为键，并将其频率作为值）
37. [functions](https://www.30secondsofcode.org/js/s/functions)（从对象自己的（并且可以继承）可枚举的属性中获取函数属性名称的数组）
38. [defaults](https://www.30secondsofcode.org/js/s/defaults)（为未定义对象中的所有属性分配默认值）
39. [nest](https://www.30secondsofcode.org/js/s/nest)（以递归方式将彼此链接的对象嵌套在一个平面数组中）
40. [omit](https://www.30secondsofcode.org/js/s/omit)（从对象中省略与给定键对应的键值对）
41. [pick](https://www.30secondsofcode.org/js/s/pick)（从对象中选择与给定键对应的键值对）
42. [pluck](https://www.30secondsofcode.org/js/s/pluck)（将对象数组转换为与指定键对应的值数组）
43. [walkThrough](https://www.30secondsofcode.org/js/s/walk-through)（创建一个生成器，遍历给定对象的所有键）
44. [findKey](https://www.30secondsofcode.org/js/s/find-key)（查找满足所提供测试功能的第一个键。 否则返回undefined）
45. [findLastKey](https://www.30secondsofcode.org/js/s/find-last-key)（查找满足所提供测试功能的最后一个键，否则返回 undefined）
46. [merge](https://www.30secondsofcode.org/js/s/merge)（通过两个或多个对象的组合创建一个新对象）
47. [combine](https://www.30secondsofcode.org/js/s/combine)（合并两个对象数组，使用指定的键来匹配对象）
48. [toPairs](https://www.30secondsofcode.org/js/s/to-pairs)（从对象或其他可迭代对象创建键值对数组的数组）
49. [transform](https://www.30secondsofcode.org/js/s/transform)（对一个累加器和对象中的每个键应用一个函数（从左到右））
50. [lowercaseKeys](https://www.30secondsofcode.org/js/s/lowercase-keys)（从指定的对象创建一个新的对象，其中所有键都小写）
51. [findKeys](https://www.30secondsofcode.org/js/s/find-keys)（在提供的对象中查找与给定值匹配的所有键）
52. [truthCheckCollection](https://www.30secondsofcode.org/js/s/truth-check-collection)（检查判定函数对于集合的所有元素是否为真）
53. [objectToQueryString](https://www.30secondsofcode.org/js/s/object-to-query-string)（从给定对象的键值对生成查询字符串）
54. [objectFromPairs](https://www.30secondsofcode.org/js/s/object-from-pairs)（根据给定的键值对创建对象）
55. [objectToEntries](https://www.30secondsofcode.org/js/s/object-to-entries)（从一个对象创建一个键值对数组）
56. [objectToPairs](https://www.30secondsofcode.org/js/s/object-to-pairs)（从一个对象创建一个键值对数组）
57. [matches](https://www.30secondsofcode.org/js/s/matches)（比较两个对象以确定第一个对象是否包含与第二个对象相同的属性值）
58. [matchesWith](https://www.30secondsofcode.org/js/s/matches-with)（根据提供的函数比较两个对象，以确定第一个对象是否包含与第二个对象相同的属性值）
59. [checkProp](https://www.30secondsofcode.org/js/s/check-prop)（创建一个函数，该函数将为给定对象上的指定属性调用判定函数）
60. [equals](https://www.30secondsofcode.org/js/s/equals)（在两个值之间进行深度比较以确定它们是否等效）
61. [isDeepFrozen](https://www.30secondsofcode.org/js/s/is-deep-frozen)（检查对象是否被深度冻结）
62. [isEmpty](https://www.30secondsofcode.org/js/s/is-empty)（检查 a 值是否为空对象/集合，没有可枚举的属性或任何不视为集合的类型）
63. [isPlainObject](https://www.30secondsofcode.org/js/s/is-plain-object)（检查提供的值是否是由 Object 构造函数创建的对象）
64. [isObject](https://www.30secondsofcode.org/js/s/is-object)
65. [isObjectLike](https://www.30secondsofcode.org/js/s/is-object-like)（检查值是否类似于对象）

#### 4）函数

1. [了解 ES6 的扩展运算符和剩余参数](https://www.30secondsofcode.org/blog/s/javascript-spread-rest-syntax)
2. [如何使用记忆函数？](https://www.30secondsofcode.org/blog/s/javascript-memoization)
3. [memoize](https://www.30secondsofcode.org/js/s/memoize)（返回备注的（缓存的）函数）
4. [如何实现单例？](https://www.30secondsofcode.org/blog/s/javascript-singleton-proxy)（使用Proxy对象在JavaScript中实现单例）
5. [了解JavaScript中的高阶函数](https://www.30secondsofcode.org/blog/s/javascript-higher-order-functions)
6. [将JavaScript生成器函数用于范围](https://www.30secondsofcode.org/blog/s/javascript-range-generator)（使用JavaScript ES6生成器和迭代器在数字范围内进行迭代）
7. [cycleGenerator](https://www.30secondsofcode.org/js/s/cycle-generator)（创建一个生成器，无限循环遍历给定数组）
8. [repeatGenerator](https://www.30secondsofcode.org/js/s/repeat-generator)（创建一个生成器，无限期地重复给定值）
9. [generatorToArray](https://www.30secondsofcode.org/js/s/generator-to-array)（将生成器函数的输出转换为数组）
10. [rangeGenerator](https://www.30secondsofcode.org/js/s/range-generator)（创建一个生成器，使用给定的步骤生成给定范围内的所有值）
11. [unfold](https://www.30secondsofcode.org/js/s/unfold)（使用迭代器函数和初始值构建数组）
12. [异步JavaScript备忘单](https://www.30secondsofcode.org/blog/s/async-javascript-cheatsheet)（了解有关Promise和异步JavaScript所需的所有知识）
13. [chainAsync](https://www.30secondsofcode.org/js/s/chain-async)（链接异步函数）
14. [了解JavaScript中的“ this”关键字](https://www.30secondsofcode.org/blog/s/javascript-this)（了解它在不同情况下的工作方式并正确使用它）
15. [如何使用布尔函数？](https://www.30secondsofcode.org/blog/s/javascript-boolean-function)
16. [什么是JavaScript闭包？](https://www.30secondsofcode.org/blog/s/javascript-closures)（学习和理解闭包（JavaScript编程的核心概念）并升级代码）
17. [JavaScript构造函数返回什么？](https://www.30secondsofcode.org/blog/s/javascript-return-constructor)
18. [call()、apply() 和 bind()](https://www.30secondsofcode.org/blog/s/javascript-function-call-apply-bind)
19. [call](https://www.30secondsofcode.org/js/s/call)（给定一个键和一组参数，在给定上下文时调用它们）
20. [bind](https://www.30secondsofcode.org/js/s/bind)（创建一个在给定上下文中调用 fn 的函数，可以选择在参数之前添加任何其他提供的参数）
21. [bindAll](https://www.30secondsofcode.org/js/s/bind-all)（将对象的方法绑定到对象本身，覆盖现有方法）
22. [bindKey](https://www.30secondsofcode.org/js/s/bind-key)（创建一个函数，该函数在对象的给定键处调用该方法，可以选择在参数的前面附加任何提供的参数）
23. [pipeAsyncFunctions](https://www.30secondsofcode.org/js/s/pipe-async-functions)（对异步功能执行从左到右的功能组合）
24. [pipeFunctions](https://www.30secondsofcode.org/js/s/pipe-functions)（执行从左到右的功能合成）
25. [converge](https://www.30secondsofcode.org/js/s/converge)（接受一个聚合函数和一个分支函数列表，将分支函数的结果作为参数传递给聚合函数）
26. [rearg](https://www.30secondsofcode.org/js/s/rearg)（创建一个函数，该函数调用提供的函数，其参数根据指定的索引排列）
27. [functions](https://www.30secondsofcode.org/js/s/functions)（从对象自己的（并且可以继承）可枚举的属性中获取函数属性名称的数组）
28. [runPromisesInSeries](https://www.30secondsofcode.org/js/s/run-promises-in-series)（连续执行一系列 Promise）
29. [promisify](https://www.30secondsofcode.org/js/s/promisify)（转换一个异步函数以返回一个 Promise）
30. [mostPerformant](https://www.30secondsofcode.org/js/s/most-performant)（返回执行最快的函数数组中的函数索引）
31. [once](https://www.30secondsofcode.org/js/s/once)（确保一个函数仅被调用一次）
32. [overArgs](https://www.30secondsofcode.org/js/s/over-args)（创建一个函数，该函数调用提供的函数并转换其参数）
33. [spreadOver](https://www.30secondsofcode.org/js/s/spread-over)（接受可变参数函数，并返回接受参数数组的函数）
34. [throttle](https://www.30secondsofcode.org/js/s/throttle)（创建一个限制的函数，每等待一个毫秒最多只能调用一次提供的函数）
35. [debounce](https://www.30secondsofcode.org/js/s/debounce)（创建一个去抖动的函数，该函数会将调用提供的函数延迟到从上次调用该函数起至少经过了 ms 毫秒）
36. [debouncePromise](https://www.30secondsofcode.org/js/s/debounce-promise)（创建一个防抖动的函数，该函数返回一个 Promise，但是将调用提供的函数的时间延迟到从上一次调用它起至少经过了ms毫秒。在此期间返回的所有 Promise 将返回相同的数据）
37. [defer](https://www.30secondsofcode.org/js/s/defer)（推迟调用函数，直到清除当前调用堆栈）
38. [sleep](https://www.30secondsofcode.org/js/s/sleep)（延迟异步功能的执行）
39. [delay](https://www.30secondsofcode.org/js/s/delay)（ms毫秒后调用提供的功能）
40. [compose](https://www.30secondsofcode.org/js/s/compose)（执行从右到左的功能组合）
41. [composeRight](https://www.30secondsofcode.org/js/s/compose-right)（执行从左到右的功能组合）
42. [partial](https://www.30secondsofcode.org/js/s/partial)（创建一个函数，该函数调用带有接收到的参数前面的部分内容的 fn）
43. [partialRight](https://www.30secondsofcode.org/js/s/partial-right)（创建一个函数，该函数调用 fn 并在接收到的参数后附加部分）
44. [uncurry](https://www.30secondsofcode.org/js/s/uncurry)（取消对深度 n 的函数）
45. [ary](https://www.30secondsofcode.org/js/s/ary)（创建一个最多接受 n 个参数的函数，忽略任何其他参数）
46. [juxt](https://www.30secondsofcode.org/js/s/juxt)（将几个函数作为参数并返回与这些函数并列的函数）
47. [over](https://www.30secondsofcode.org/js/s/over)（创建一个函数，该函数使用接收到的参数调用每个提供的函数并返回结果）
48. [flip](https://www.30secondsofcode.org/js/s/flip)（将函数作为参数，然后使第一个参数成为最后一个参数）
49. [nthArg](https://www.30secondsofcode.org/js/s/nth-arg)（创建一个函数以索引 n 获取参数）
50. [coalesceFactory](https://www.30secondsofcode.org/js/s/coalesce-factory)（自定义合并函数，该函数根据给定的验证器返回第一个为真的参数）
51. [collectInto](https://www.30secondsofcode.org/js/s/collect-into)（将接受数组的函数更改为可变函数）
52. [generateItems](https://www.30secondsofcode.org/js/s/generate-items)（使用给定的函数生成具有给定数量的项的数组）
53. [checkProp](https://www.30secondsofcode.org/js/s/check-prop)（创建一个函数，该函数将为给定对象上的指定属性执行回调函数）
54. [curry](https://www.30secondsofcode.org/js/s/curry)（科里化一个函数）
55. [times](https://www.30secondsofcode.org/js/s/times)（遍历回调 n 次）
56. [timeTaken](https://www.30secondsofcode.org/js/s/time-taken)（衡量功能执行所需的时间）
57. [attempt](https://www.30secondsofcode.org/js/s/attempt)（尝试使用提供的参数调用函数，返回结果或捕获的错误对象）
58. [binary](https://www.30secondsofcode.org/js/s/binary)（创建一个最多接受两个参数的函数，忽略任何其他参数）
59. [complement](https://www.30secondsofcode.org/js/s/complement)（返回一个函数，该函数是给定函数 fn 的逻辑补码）
60. [either](https://www.30secondsofcode.org/js/s/either)（检查是否有至少一个函数针对给定的一组参数返回 true）
61. [negate](https://www.30secondsofcode.org/js/s/negate)（否定判定函数的结果）
62. [functionName](https://www.30secondsofcode.org/js/s/function-name)（记录函数名称）
63. [unary](https://www.30secondsofcode.org/js/s/unary)（创建一个函数，该函数最多接受一个参数，而忽略任何其他参数）
64. [when](https://www.30secondsofcode.org/js/s/when)（返回一个函数，该函数接受一个参数，如果不正确，则运行回调，如果错误，则返回）
65. [isPromiseLike](https://www.30secondsofcode.org/js/s/is-promise-like)（检查对象是否看起来像一个 Promise）
66. [isAsyncFunction](https://www.30secondsofcode.org/js/s/is-async-function)（检查给定参数是否为异步函数）
67. [isGeneratorFunction](https://www.30secondsofcode.org/js/s/is-generator-function)（检查给定参数是否为生成器函数）
68. [isFunction](https://www.30secondsofcode.org/js/s/is-function)（检查给定参数是否为函数）

#### 5）Math

1. [sum](https://www.30secondsofcode.org/js/s/sum)（计算两个或多个数字/数组的总和）
2. [sumN](https://www.30secondsofcode.org/js/s/sum-n)（对 1 到 n 之间的所有数字求和）
3. [sumPower](https://www.30secondsofcode.org/js/s/sum-power)（计算从开始到结束（包括两个端点）的所有数字的幂和）
4. [toOrdinalSuffix](https://www.30secondsofcode.org/js/s/to-ordinal-suffix)（接受数字并将其作为字符串返回，并带有正确的序号指示符后缀）
5. [toDecimalMark](https://www.30secondsofcode.org/js/s/to-decimal-mark)（将数字转换为小数点格式的字符串）
6. [toSafeInteger](https://www.30secondsofcode.org/js/s/to-safe-integer)（将值转换为安全整数）
7. [averageBy](https://www.30secondsofcode.org/js/s/average-by)（使用提供的函数将每个元素映射到一个值后，计算数组的平均值）
8. [sumBy](https://www.30secondsofcode.org/js/s/sum-by)（使用提供的函数将每个元素映射到值后，计算数组的总和）
9. [maxBy](https://www.30secondsofcode.org/js/s/max-by)（使用提供的函数将每个元素映射到一个值后，返回数组的最大值）
10. [minBy](https://www.30secondsofcode.org/js/s/min-by)（使用提供的函数将每个元素映射到一个值后，返回数组的最小值）
11. [sortedIndexBy](https://www.30secondsofcode.org/js/s/sorted-index-by)（根据提供的迭代器函数，找到应将值插入数组以保持其排序顺序的最低索引）
12. [digitize](https://www.30secondsofcode.org/js/s/digitize)（将数字转换为数字数组，必要时删除其符号）
13. [accumulate](https://www.30secondsofcode.org/js/s/accumulate)（创建一个部分和数组）
14. [maxN](https://www.30secondsofcode.org/js/s/max-n)（从提供的数组中返回 n 个最大元素）
15. [minN](https://www.30secondsofcode.org/js/s/min-n)（返回提供的数组中的 n 个最小元素）
16. [fibonacci](https://www.30secondsofcode.org/js/s/fibonacci)（生成一个包含斐波那契序列的数组，直到第 n 个项）
17. [xProd](https://www.30secondsofcode.org/js/s/x-prod)（通过从数组中创建每个可能的对，在提供的两个数组中创建一个新数组）
18. [sdbm](https://www.30secondsofcode.org/js/s/sdbm)（将输入字符串哈希为整数）
19. [symmetricDifference](https://www.30secondsofcode.org/js/s/symmetric-difference)（返回两个数组之间的对称差，而不会过滤出重复的值）
20. [uniqueSymmetricDifference](https://www.30secondsofcode.org/js/s/unique-symmetric-difference)（返回两个数组之间的唯一对称差异，不包含两个数组中的重复值）
21. [cartesianProduct](https://www.30secondsofcode.org/js/s/cartesian-product)（计算两个数组的笛卡尔积）
22. [standardDeviation](https://www.30secondsofcode.org/js/s/standard-deviation)（计算数字数组的标准差）
23. [weightedAverage](https://www.30secondsofcode.org/js/s/weighted-average)（计算两个或多个数字的加权平均值）
24. [percentile](https://www.30secondsofcode.org/js/s/percentile)（计算给定数组中小于或等于给定值的数字的百分比）
25. [prod](https://www.30secondsofcode.org/js/s/prod)（计算两个或多个数字/数组的乘积）
26. [sortedIndex](https://www.30secondsofcode.org/js/s/sorted-index)（查找应将值插入数组以保持其排序顺序的最低索引）
27. [average](https://www.30secondsofcode.org/js/s/average)（计算两个或多个数字的平均值）
28. [median](https://www.30secondsofcode.org/js/s/median)（计算数字数组的中位数）
29. [factorial](https://www.30secondsofcode.org/js/s/factorial)（计算数字的阶乘）
30. [logBase](https://www.30secondsofcode.org/js/s/log-base)（计算给定基数中给定数字的对数）
31. [nthRoot](https://www.30secondsofcode.org/js/s/nth-root)（计算给定数字的第 n 个根）
32. [vectorAngle](https://www.30secondsofcode.org/js/s/vector-angle)（计算两个向量之间的角度（θ））
33. [midpoint](https://www.30secondsofcode.org/js/s/midpoint)（计算两对 (x, y) 点之间的中点）
34. [distance](https://www.30secondsofcode.org/js/s/distance)（计算两点之间的距离）
35. [round](https://www.30secondsofcode.org/js/s/round)（将数字四舍五入到指定位数）
36. [similarity](https://www.30secondsofcode.org/js/s/similarity)（返回同时出现在两个数组中的元素数组）
37. [randomIntArrayInRange](https://www.30secondsofcode.org/js/s/random-int-array-in-range)（生成指定范围内 n 个随机整数的数组）
38. [randomIntegerInRange](https://www.30secondsofcode.org/js/s/random-integer-in-range)（生成指定范围内的随机整数）
39. [randomNumberInRange](https://www.30secondsofcode.org/js/s/random-number-in-range)（生成指定范围内的随机数）
40. [randomHexColorCode](https://www.30secondsofcode.org/js/s/random-hex-color-code)（生成随机的十六进制颜色代码）
41. [randomBoolean](https://www.30secondsofcode.org/js/s/random-boolean)（产生一个随机的布尔值）
42. [mapNumRange](https://www.30secondsofcode.org/js/s/map-num-range)（将数字从一个范围映射到另一个范围）
43. [divmod](https://www.30secondsofcode.org/js/s/divmod)（返回由给定数字的商和余数组成的数组）
44. [clampNumber](https://www.30secondsofcode.org/js/s/clamp-number)（在指定区间内根据条件返回一个数字）
45. [copySign](https://www.30secondsofcode.org/js/s/copy-sign)（返回第一个数字的绝对值，第二个的符号）
46. [degreesToRads](https://www.30secondsofcode.org/js/s/degrees-to-rads)（将角度从度转换为弧度）
47. [radsToDegrees](https://www.30secondsofcode.org/js/s/rads-to-degrees)（将角度从弧度转换为度）
48. [approximatelyEqual](https://www.30secondsofcode.org/js/s/approximately-equal)（检查两个数字是否彼此相等）
49. [inRange](https://www.30secondsofcode.org/js/s/in-range)（检查给定数字是否在给定范围内）
50. [validateNumber](https://www.30secondsofcode.org/js/s/validate-number)（检查给定值是否为数字）
51. [isPowerOfTen](https://www.30secondsofcode.org/js/s/is-power-of-ten)（检查给定数字是否为10的幂）
52. [isDivisible](https://www.30secondsofcode.org/js/s/is-divisible)（检查第一个数字参数是否可被第二个参数整除）
53. [isEven](https://www.30secondsofcode.org/js/s/is-even)（检查给定数字是否为偶数）
54. [isOdd](https://www.30secondsofcode.org/js/s/is-odd)（检查给定数字是否为奇数）
55. [isPowerOfTwo](https://www.30secondsofcode.org/js/s/is-power-of-two)（检查给定数字是否为 2 的幂）
56. [isNegativeZero](https://www.30secondsofcode.org/js/s/is-negative-zero)（检查给定值是否等于负零（-0））
57. [isNumber](https://www.30secondsofcode.org/js/s/is-number)（检查给定参数是否为数字）
58. [isPrime](https://www.30secondsofcode.org/js/s/is-prime)（检查提供的整数是否为质数）

#### 6）类型

1. [如何使用布尔函数？](https://www.30secondsofcode.org/blog/s/javascript-boolean-function)
2. [声明提升是指什么](https://www.30secondsofcode.org/blog/s/javascript-variable-hoisting)
3. [如何使用可选的链接和无效合并？](https://www.30secondsofcode.org/blog/s/javascript-nullish-coalescing-optional-chaining)（ES2020 中的概念）
4. [了解JavaScript变量和作用域](https://www.30secondsofcode.org/blog/s/javascript-variable-scope)
5. [相等运算符之间有什么区别？](https://www.30secondsofcode.org/blog/s/javascript-equality)
6. [equals](https://www.30secondsofcode.org/js/s/equals)（在两个值之间进行深度比较以确定它们是否等效）
7. [coalesceFactory](https://www.30secondsofcode.org/js/s/coalesce-factory)
8. [getType](https://www.30secondsofcode.org/js/s/get-type)（返回值的本地类型）
9. [castArray](https://www.30secondsofcode.org/js/s/cast-array)（如果提供的值不是一个，则将其强制转换为数组）
10. [cloneRegExp](https://www.30secondsofcode.org/js/s/clone-reg-exp)（克隆正则表达式）
11. [coalesce](https://www.30secondsofcode.org/js/s/coalesce)（返回第一个已定义的非空参数）
12. [is](https://www.30secondsofcode.org/js/s/is)（检查提供的值是否为指定类型）
13. [isDuplexStream](https://www.30secondsofcode.org/js/s/is-duplex-stream)
14. [isWritableStream](https://www.30secondsofcode.org/js/s/is-writable-stream)
15. [isEmpty](https://www.30secondsofcode.org/js/s/is-empty)
16. [isPromiseLike](https://www.30secondsofcode.org/js/s/is-promise-like)（检查对象是否看起来像一个Promise）
17. [isReadableStream](https://www.30secondsofcode.org/js/s/is-readable-stream)
18. [isAsyncFunction](https://www.30secondsofcode.org/js/s/is-async-function)
19. [isGeneratorFunction](https://www.30secondsofcode.org/js/s/is-generator-function)
20. [isArrayLike](https://www.30secondsofcode.org/js/s/is-array-like)（检查所提供的参数是否类似于数组（即可迭代））
21. [isStream](https://www.30secondsofcode.org/js/s/is-stream)
22. [isValidJSON](https://www.30secondsofcode.org/js/s/is-valid-json)
23. [isPlainObject](https://www.30secondsofcode.org/js/s/is-plain-object)（检查提供的值是否是由Object构造函数创建的对象）
24. [isPrimitive](https://www.30secondsofcode.org/js/s/is-primitive)（检查传递的值是否为原始值）
25. [isFunction](https://www.30secondsofcode.org/js/s/is-function)
26. [isObject](https://www.30secondsofcode.org/js/s/is-object)
27. [isObjectLike](https://www.30secondsofcode.org/js/s/is-object-like)（检查值是否类似于对象）
28. [isString](https://www.30secondsofcode.org/js/s/is-string)
29. [isBoolean](https://www.30secondsofcode.org/js/s/is-boolean)
30. [isNumber](https://www.30secondsofcode.org/js/s/is-number)
31. [isSymbol](https://www.30secondsofcode.org/js/s/is-symbol)
32. [isNil](https://www.30secondsofcode.org/js/s/is-nil)（检查指定的值是否为 null 或 undefined）
33. [isNull](https://www.30secondsofcode.org/js/s/is-null)
34. [isUndefined](https://www.30secondsofcode.org/js/s/is-undefined)

#### 7）日期

1. [formatDuration](https://www.30secondsofcode.org/js/s/format-duration)（返回人类可读格式的给定毫秒数）
2. [如何实现 sleep 功能？](https://www.30secondsofcode.org/blog/s/javascript-sleep)（实现 sleep() 函数的所有不同方式）
3. [countWeekDaysBetween](https://www.30secondsofcode.org/js/s/count-week-days-between)（计算两个日期之间的工作日）
4. [toISOStringWithTimezone](https://www.30secondsofcode.org/js/s/to-iso-string-with-timezone)（将日期转换为扩展的ISO格式（ISO 8601），包括时区偏移量）
5. [addWeekDays](https://www.30secondsofcode.org/js/s/add-week-days)（在添加给定的工作日后计算日期）
6. [addDaysToDate](https://www.30secondsofcode.org/js/s/add-days-to-date)（计算给定日期后 n 天的日期，并返回其字符串的表示形式）
7. [addMinutesToDate](https://www.30secondsofcode.org/js/s/add-minutes-to-date)（从给定日期计算 n 分钟的日期，并返回其字符串的表示形式）
8. [quarterOfYear](https://www.30secondsofcode.org/js/s/quarter-of-year)（返回提供的日期所属的季度和年份）
9. [dayOfYear](https://www.30secondsofcode.org/js/s/day-of-year)（从Date对象获取一年中的一天（范围为1-366的数字））
10. [daysFromNow](https://www.30secondsofcode.org/js/s/days-from-now)（以字符串表示形式计算从今天开始的 n 天的日期）
11. [maxDate](https://www.30secondsofcode.org/js/s/max-date)（返回给定日期的最大值）
12. [minDate](https://www.30secondsofcode.org/js/s/min-date)（返回给定日期的最小值）
13. [dayName](https://www.30secondsofcode.org/js/s/day-name)（从 Date 对象获取工作日的名称）
14. [daysAgo](https://www.30secondsofcode.org/js/s/days-ago)（计算从今天起 n 天前的日期，以字符串表示形式）
15. [getColonTimeFromDate](https://www.30secondsofcode.org/js/s/get-colon-time-from-date)（从Date对象返回形式为 HH:MM:SS 的字符串）
16. [getMonthsDiffBetweenDates](https://www.30secondsofcode.org/js/s/get-months-diff-between-dates)（计算两个日期之间的差异（以月为单位））
17. [getDaysDiffBetweenDates](https://www.30secondsofcode.org/js/s/get-days-diff-between-dates)（计算两个日期之间的差异（以天为单位））
18. [getMeridiemSuffixOfInteger](https://www.30secondsofcode.org/js/s/get-meridiem-suffix-of-integer)（将整数转换为带后缀的字符串，并根据其值添加am或pm）
19. [getTimestamp](https://www.30secondsofcode.org/js/s/get-timestamp)（从 Date 对象获取 Unix 时间戳）
20. [fromTimestamp](https://www.30secondsofcode.org/js/s/from-timestamp)（从 Unix 时间戳创建 Date 对象）
21. [tomorrow](https://www.30secondsofcode.org/js/s/tomorrow)（结果以字符串形式表示明天的日期）
22. [yesterday](https://www.30secondsofcode.org/js/s/yesterday)（结果以字符串形式表示昨天的日期）
23. [lastDateOfMonth](https://www.30secondsofcode.org/js/s/last-date-of-month)（返回给定日期月份的最后一个日期的字符串表示形式）
24. [isSameDate](https://www.30secondsofcode.org/js/s/is-same-date)（检查一个日期是否与另一个日期相同）
25. [isLeapYear](https://www.30secondsofcode.org/js/s/is-leap-year)（检查给定年份是否为闰年）
26. [isWeekday](https://www.30secondsofcode.org/js/s/is-weekday)（检查给定的日期是否是工作日）
27. [isWeekend](https://www.30secondsofcode.org/js/s/is-weekend)（检查给定的日期是否是周末）
28. [isAfterDate](https://www.30secondsofcode.org/js/s/is-after-date)（检查某个日期是否在另一个日期之后）
29. [isBeforeDate](https://www.30secondsofcode.org/js/s/is-before-date)（检查日期是否在另一个日期之前）
30. [isBetweenDates](https://www.30secondsofcode.org/js/s/is-between-dates)（检查日期是否在其他两个日期之间）
31. [isDateValid](https://www.30secondsofcode.org/js/s/is-date-valid)（检查是否可以根据给定的值创建有效的日期对象）
32. [isISOString](https://www.30secondsofcode.org/js/s/is-iso-string)（检查给定的字符串在简化的扩展ISO格式（ISO 8601）中是否有效）

#### 8）浏览器

1. [如何检测Caps Lock大写键是否已启用？](https://www.30secondsofcode.org/blog/s/detect-caps-lock-is-on)
2. [可以用箭头函数作为事件监听器的回调吗？](https://www.30secondsofcode.org/blog/s/javascript-arrow-function-event-listeners)
3. [如何使用JavaScript将文本复制到剪贴板？](https://www.30secondsofcode.org/blog/s/copy-text-to-clipboard-with-javascript)
4. [copyToClipboard](https://www.30secondsofcode.org/js/s/copy-to-clipboard)（将字符串复制到剪贴板，仅由于用户操作（即在 click 事件侦听器内部）而起作用）
5. [Tip: 您可以将输入元素的值获取为数字](https://www.30secondsofcode.org/blog/s/javascript-value-as-number)
6. [了解JavaScript中的事件冒泡，捕获和委托](https://www.30secondsofcode.org/blog/s/javascript-event-bubbling-capturing-delegation)
7. [如何最多执行一次事件处理程序？](https://www.30secondsofcode.org/blog/s/javascript-listen-once)
8. [addMultipleListeners](https://www.30secondsofcode.org/js/s/add-multiple-events)（将具有相同处理程序的多个事件侦听器添加到元素中）
9. [createEventHub](https://www.30secondsofcode.org/js/s/create-event-hub)（使用emit，on和off方法创建一个发布/订阅（发布-订阅）事件中心）
10. [on](https://www.30secondsofcode.org/js/s/on)（将事件侦听器添加到具有事件委托功能的元素）
11. [off](https://www.30secondsofcode.org/js/s/off)（从元素中删除事件监听器）
12. [listenOnce](https://www.30secondsofcode.org/js/s/listen-once)（将事件侦听器添加到仅在首次触发事件时运行回调的元素）
13. [triggerEvent](https://www.30secondsofcode.org/js/s/trigger-event)（在给定元素上触发特定事件，可以选择传递自定义数据）
14. [onClickOutside](https://www.30secondsofcode.org/js/s/on-click-outside)（每当用户单击指定元素之外时，运行回调）
15. [onUserInputChange](https://www.30secondsofcode.org/js/s/on-user-input-change)（每当用户输入类型更改（鼠标或触摸）时运行回调）
16. [supportsTouchEvents](https://www.30secondsofcode.org/js/s/supports-touch-events)（检查是否支持触摸事件）
17. [cookies，localStorage 和 sessionStorage 之间有什么区别？](https://www.30secondsofcode.org/blog/s/cookies-local-storage-session)
18. [parseCookie](https://www.30secondsofcode.org/js/s/parse-cookie)（解析HTTP Cookie头字符串，返回所有 Cookie 键值对的对象）
19. [serializeCookie](https://www.30secondsofcode.org/js/s/serialize-cookie)（将 Cookie 键值对序列化为 Set-Cookie 头字符串）
20. [runAsync](https://www.30secondsofcode.org/js/s/run-async)（使用Web Worker在单独的线程中运行一个函数，从而允许长时间运行的函数不会阻塞UI）
21. [renderElement](https://www.30secondsofcode.org/js/s/render-element)（在指定的 DOM 元素中呈现给定的 DOM 树）
22. [serializeForm](https://www.30secondsofcode.org/js/s/serialize-form)（将一组表单元素编码为查询字符串）
23. [Tip: 创建自己的查询选择器](https://www.30secondsofcode.org/blog/s/javascript-query-selector-shorthand)（是否曾经想创建自己的类似jquery的查询选择器的简写？可以这么做）
24. [encodeURI() 和encodeURIComponent() 有什么区别？](https://www.30secondsofcode.org/blog/s/javascript-encodeuri-encodeuricomponent)
25. [getSiblings](https://www.30secondsofcode.org/js/s/get-siblings)（返回一个包含给定元素的所有同级元素的数组）
26. [getImages](https://www.30secondsofcode.org/js/s/get-images)（从一个元素中获取所有图像，并将它们放入数组中）
27. [getAncestors](https://www.30secondsofcode.org/js/s/get-ancestors)（返回从文档根到给定元素之间的所有祖先）
28. [getElementsBiggerThanViewport](https://www.30secondsofcode.org/js/s/get-elements-bigger-than-viewport)（返回宽度大于视口宽度的 HTML 元素数组）
29. [getSelectedText](https://www.30secondsofcode.org/js/s/get-selected-text)（获取当前选择的文本）
30. [nodeListToArray](https://www.30secondsofcode.org/js/s/node-list-to-array)（将 NodeList 转换为数组）
31. [arrayToHTMLList](https://www.30secondsofcode.org/js/s/array-to-html-list)（将给定的数组元素转换为<li>标签，并将它们附加到给定 id 的列表中）
32. [escapeHTML](https://www.30secondsofcode.org/js/s/escape-html)（转义用于HTML的字符串）
33. [unescapeHTML](https://www.30secondsofcode.org/js/s/unescape-html)（转义转义的HTML字符）
34. [Tip: 避免为空链接使用“ javascript：void(0)”](https://www.30secondsofcode.org/blog/s/javascript-void-links)
35. [observeMutations](https://www.30secondsofcode.org/js/s/observe-mutations)（创建一个新的 MutationObserver 并为指定元素上的每个突变提供回调）
36. [getParentsUntil](https://www.30secondsofcode.org/js/s/get-parents-until)（查找元素的所有祖先，直到该元素与指定选择器匹配为止）
37. [Tip: 最小化 DOM 访问](https://www.30secondsofcode.org/blog/s/javascript-store-dom-items)（提高使用DOM时JavaScript代码的性能）
38. [formToObject](https://www.30secondsofcode.org/js/s/form-to-object)（将一组表单元素编码为一个对象）
39. [UUIDGeneratorBrowser](https://www.30secondsofcode.org/js/s/uuid-generator-browser)（在浏览器中生成 UUID）
40. [hashBrowser](https://www.30secondsofcode.org/js/s/hash-browser)（使用 SHA-256 算法为值创建哈希值并返回一个 Promise 对象）
41. [Tip: 使用target =“ \_ blank”时保护用户免受恶意网站的侵害](https://www.30secondsofcode.org/blog/s/javascript-target-blank)（在新选项卡中打开链接会附带一个安全漏洞）
42. [show](https://www.30secondsofcode.org/js/s/show)（显示所有指定的元素）
43. [hide](https://www.30secondsofcode.org/js/s/hide)（隐藏所有指定的元素）
44. [recordAnimationFrames](https://www.30secondsofcode.org/js/s/record-animation-frames)（在每个动画帧上调用提供的回调）
45. [httpGet](https://www.30secondsofcode.org/js/s/http-get)（对传递的URL进行GET请求）
46. [httpPost](https://www.30secondsofcode.org/js/s/http-post)（向传递的网址发出POST请求）
47. [httpPut](https://www.30secondsofcode.org/js/s/http-put)（对传递的网址进行PUT请求）
48. [httpDelete](https://www.30secondsofcode.org/js/s/http-delete)（对传递的网址进行DELETE请求）
49. [httpsRedirect](https://www.30secondsofcode.org/js/s/https-redirect)（如果页面当前位于HTTP中，则将其重定向到HTTPS）
50. [counter](https://www.30secondsofcode.org/js/s/counter)（为指定选择器创建具有指定范围，步长和持续时间的计数器）
51. [scrollToTop](https://www.30secondsofcode.org/js/s/scroll-to-top)（平滑滚动到页面顶部）
52. [getScrollPosition](https://www.30secondsofcode.org/js/s/get-scroll-position)（返回当前页面的滚动位置）
53. [onScrollStop](https://www.30secondsofcode.org/js/s/on-scroll-stop)（用户停止滚动时运行回调）
54. [smoothScroll](https://www.30secondsofcode.org/js/s/smooth-scroll)（平滑地将调用其的元素滚动到浏览器窗口的可见区域）
55. [detectLanguage](https://www.30secondsofcode.org/js/s/detect-language)（检测当前用户的首选语言）
56. [hasClass](https://www.30secondsofcode.org/js/s/has-class)（检查给定元素是否具有指定的类）
57. [addClass](https://www.30secondsofcode.org/js/s/add-class)（将类添加到HTML元素中）
58. [removeClass](https://www.30secondsofcode.org/js/s/remove-class)（从HTML元素中删除类）
59. [toggleClass](https://www.30secondsofcode.org/js/s/toggle-class)（切换HTML元素的类）
60. [prefix](https://www.30secondsofcode.org/js/s/prefix)（自动添加 CSS 浏览器前缀）
61. [injectCSS](https://www.30secondsofcode.org/js/s/inject-css)（将给定的 CSS 代码注入当前文档）
62. [addStyles](https://www.30secondsofcode.org/js/s/add-styles)（将提供的样式添加到给定的元素）
63. [setStyle](https://www.30secondsofcode.org/js/s/set-style)（为指定的HTML元素设置CSS规则的值）
64. [getStyle](https://www.30secondsofcode.org/js/s/get-style)（检索指定元素的CSS规则的值）
65. [getVerticalOffset](https://www.30secondsofcode.org/js/s/get-vertical-offset)（查找从给定元素到文档顶部的距离）
66. [createElement](https://www.30secondsofcode.org/js/s/create-element)（从字符串创建一个元素（不将其附加到文档中）如果给定的字符串包含多个元素，则仅返回第一个元素）
67. [removeElement](https://www.30secondsofcode.org/js/s/remove-element)（从 DOM 中删除一个元素）
68. [detectDeviceType](https://www.30secondsofcode.org/js/s/detect-device-type)（检测页面是在移动设备上还是在桌面上）
69. [fullscreen](https://www.30secondsofcode.org/js/s/fullscreen)（以全屏模式打开或关闭元素）
70. [bottomVisible](https://www.30secondsofcode.org/js/s/bottom-visible)（检查页面底部是否可见）
71. [insertAfter](https://www.30secondsofcode.org/js/s/insert-after)（在指定元素的末尾插入HTML字符串）
72. [insertBefore](https://www.30secondsofcode.org/js/s/insert-before)（在指定元素的开头之前插入HTML字符串）
73. [elementContains](https://www.30secondsofcode.org/js/s/element-contains)（检查父元素是否包含子元素）
74. [elementIsVisibleInViewport](https://www.30secondsofcode.org/js/s/element-is-visible-in-viewport)（检查指定的元素在视口中是否可见）
75. [elementIsFocused](https://www.30secondsofcode.org/js/s/element-is-focused)（检查给定元素是否聚焦）
76. [isBrowserTabFocused](https://www.30secondsofcode.org/js/s/is-browser-tab-focused)（检查页面的浏览器选项卡是否聚焦）
77. [isBrowser](https://www.30secondsofcode.org/js/s/is-browser)（确定当前运行时环境是否是浏览器，以便前端模块可以在服务器（节点）上运行而不会引发错误）
78. [isLocalStorageEnabled](https://www.30secondsofcode.org/js/s/is-local-storage-enabled)（检查是否启用了 localStorage）
79. [isSessionStorageEnabled](https://www.30secondsofcode.org/js/s/is-session-storage-enabled)（检查 sessionStorage 是否启用）

#### 9）颜色

1. [prefersDarkColorScheme](https://www.30secondsofcode.org/js/s/prefers-dark-color-scheme)（检查用户配色方案首选项是否为深色）
2. [prefersLightColorScheme](https://www.30secondsofcode.org/js/s/prefers-light-color-scheme)（检查用户配色方案首选项是否为浅色）
3. [toHSLObject](https://www.30secondsofcode.org/js/s/to-hsl-object)（将 hsl() 颜色字符串转换为具有每种颜色值的对象）
4. [toRGBObject](https://www.30secondsofcode.org/js/s/to-rgb-object)（将 rgb() 颜色字符串转换为具有每种颜色值的对象）
5. [changeLightness](https://www.30secondsofcode.org/js/s/change-lightness)（更改 hsl() 颜色字符串的亮度值）
6. [toHSLArray](https://www.30secondsofcode.org/js/s/to-hsl-array)（将 hsl() 颜色字符串转换为值数组）
7. [toRGBArray](https://www.30secondsofcode.org/js/s/to-rgb-array)（将 rgbl() 颜色字符串转换为值数组）
8. [extendHex](https://www.30secondsofcode.org/js/s/extend-hex)（将 3 位数颜色代码扩展为 6 位数颜色代码）
9. [hexToRGB](https://www.30secondsofcode.org/js/s/hex-to-rgb)（如果提供了 alpha 值，则将颜色代码转换为 rgb() 或 rgba() 字符串）
10. [RGBToHex](https://www.30secondsofcode.org/js/s/rgb-to-hex)（将 RGB 分量的值转换为十六进制颜色代码）
11. [HSBToRGB](https://www.30secondsofcode.org/js/s/hsb-to-rgb)（将 HSB 颜色元组转换为 RGB 格式）
12. [HSLToRGB](https://www.30secondsofcode.org/js/s/hsl-to-rgb)（将 HSL 颜色元组转换为 RGB 格式）
13. [RGBToHSB](https://www.30secondsofcode.org/js/s/rgb-to-hsb)（将 RGB 颜色元组转换为 HSB 格式）
14. [RGBToHSL](https://www.30secondsofcode.org/js/s/rgb-to-hsl)（将 RGB 颜色元组转换为 HSL 格式）

## 数组技巧

1. [any](https://www.30secondsofcode.org/js/s/any)（检查所提供的回调函数是否对集合中的至少一个元素返回 true）
2. [all](https://www.30secondsofcode.org/js/s/all)（检查所提供的回调函数是否对集合中的所有元素返回 true）
3. [allEqual](https://www.30secondsofcode.org/js/s/all-equal)（检查数组中的所有元素是否相等）
4. [allEqualBy](https://www.30secondsofcode.org/js/s/all-equal-by)（根据提供的映射函数检查数组中的所有元素是否相等）
5. [allUnique](https://www.30secondsofcode.org/js/s/all-unique)（检查数组中的所有元素是否唯一）
6. [allUniqueBy](https://www.30secondsofcode.org/js/s/all-unique-by)（根据提供的映射函数检查数组中的所有元素是否唯一）
7. [bifurcate](https://www.30secondsofcode.org/js/s/bifurcate)（根据给定过滤器数组的结果将值分为两组）
8. [bifurcateBy](https://www.30secondsofcode.org/js/s/bifurcate-by)（根据给定过滤功能的结果将值分为两组）
9. [chunk](https://www.30secondsofcode.org/js/s/chunk)（将数组分成指定大小的较小数组）
10. [chunkIntoN](https://www.30secondsofcode.org/js/s/chunk-into-n)（将一个数组分成 n 个较小的数组）
11. [countOccurrences](https://www.30secondsofcode.org/js/s/count-occurrences)（计算数组中某个值的出现次数）
12. [compact](https://www.30secondsofcode.org/js/s/compact)（从数组中删除假值）
13. [flatten](https://www.30secondsofcode.org/js/s/flatten)（将数组扁平化到指定深度）
14. [deepFlatten](https://www.30secondsofcode.org/js/s/deep-flatten)（深度扁平化数组）
15. [difference](https://www.30secondsofcode.org/js/s/difference)（计算两个数组之间的差，并且不过滤重复值）
16. [differenceBy](https://www.30secondsofcode.org/js/s/difference-by)（将提供的函数应用于两个数组的每个元素后，返回两个数组之间的差）
17. [differenceWith](https://www.30secondsofcode.org/js/s/difference-with)（从比较器函数未返回 true 的数组中筛选出所有值）
18. [drop](https://www.30secondsofcode.org/js/s/drop)（创建一个新数组，从左侧删除 n 个元素）
19. [dropRight](https://www.30secondsofcode.org/js/s/drop-right)（创建一个新数组，从右边删除 n 个元素）
20. [dropRightWhile](https://www.30secondsofcode.org/js/s/drop-right-while)（从数组末尾删除元素，直到传递的函数返回 true，返回数组中剩余的元素）
21. [dropWhile](https://www.30secondsofcode.org/js/s/drop-while)（删除数组中的元素，直到传递的函数返回 true，返回数组中剩余的元素）
22. [equals](https://www.30secondsofcode.org/js/s/equals)（在两个值之间进行深度比较以确定它们是否等效）
23. [everyNth](https://www.30secondsofcode.org/js/s/every-nth)（返回数组中的每个第 n 个元素）
24. [filterNonUnique](https://www.30secondsofcode.org/js/s/filter-non-unique)（创建一个滤除非唯一值的数组）
25. [filterNonUniqueBy](https://www.30secondsofcode.org/js/s/filter-non-unique-by)（根据提供的比较器函数，创建一个滤除非唯一值的数组）
26. [filterUnique](https://www.30secondsofcode.org/js/s/filter-unique)（创建一个滤除唯一值的数组）
27. [filterUniqueBy](https://www.30secondsofcode.org/js/s/filter-unique-by)（根据提供的比较器函数，创建一个过滤出唯一值的数组）
28. [findLast](https://www.30secondsofcode.org/js/s/find-last)（查找提供的函数为其返回真的最后一个元素）
29. [findLastIndex](https://www.30secondsofcode.org/js/s/find-last-index)（查找提供的函数为其返回真的最后一个元素的索引）
30. [forEachRight](https://www.30secondsofcode.org/js/s/for-each-right)（从数组的最后一个元素开始，为每个数组元素执行一次提供的函数）
31. [frequencies](https://www.30secondsofcode.org/js/s/frequencies)（创建一个对象，将数组的唯一值作为键，并将其频率作为值）
32. [frozenSet](https://www.30secondsofcode.org/js/s/frozen-set)（创建一个冻结的Set对象）
33. [hasDuplicates](https://www.30secondsofcode.org/js/s/has-duplicates)（检查数组中是否存在重复值）
34. [haveSameContents](https://www.30secondsofcode.org/js/s/have-same-contents)（检查两个数组是否包含相同的元素，而不管顺序如何）
35. [head](https://www.30secondsofcode.org/js/s/head)（返回数组的头）
36. [tail](https://www.30secondsofcode.org/js/s/tail)（返回数组中除第一个元素外的所有元素）
37. [last](https://www.30secondsofcode.org/js/s/last)（返回数组中的最后一个元素）
38. [includesAll](https://www.30secondsofcode.org/js/s/includes-all)（检查值中的所有元素是否都包含在 arr 中）
39. [includesAny](https://www.30secondsofcode.org/js/s/includes-any)（检查 arr 中是否包含至少一个值元素）
40. [indexOfAll](https://www.30secondsofcode.org/js/s/index-of-all)（查找数组中的所有值索引。如果值永远不会出现，则返回一个空数组）
41. [initial](https://www.30secondsofcode.org/js/s/initial)（返回数组中除最后一个元素外的所有元素）
42. [initialize2DArray](https://www.30secondsofcode.org/js/s/initialize2-d-array)（初始化给定宽度，高度和值的 2D 数组）
43. [initializeArrayWithRange](https://www.30secondsofcode.org/js/s/initialize-array-with-range)（初始化一个数组，该数组包含指定范围内的数字，其中开始和结束包含其共同差值步骤）
44. [initializeArrayWithRangeRight](https://www.30secondsofcode.org/js/s/initialize-array-with-range-right)（初始化一个数组，该数组包含指定范围内的数字（反向），其中开始和结束包含其共同差值步骤）
45. [initializeArrayWithValues](https://www.30secondsofcode.org/js/s/initialize-array-with-values)（用指定的值初始化并填充数组）
46. [initializeNDArray](https://www.30secondsofcode.org/js/s/initialize-nd-array)（创建具有给定值的 n 维数组）
47. [insertAt](https://www.30secondsofcode.org/js/s/insert-at)（突变原始数组以在给定索引之后插入给定值）
48. [union](https://www.30secondsofcode.org/js/s/union)（计算两个数组的并集）
49. [unionBy](https://www.30secondsofcode.org/js/s/union-by)（在将提供的函数应用于两个数组的每个元素后，至少返回一次存在于两个数组中的每个数组）
50. [unionWith](https://www.30secondsofcode.org/js/s/union-with)（使用提供的比较器函数，至少返回一次存在于两个数组中的每个元素）
51. [intersection](https://www.30secondsofcode.org/js/s/intersection)（计算两个数组的交集）
52. [intersectionBy](https://www.30secondsofcode.org/js/s/intersection-by)（将提供的函数应用于两个数组的每个元素后，返回两个数组中存在的元素）
53. [intersectionWith](https://www.30secondsofcode.org/js/s/intersection-with)（使用提供的比较器函数，返回两个数组中都存在的元素）
54. [isContainedIn](https://www.30secondsofcode.org/js/s/is-contained-in)（检查第一个数组的元素是否包含在第二个数组中，而不管顺序如何）
55. [isDisjoint](https://www.30secondsofcode.org/js/s/is-disjoint)（检查两个可迭代对象是否不相交（没有通用值））
56. [isSorted](https://www.30secondsofcode.org/js/s/is-sorted)（检查数字数组是否已排序）
57. [join](https://www.30secondsofcode.org/js/s/join)（将数组的所有元素连接到一个字符串中并返回此字符串。使用分隔符和结尾分隔符）
58. [longestItem](https://www.30secondsofcode.org/js/s/longest-item)（接受任意数量的可迭代对象或具有 length 属性的对象，并返回最长的一个）
59. [mergeSortedArrays](https://www.30secondsofcode.org/js/s/merge-sorted-arrays)（将两个排序的数组合并为一个）
60. [mostFrequent](https://www.30secondsofcode.org/js/s/most-frequent)（返回数组中最频繁的元素）
61. [none](https://www.30secondsofcode.org/js/s/none)（检查提供的回调函数是否对集合中的所有元素返回 false）
62. [nthElement](https://www.30secondsofcode.org/js/s/nth-element)（返回数组的第 n 个元素）
63. [offset](https://www.30secondsofcode.org/js/s/offset)（将指定数量的元素移动到数组的末尾）
64. [pull](https://www.30secondsofcode.org/js/s/pull)（突变原始数组以滤除指定的值）
65. [pullAtIndex](https://www.30secondsofcode.org/js/s/pull-at-index)（突变原始数组以滤除指定索引处的值。返回移除的元素）
66. [pullAtValue](https://www.30secondsofcode.org/js/s/pull-at-value)（突变原始数组以滤除指定的值。返回移除的元素）
67. [pullBy](https://www.30secondsofcode.org/js/s/pull-by)（根据给定的迭代器函数，对原始数组进行突变以滤除指定的值）
68. [reducedFilter](https://www.30secondsofcode.org/js/s/reduced-filter)（根据条件过滤对象数组，同时也过滤掉未指定的键）
69. [reduceSuccessive](https://www.30secondsofcode.org/js/s/reduce-successive)（对一个累加器和数组中的每个元素（从左到右）应用一个函数，返回一个连续减小值的数组）
70. [reduceWhich](https://www.30secondsofcode.org/js/s/reduce-which)（应用提供的函数设置比较规则后，返回数组的最小值/最大值）
71. [reject](https://www.30secondsofcode.org/js/s/reject)（根据判定函数过滤数组的值，仅返回判定函数返回 false 的值）
72. [remove](https://www.30secondsofcode.org/js/s/remove)（通过删除给定函数为其返回 false 的元素来对数组进行突变）
73. [sample](https://www.30secondsofcode.org/js/s/sample)（从数组中获取随机元素）
74. [sampleSize](https://www.30secondsofcode.org/js/s/sample-size)（从数组中获取唯一键处的 n 个随机元素，最大为数组的大小）
75. [weightedSample](https://www.30secondsofcode.org/js/s/weighted-sample)（使用提供的权重作为每个元素的概率从数组中获取随机元素）
76. [shank](https://www.30secondsofcode.org/js/s/shank)（具有与 Array.prototype.splice() 相同的功能，但是返回一个新数组，而不是对原始数组进行突变）
77. [shuffle](https://www.30secondsofcode.org/js/s/shuffle)（随机化数组值的顺序，返回一个新数组）
78. [similarity](https://www.30secondsofcode.org/js/s/similarity)（返回同时出现在两个数组中的元素数组）
79. [sortedIndex](https://www.30secondsofcode.org/js/s/sorted-index)（查找应将值插入数组以保持其排序顺序的最低索引）
80. [sortedIndexBy](https://www.30secondsofcode.org/js/s/sorted-index-by)（根据提供的迭代器函数，找到应将值插入数组以保持其排序顺序的最低索引）
81. [sortedLastIndex](https://www.30secondsofcode.org/js/s/sorted-last-index)（查找应将值插入数组以保持其排序顺序的最高索引）
82. [sortedLastIndexBy](https://www.30secondsofcode.org/js/s/sorted-last-index-by)（基于提供的迭代器函数，找到一个值，该值应插入到数组中以保持其排序顺序的最高索引）
83. [stableSort](https://www.30secondsofcode.org/js/s/stable-sort)（对数组执行稳定的排序，当项的值相同时保留项的初始索引）
84. [subSet](https://www.30secondsofcode.org/js/s/sub-set)（检查第一个可迭代对象是否是第二个可迭代对象的子集，不包括重复值）
85. [superSet](https://www.30secondsofcode.org/js/s/super-set)（检查第一个可迭代对象是否是第二个可迭代对象的超集，不包括重复值）
86. [symmetricDifference](https://www.30secondsofcode.org/js/s/symmetric-difference)（返回两个数组之间的对称差，而不过滤掉重复的值）
87. [symmetricDifferenceBy](https://www.30secondsofcode.org/js/s/symmetric-difference-by)（将提供的函数应用于两个数组的每个数组元素后，返回两个数组之间的对称差）
88. [symmetricDifferenceWith](https://www.30secondsofcode.org/js/s/symmetric-difference-with)（使用提供的函数作为比较器，返回两个数组之间的对称差）
89. [take](https://www.30secondsofcode.org/js/s/take)（创建一个数组，其中从开头删除了 n 个元素）
90. [takeRight](https://www.30secondsofcode.org/js/s/take-right)（创建一个从末尾删除 n 个元素的数组）
91. [takeRightUntil](https://www.30secondsofcode.org/js/s/take-right-until)（从数组末尾删除元素，直到传递的函数返回 true。返回移除的元素）
92. [takeRightWhile](https://www.30secondsofcode.org/js/s/take-right-while)（从数组末尾删除元素，直到传递的函数返回 false。返回移除的元素）
93. [takeUntil](https://www.30secondsofcode.org/js/s/take-until)（删除数组中的元素，直到传递的函数返回 true。返回移除的元素）
94. [takeWhile](https://www.30secondsofcode.org/js/s/take-while)（删除数组中的元素，直到传递的函数返回 false。返回移除的元素）
95. [toHash](https://www.30secondsofcode.org/js/s/to-hash)（Reduces a given array-like into a value hash (keyed data store)）
96. [uniqueElements](https://www.30secondsofcode.org/js/s/unique-elements)（查找数组中的所有唯一值）
97. [uniqueElementsBy](https://www.30secondsofcode.org/js/s/unique-elements-by)（根据提供的比较器函数查找数组的所有唯一值）
98. [uniqueElementsByRight](https://www.30secondsofcode.org/js/s/unique-elements-by-right)（从右侧开始，根据提供的比较器函数查找数组的所有唯一值）
99. [uniqueSymmetricDifference](https://www.30secondsofcode.org/js/s/unique-symmetric-difference)（返回两个数组之间的唯一对称差异，不包含两个数组中的重复值）
100. [unzip](https://www.30secondsofcode.org/js/s/unzip)（创建一个数组数组，将 zip 产生的数组中的元素取消分组）
101. [unzipWith](https://www.30secondsofcode.org/js/s/unzip-with)（创建一个元素数组，将 zip 产生的数组中的元素取消分组并应用提供的函数）
102. [without](https://www.30secondsofcode.org/js/s/without)（筛选出具有指定值之一的数组元素）
103. [zip](https://www.30secondsofcode.org/js/s/zip)（创建一个元素数组，根据元素在原始数组中的位置进行分组）
104. [zipObject](https://www.30secondsofcode.org/js/s/zip-object)（将属性与值，给定的有效属性标识符数组和值数组相关联）
105. [zipWith](https://www.30secondsofcode.org/js/s/zip-with)（创建一个元素数组，根据原始数组中的位置进行分组，并使用函数指定应如何组合分组的值）

## 算法

1. [kMeans](https://www.30secondsofcode.org/js/s/k-means)（使用 k-means 聚类算法将给定数据分组为 k 个聚类）
2. [kNearestNeighbors](https://www.30secondsofcode.org/js/s/k-nearest-neighbors)（使用 k 最近邻算法对相对于标记数据集的数据点进行分类）
3. [quickSort](https://www.30secondsofcode.org/js/s/quick-sort)（使用 quicksort 算法对数字数组进行排序）
4. [indexOfSubstrings](https://www.30secondsofcode.org/js/s/index-of-substrings)（查找给定字符串中子字符串的所有索引）
5. [luhnCheck](https://www.30secondsofcode.org/js/s/luhn-check)（Luhn算法的实现，用于验证各种标识号，例如信用卡号，IMEI编号，国家提供商标识号等）
6. [bucketSort](https://www.30secondsofcode.org/js/s/bucket-sort)（使用存储桶排序算法对数字数组进行排序）
7. [heapSort](https://www.30secondsofcode.org/js/s/heapsort)（使用堆排序算法对数字数组进行排序）
8. [selectionSort](https://www.30secondsofcode.org/js/s/selection-sort)（使用选择排序算法对数字数组进行排序）
9. [permutations](https://www.30secondsofcode.org/js/s/permutations)（生成数组元素的所有排列（包含重复项））
10. [caesarCipher](https://www.30secondsofcode.org/js/s/caesar-cipher)（使用 Caesar 密码对给定的字符串进行加密或解密）
11. [primes](https://www.30secondsofcode.org/js/s/primes)（使用 Eratosthenes 筛子产生最多给定数的素数）
12. [countSubstrings](https://www.30secondsofcode.org/js/s/count-substrings)（计算给定字符串中子字符串的出现次数）
13. [euclideanDistance](https://www.30secondsofcode.org/js/s/euclidean-distance)（计算任意数量维度中两点之间的距离）
14. [insertionSort](https://www.30secondsofcode.org/js/s/insertion-sort)（使用插入排序算法对数字数组进行排序）
15. [mergeSort](https://www.30secondsofcode.org/js/s/merge-sort)（使用合并排序算法对数字数组进行排序）
16. [arithmeticProgression](https://www.30secondsofcode.org/js/s/arithmetic-progression)（以等差数列创建一个数字数组，从给定的正整数开始，直至指定的限制）
17. [gcd](https://www.30secondsofcode.org/js/s/gcd)（计算两个或多个数字/数组之间的最大公约数）
18. [bubbleSort](https://www.30secondsofcode.org/js/s/bubble-sort)（使用冒泡排序算法对数字数组进行排序）
19. [geometricProgression](https://www.30secondsofcode.org/js/s/geometric-progression)（初始化一个数组，该数组包含指定范围内的数字，其中开始和结束均包括在内，并且两项之间的比率为 step。如果 step 等于1，则返回错误）
20. [levenshteinDistance](https://www.30secondsofcode.org/js/s/levenshtein-distance)（使用 Levenshtein 距离算法计算两个字符串之间的差异）
21. [powerset](https://www.30secondsofcode.org/js/s/powerset)（返回给定数字数组的幂集）
22. [binarySearch](https://www.30secondsofcode.org/js/s/binary-search)（使用二进制搜索算法在排序数组中查找给定元素的索引）
23. [fibonacci](https://www.30secondsofcode.org/js/s/fibonacci)（生成一个包含斐波那契序列的数组，直到第n个项）
24. [linearSearch](https://www.30secondsofcode.org/js/s/linear-search)（使用线性搜索算法查找数组中给定元素的第一个索引）
25. [primeFactors](https://www.30secondsofcode.org/js/s/prime-factors)（使用试除法找到给定数的素数）
26. [vectorDistance](https://www.30secondsofcode.org/js/s/vector-distance)（计算两个向量之间的距离）
27. [lcm](https://www.30secondsofcode.org/js/s/lcm)（计算两个或多个数字的最小公倍数）
28. [hammingDistance](https://www.30secondsofcode.org/js/s/hamming-distance)（计算两个值之间的汉明距离）
29. [binomialCoefficient](https://www.30secondsofcode.org/js/s/binomial-coefficient)（计算从 n 个项目中选择 k 个项目（无重复且无顺序）的方式数量）
30. [factorial](https://www.30secondsofcode.org/js/s/factorial)（计算数字的阶乘）
31. [distance](https://www.30secondsofcode.org/js/s/distance)（计算两点之间的距离）
32. [isPrime](https://www.30secondsofcode.org/js/s/is-prime)（检查提供的整数是否为质数）

## CSS

#### 1）动画

1. [图像悬停菜单](https://www.30secondsofcode.org/css/s/image-hover-menu)（悬停图像时显示菜单覆盖）
2. [悬停图像叠加](https://www.30secondsofcode.org/css/s/image-overlay-hover)（在悬停时显示图像叠加效果）
3. [交错动画](https://www.30secondsofcode.org/css/s/staggered-animation)（为列表的元素创建交错动画）
4. [弹跳装载机](https://www.30secondsofcode.org/css/s/bouncing-loader)（创建一个弹跳加载器动画）
5. [高度过渡](https://www.30secondsofcode.org/css/s/height-transition)（当元素的高度未知时，将其高度从0转换为自动）
6. [图像在悬停时旋转](https://www.30secondsofcode.org/css/s/image-hover-rotate)（在悬停时为图像创建旋转效果）
7. [旋转卡](https://www.30secondsofcode.org/css/s/rotating-card)（创建一个在悬停时旋转的双面卡片）
8. [Tip: CSS缓动变量](https://www.30secondsofcode.org/blog/s/css-easing-variables)（了解如何使用缓动函数的cubic-bezier()类并创建引人注目的精美动画）
9. [Tip: CSS过渡的完美持续时间](https://www.30secondsofcode.org/blog/s/perfect-css-transition-duration)（了解如何在用户与页面上的元素进行交互时使CSS过渡完美）
10. [悬停下划线动画](https://www.30secondsofcode.org/css/s/hover-underline-animation)（当文本悬停在上方时，创建动画下划线效果）
11. [按钮边框动画](https://www.30secondsofcode.org/css/s/button-border-animation)（在悬停时创建边框动画）
12. [脉冲装载机](https://www.30secondsofcode.org/css/s/pulse-loader)（使用 animation-delay 属性创建脉冲效果加载器动画）
13. [甜甜圈加载器](https://www.30secondsofcode.org/css/s/donut-spinner)（创建一个甜甜圈加载器，可用于指示内容的加载）

#### 2）交互

1. [鼠标光标渐变跟踪](https://www.30secondsofcode.org/css/s/mouse-cursor-gradient-tracking)（悬停效果，其中渐变跟随鼠标光标）
2. [水平滚动捕捉](https://www.30secondsofcode.org/css/s/horizontal-scroll-snap)（创建一个可水平滚动的容器，在滚动时将捕捉到元素）
3. [垂直滚动捕捉](https://www.30secondsofcode.org/css/s/vertical-scroll-snap)（创建一个可垂直滚动的容器，在滚动时将捕捉元素）
4. [带前缀输入](https://www.30secondsofcode.org/css/s/input-with-prefix)（创建带有视觉，不可编辑前缀的输入）
5. [专注于](https://www.30secondsofcode.org/css/s/focus-within)（如果窗体的任何子项都处于焦点状态，则更改窗体的外观）
6. [汉堡按钮](https://www.30secondsofcode.org/css/s/hamburger-button)（显示一个汉堡菜单，该菜单在悬停时转换为十字按钮）
7. [弹出菜单](https://www.30secondsofcode.org/css/s/popout-menu)（显示悬停/焦点上的交互式弹出菜单）
8. [拨动开关](https://www.30secondsofcode.org/css/s/toggle-switch)（仅使用CSS创建切换开关）
9. [淡出兄弟元素](https://www.30secondsofcode.org/css/s/sibling-fade)（淡出悬停位置的兄弟元素）
10. [禁用选择](https://www.30secondsofcode.org/css/s/disable-selection)（使内容无法选择）

#### 3）布局

1. [响应式图像拼接](https://www.30secondsofcode.org/css/s/image-mosaic)（创建响应式图像马赛克）
2. [什么是CSS变量？在哪里可以使用它们？](https://www.30secondsofcode.org/blog/s/css-variables)
3. [Flexbox 备忘单](https://www.30secondsofcode.org/blog/s/flexbox-cheatsheet)（如果您发现自己不断查找语法或其工作方式，那么您只需准备此便捷的备忘单）
4. [屏幕外](https://www.30secondsofcode.org/css/s/offscreen)（在视觉和位置上完全隐藏元素，同时仍允许对其进行访问）
5. [3格布局](https://www.30secondsofcode.org/css/s/tile-layout-using-inline-block)（使用 display:inline-block 水平对齐项目，以创建3格布局）
6. [Clearfix](https://www.30secondsofcode.org/css/s/clearfix)（确保元素自我清除其子级）
7. [砌体布局](https://www.30secondsofcode.org/css/s/masonry-layout)（创建砖石风格的布局，这在处理图像时特别有用）
8. [截断多行文字](https://www.30secondsofcode.org/css/s/truncate-text-multiline)（截断长于一行的文本）
9. [使图像适合容器](https://www.30secondsofcode.org/css/s/fit-image-in-container)（在保持其宽高比的同时，将图像适当放置在其容器内的位置）
10. [CSS单位备忘单](https://www.30secondsofcode.org/blog/s/css-units-cheatsheet)（了解有关CSS单位的所有信息）
11. [带有侧边栏的响应式布局](https://www.30secondsofcode.org/css/s/responsive-layout-sidebar)（创建带有内容区域和侧边栏的自适应布局）
12. [box-sizing 重置](https://www.30secondsofcode.org/css/s/box-sizing-reset)（重置盒模型，以使宽度和高度不受边框或填充的影响）
13. [分布均匀的子元素](https://www.30secondsofcode.org/css/s/evenly-distributed-children)（将子元素平均分配到父元素中）
14. [全屏宽度的图像](https://www.30secondsofcode.org/css/s/full-width)
15. [截断文字](https://www.30secondsofcode.org/css/s/truncate-text)（截断超过一行的文本，并在末尾添加省略号（…））
16. [恒定的宽高比](https://www.30secondsofcode.org/css/s/constant-width-to-height-ratio)（确保宽度可变的元素将保留比例的高度值）

#### 4）视觉

1. [带有浮动标题的列表](https://www.30secondsofcode.org/css/s/floating-list-titles)
2. [在悬停时显示其他内容](https://www.30secondsofcode.org/css/s/hover-additional-content)（创建一个卡，在悬停时显示其他内容）
3. [导航列表项的悬停和焦点效果](https://www.30secondsofcode.org/css/s/navigation-list-item-hover-and-focus-effect)（使用 CSS 变形为导航项创建自定义悬停和焦点效果）
4. [带有粘滞标题的列表](https://www.30secondsofcode.org/css/s/sticky-list-titles)
5. [25个 CSS 渐变](https://www.30secondsofcode.org/blog/s/25-css-gradients)
6. [如何为暗黑模式创建自定义响应图标？](https://www.30secondsofcode.org/blog/s/responsive-favicon-dark-mode)
7. [全屏](https://www.30secondsofcode.org/css/s/fullscreen)（在全屏模式下将样式应用于元素）
8. [溢出滚动渐变](https://www.30secondsofcode.org/css/s/overflow-scroll-gradient)（向溢出元素添加渐变，以更好地指示还有更多内容要滚动）
9. [图片文字叠加](https://www.30secondsofcode.org/css/s/text-backdrop-overlay)（使用覆盖图在图像上方显示文本）
10. [动态阴影](https://www.30secondsofcode.org/css/s/dynamic-shadow)（根据元素本身的颜色创建类似于 box-shadow 的阴影）
11. [漂亮的文本下划线](https://www.30secondsofcode.org/css/s/pretty-text-underline)
12. [蚀刻文字](https://www.30secondsofcode.org/css/s/etched-text)（使文本看起来被“蚀刻”或雕刻到背景中）
13. [系统字体](https://www.30secondsofcode.org/css/s/system-font-stack)（使用操作系统的本机字体来接近本机应用程序的感觉）
14. [斑马条纹列表](https://www.30secondsofcode.org/css/s/zebra-striped-list)（创建具有交替背景颜色的条纹列表）
15. [自定义滚动条](https://www.30secondsofcode.org/css/s/custom-scrollbar)
16. [顶部三角形的边框](https://www.30secondsofcode.org/css/s/border-with-top-triangle)（创建一个内容容器，顶部带有一个三角形）
17. [计数器](https://www.30secondsofcode.org/css/s/counter)（创建一个自定义列表计数器来说明嵌套列表元素）
18. [棋盘背景图案](https://www.30secondsofcode.org/css/s/checkerboard-pattern)
19. [Drop cap](https://www.30secondsofcode.org/css/s/drop-cap)（使第一段的第一个字母大于文本的其余部分）
20. [圆点背景图案](https://www.30secondsofcode.org/css/s/polka-dot-pattern)
21. [曲折背景图案](https://www.30secondsofcode.org/css/s/zig-zag-pattern)
22. [自定义文字选择](https://www.30secondsofcode.org/css/s/custom-text-selection)（更改文本选择的样式）
23. [条纹背景图案](https://www.30secondsofcode.org/css/s/stripes-pattern)
24. [渐变文字](https://www.30secondsofcode.org/css/s/gradient-text)
25. [圆](https://www.30secondsofcode.org/css/s/circle)（使用纯CSS创建圆形）
26. [三角形](https://www.30secondsofcode.org/css/s/triangle)（使用纯CSS创建三角形）
27. [重置所有样式](https://www.30secondsofcode.org/css/s/reset-all-styles)（仅使用一个属性将所有样式重置为默认值）

#### 5）居中

1. [Flexbox](https://www.30secondsofcode.org/css/s/flexbox-centering)（使用 Flexbox 将子元素在父元素中水平和垂直居中）
2. [Grid](https://www.30secondsofcode.org/css/s/grid-centering)（使用网格将子元素在父元素内水平和垂直居中）
3. [Transform](https://www.30secondsofcode.org/css/s/transform-centering)（使用CSS transform将子元素在其父元素内垂直和水平居中）
4. [Table](https://www.30secondsofcode.org/css/s/display-table-centering)（使用 display:table 将子元素在其父元素内垂直和水平居中）

## Node.js

1. [hashNode](https://www.30secondsofcode.org/js/s/hash-node)（使用SHA-256算法为值创建哈希并返回一个 Promise 对象）
2. [readFileLines](https://www.30secondsofcode.org/js/s/read-file-lines)（返回指定文件中的行数组）
3. [JSONToFile](https://www.30secondsofcode.org/js/s/json-to-file)（将JSON对象写入文件）
4. [Tip: 使用Chrome开发人员工具调试 Node.js](https://www.30secondsofcode.org/blog/s/nodejs-chrome-debugging)
5. [colorize](https://www.30secondsofcode.org/js/s/colorize)（在文本中添加特殊字符以在控制台中以彩色打印（与console.log()组合））
6. [btoa](https://www.30secondsofcode.org/js/s/btoa)（从String对象创建一个base-64编码的ASCII字符串，其中字符串中的每个字符都被视为二进制数据的字节）
7. [untildify](https://www.30secondsofcode.org/js/s/untildify)（将波浪号路径转换为绝对路径）
8. [hasFlags](https://www.30secondsofcode.org/js/s/has-flags)（检查当前进程的参数是否包含指定的标志）
9. [isTravisCI](https://www.30secondsofcode.org/js/s/is-travis-ci)（检查当前环境是否为Travis CI）
10. [atob](https://www.30secondsofcode.org/js/s/atob)（解码已使用base-64编码的数据字符串）
11. [requireUncached](https://www.30secondsofcode.org/js/s/require-uncached)（从缓存中删除模块后加载模块（如果存在））
12. [createDirIfNotExists](https://www.30secondsofcode.org/js/s/create-dir-if-not-exists)（创建目录（如果不存在））
13. [isDuplexStream](https://www.30secondsofcode.org/js/s/is-duplex-stream)（检查给定的参数是否为双工（可读和可写）流）
14. [isWritableStream](https://www.30secondsofcode.org/js/s/is-writable-stream)（检查给定的参数是否为可写流）
15. [isReadableStream](https://www.30secondsofcode.org/js/s/is-readable-stream)（检查给定参数是否为可读流）
16. [isNode](https://www.30secondsofcode.org/js/s/is-node)（确定当前运行时环境是否为Node.js）
17. [isStream](https://www.30secondsofcode.org/js/s/is-stream)（检查给定参数是否为流）

## React

#### 1）Hooks

1. [usePersistedState](https://www.30secondsofcode.org/react/s/use-persisted-state)（返回持久存储在localStorage中的状态值，以及用于更新它的函数）
2. [useSSR](https://www.30secondsofcode.org/react/s/use-ssr)（检查代码是否在浏览器或服务器上运行）
3. [useAsync](https://www.30secondsofcode.org/react/s/use-async)（处理异步调用）
4. [useNavigatorOnLine](https://www.30secondsofcode.org/react/s/use-navigator-on-line)（检查客户端在线还是离线）
5. [useFetch](https://www.30secondsofcode.org/react/s/use-fetch)（以声明的方式实现 fetch）
6. [useUnload](https://www.30secondsofcode.org/react/s/use-unload)（处理beforeunload窗口事件）
7. [useDebounce](https://www.30secondsofcode.org/react/s/use-debounce)（Debounce 给定的值）
8. [useClickInside](https://www.30secondsofcode.org/react/s/use-click-inside)（处理在包装的组件内部单击的事件）
9. [useClickOutside](https://www.30secondsofcode.org/react/s/use-click-outside)（处理在包装的组件外部单击的事件）
10. [useMediaQuery](https://www.30secondsofcode.org/react/s/use-media-query)（检查当前环境是否匹配给定的媒体查询并返回适当的值）
11. [useCopyToClipboard](https://www.30secondsofcode.org/react/s/use-copy-to-clipboard)（将给定的文本复制到剪贴板）
12. [useTimeout](https://www.30secondsofcode.org/react/s/use-timeout)（以声明的方式实现 setTimeout）
13. [useComponentDidMount](https://www.30secondsofcode.org/react/s/use-component-did-mount)（挂载组件后立即执行回调）
14. [useComponentWillUnmount](https://www.30secondsofcode.org/react/s/use-component-will-unmount)（在卸载和销毁组件之前立即执行回调）
15. [useInterval](https://www.30secondsofcode.org/react/s/use-interval)（以声明的方式实现 setInterval）
16. [usePrevious](https://www.30secondsofcode.org/react/s/use-previous)（存储先前的 state 或 props）
17. [useToggler](https://www.30secondsofcode.org/react/s/use-toggler)（提供一个布尔状态变量，可以在其两个状态之间切换）

#### 2）组件

1. [TagInput](https://www.30secondsofcode.org/react/s/tag-input)（标签输入字段）
2. [MultiselectCheckbox](https://www.30secondsofcode.org/react/s/multiselect-checkbox)（渲染一个复选框列表，该列表使用回调函数将其选定的一个或多个值传递给父组件）
3. [LimitedWordTextarea](https://www.30secondsofcode.org/react/s/limited-word-textarea)（具有字数限制的 textarea 组件）
4. [Modal](https://www.30secondsofcode.org/react/s/modal)（渲染可通过事件控制的 Modal 组件）
5. [Tabs](https://www.30secondsofcode.org/react/s/tabs)（选项卡式菜单和视图组件）
6. [Accordion](https://www.30secondsofcode.org/react/s/accordion)（具有多个可折叠内容元素的手风琴菜单）
7. [TreeView](https://www.30secondsofcode.org/react/s/tree-view)（可折叠内容的JSON对象或数组的树状图）
8. [RippleButton](https://www.30secondsofcode.org/react/s/ripple-button)（渲染一个按钮，当单击该按钮时可以对波纹效果进行动画处理）
9. [Alert](https://www.30secondsofcode.org/react/s/alert)（使用 prop 类型呈现警报组件）
10. [CountDown](https://www.30secondsofcode.org/react/s/count-down)（倒计时组件，该计时器在到达零时显示一条消息）
11. [FileDrop](https://www.30secondsofcode.org/react/s/file-drop)（文件拖放组件）
12. [StarRating](https://www.30secondsofcode.org/react/s/star-rating)（星级评分组件）
13. [MappedTable](https://www.30secondsofcode.org/react/s/mapped-table)（渲染具有从对象数组和属性名称列表动态创建的行的表）
14. [Tip: React 有条件的 className、空字符串和 null](https://www.30secondsofcode.org/blog/s/react-conditional-classname)
15. [Carousel](https://www.30secondsofcode.org/react/s/carousel)（轮播组件）
16. [如何在React中设置选择输入的值？](https://www.30secondsofcode.org/blog/s/react-selected-option)
17. [PropTypes.objectOf() VS PropTypes.shape()](https://www.30secondsofcode.org/blog/s/react-proptypes-objectof-vs-shape)
18. [Select](https://www.30secondsofcode.org/react/s/select)（一个不受控制的<select>元素，该元素使用回调函数将其值传递给父组件）
19. [Toggle](https://www.30secondsofcode.org/react/s/toggle)（切换组件）
20. [LimitedTextarea](https://www.30secondsofcode.org/react/s/limited-textarea)（具有字符数限制的textarea组件）
21. [Loader](https://www.30secondsofcode.org/react/s/loader)（渲染旋转加载程序组件）
22. [Tooltip](https://www.30secondsofcode.org/react/s/tooltip)（工具提示组件）
23. [AutoLink](https://www.30secondsofcode.org/react/s/auto-link)（将字符串呈现为纯文本，并将URL转换为适当的链接元素）
24. [Collapse](https://www.30secondsofcode.org/react/s/collapse)（渲染具有可折叠内容的组件）
25. [DataList](https://www.30secondsofcode.org/react/s/data-list)（从数组中渲染元素列表）
26. [ControlledInput](https://www.30secondsofcode.org/react/s/controlled-input)（呈现一个受控的<input>元素，该元素使用回调函数来通知其父组件有关值更新的信息）
27. [UncontrolledInput](https://www.30secondsofcode.org/react/s/uncontrolled-input)（呈现一个不受控制的<input>元素，该元素使用回调函数来通知其父组件有关值更新的信息）
28. [Slider](https://www.30secondsofcode.org/react/s/slider)（渲染一个不受控制的范围输入元素，该元素使用回调函数将其值传递给父组件）
29. [DataTable](https://www.30secondsofcode.org/react/s/data-table)（可动态创建行的表格）
30. [PasswordRevealer](https://www.30secondsofcode.org/react/s/password-revealer)（使用显示按钮呈现密码输入字段）
31. [TextArea](https://www.30secondsofcode.org/react/s/text-area)（呈现一个不受控制的<textarea>元素，该元素使用回调函数将其值传递给父组件）
32. [Mailto](https://www.30secondsofcode.org/react/s/mailto)（一个设置为发送电子邮件的链接（mailto：链接））
33. [Callto](https://www.30secondsofcode.org/react/s/callto)（呈现一个格式设置为呼叫电话号码的链接（电话：链接））

#### 3）渲染

1. [React 渲染基础](https://www.30secondsofcode.org/blog/s/react-rendering-basics)（深入了解React的渲染过程，并了解流行的JavaScript框架背后的基础知识）
2. [React 渲染优化](https://www.30secondsofcode.org/blog/s/react-rendering-optimization)（深入研究React的渲染过程，并了解如何进行小而强大的调整以优化性能）
3. [React 渲染状态](https://www.30secondsofcode.org/blog/s/react-rendering-state)（深入研究React的渲染过程并了解Context API和Redux在其中的作用）

#### 4）测试

1. [一种测试有状态 React 组件的方法](https://www.30secondsofcode.org/blog/s/testing-stateful-ui-components)
2. [测试使用React Testing库异步更新的 React 组件](https://www.30secondsofcode.org/blog/s/testing-async-react-components)
3. [使用React Testing Library测试Redux连接的组件](https://www.30secondsofcode.org/blog/s/testing-redux-connected-components)