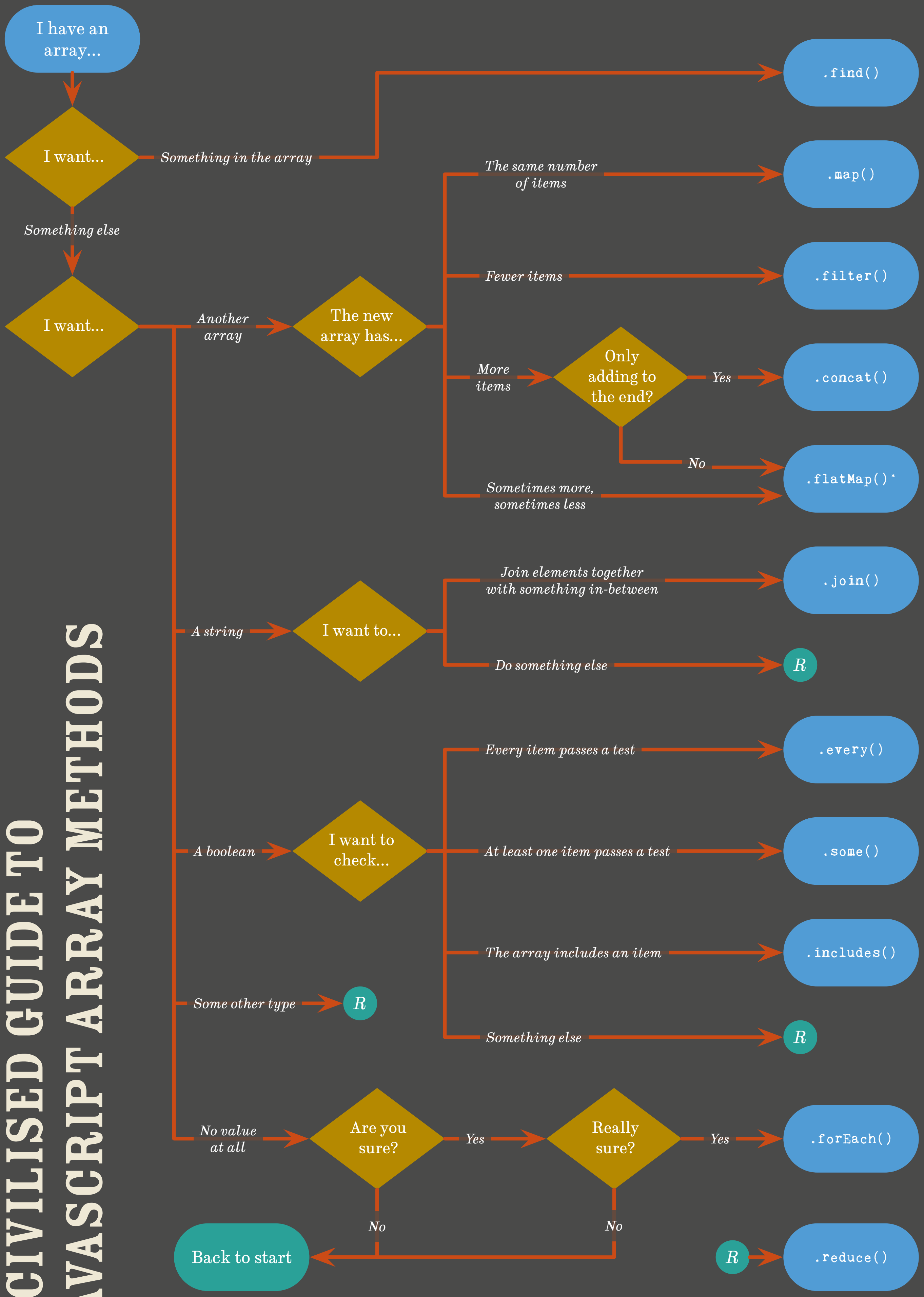
A CIVILISED GUIDE TO JAVASCRIPT ARRAY METHODS



https://jrsinclair.com

SAMPLE DATA

This data is used across all the examples for each array method.

```
const heroes = [
    {name: 'Hulk', strength: 90000},
    {name: 'Spider-Man', strength: 25000},
    {name: 'Hawk Eye', strength: 136},
    {name: 'Thor', strength: 100000},
    {name: 'Black Widow', strength: 136},
    {name: 'Vision', strength: 5000},
    {name: 'Scarlet Witch', strength: 60},
    {name: 'Mystique', strength: 120},
    {name: 'Namora', strength: 75000},
    {name: 'Captain America', strength: 362},
    {name: 'Deadpool', strength: 1814},
    {name: 'Black Panther', strength: 1814},
];
```

.find()

The `.find()` method will return the first element in the array that matches a test you provide.

EXAMPLE:

```
function isHulk(hero) {
  return hero.name === 'Hulk';
}
const hulk = heroes.find(isHulk);
```

.map()

The `.map()` method will apply a given function to every item in your array and give you a new array with those values.

EXAMPLE:

```
function getName(hero) {
  return hero.name;
}
const names = heroes.map(getName);
```

.filter()

The `.filter()` method takes your array and removes items that don't pass a test you give it.

EXAMPLE:

```
function strong(hero) {
  return hero.strength >= 200;
}
const tuff = heroes.filter(strong);
```

.concat()

The `.concat()` method adds new items to the end of your array.

EXAMPLE:

```
const extras = [
  {name: 'Cyclops', strength: 136},
  {name: 'Gambit', strength: 136},
];
const more = heroes.concat(extras);
```

.flatMap()

This method is only a proposal, so it's not available everywhere. You pass it a function that returns an array and it will squish all the results together into a flat array.

EXAMPLE:

```
function space(hero, i) {
  return ((i > 0) && (i % 5 === 0))
    ? ['<hr/>', hero.name]
    : [hero.name];
}
const list = heroes.flatMap(space);
```

.join()

The `.join()` method will insert a given string between each item, and return a joined-up string.

EXAMPLE:

```
function getName(hero) {
  return hero.name;
}
const list = heroes
    .map(getName)
    .join('\n');
```

.every()

The `.every()` method checks that every single item in your array matches some criteria.

EXAMPLE:

```
function strong(hero) {
  return hero.strength >= 200;
}
const tuff = heroes.every(strong);
```

.some()

The `.some()` method checks that at least one item in your array matches some criteria.

EXAMPLE:

```
function isHulk(hero) {
  return hero.name === 'Hulk';
}
const hulkIn = heroes.some(isHulk);
```

.includes()

The `.includes()` method checks that at least one item in your array matches some criteria.

EXAMPLE:

```
function getName(hero) {
  return hero.name;
}
const hulkIn = heroes
    .map(getName)
    .includes('Hulk');
```

.reduce()

The `.reduce()` method is the most flexible array iterator. It processes each item of the array and lets you modify a value as you go.

EXAMPLE:

```
function sumStrength(total, hero) {
  return total + hero.stength;
}
const totalStength = heroes.reduce(
  sumStrength,
  0
);
```

.forEach()

The `.forEach()` method applies a given function to every element in the array. It doesn't return a value though. So, by definition, it's only useful for side effects.

EXAMPLE:

```
function logHero(h) {
  console.log(
    'Name: ' + h.name
    + '\nStrength: ' + h.strength
  );
}
heroes.forEach(logHero);
```

A CIVILISED GUIDE TO JAVASCRIPT ARRAY METHODS