JavaScript Static

Learning Objectives

- Understand what static means in the context of classes
- Understand how to use static properties and methods in JavaScript

Material

Static

Sometimes, it makes sense for methods and properties to belong to the class itself rather than a particular instance. For example, let's imagine we are creating a class system to handle cars and need a function that converts miles-per-gallon to kilometers-per-litre. We could make this a normal method but then we'd have to instantiate a specific car in order to perform our calculation:

```
class Car {
    constructor(make, model) {
        this.make = make;
        this.model = model;
    }

    mpgToKmpl(mpg) {
        return mpg / 2.352;
    }
}

const car1 = new Car("Toyota", "Yaris"); // awkward
console.log(car1.mpgToKmpl(20)); // 8.5034
```

Whenever we have a method which doesn't use any properties on the instance, it's usually a good idea to make it static:

```
class Car {
    constructor(make, model) {
        this.make = make;
        this.model = model;
    }

    static mpgToKmpl(mpg) {
```

```
        return mpg / 2.352;
    }
}

console.log(Car.mpgToKmpl(20)); // 8.5034
```

Notice how we call the function directly on the class itself.

We can also create static properties:

```
class Car {
    static definition = "a four-wheeled road vehicle.";

    constructor(make, model) {
        this.make = make;
        this.model = model;
    }

    static mpgToKmpl(mpg) {
        return mpg / 2.352;
    }
}

console.log(Car.definition); // a four-wheeled road vehicle.
```