MISSION READY

DARE TO
DEVELOP

Data Types - Booleans, Objects

Reuben Simpson

Boolean

- Booleans are the logical values, either true or false. They are the result of logical comparisons.
- Boolean conditionals are often used to decide which sections of code to execute (such as in if statements) or repeat (such as in for loops).

```
if (Boolean_conditional) {
  console.log("boolean conditional resolved to true");
}
else {
  console.log("boolean conditional resolved to false");
}
```

Truthy and Falsy Values

- Falsy values are not exactly false, but evaluate to false when converted to a Boolean
- Truthy values are not exactly true, but evaluate to true when converted to a Boolean
- ***All values are truthy*** unless they are defined as falsy. That is, all values are truthy except false , 0, "" , null , undefined , and NaN

```
console.log(Boolean(false));
console.log(Boolean(null));
console.log(Boolean(undefined));
console.log(Boolean(0));
console.log(Boolean(NaN));
```

Boolean basics

- In Boolean logic, a statement can have two values, true or false
  - Example 0 - It is raining today.
- Boolean logic evaluates a whole statement to see whether it is either true or false.
  - Example 1 - It is raining today **AND** my feet are getting wet
  - Example 2 - It is raining today **OR** my feet are getting wet

Are both statements true?

Is either statement true?

Boolean basics continued

```
const b1 = !false; // true
const b2 = true && false; // false
const b3 = true || false; // true
const b4 = 123 === "456"; // false
const b5 = 1.23 === 123e-2; // true
```

NOT

A	!A
T	F
F	T

This toggles a statement from true to false or from false to true.

AND

A	B	A && B
T	T	T
T	F	F
F	T	F
F	F	F

True when both elements are true.

OR

A	B	A || B
T	T	T
T	F	T
F	T	T
F	F	F

True when at least one of the elements is true.

Boolean basics continued

- As in mathematics, the bit that's between the brackets () is evaluated first.
- BEDMAS (Brackets, Exponents, Division, Multiplication, Addition, Subtraction)
- What is the output of the following?

```
let x = 4,
  y = 2,
  z = 0;
if (x == 4 && (!(y == 1) || z == 0)) {
  console.log(true);
} else {
  console.log(false);
}
```

Exercise 1

Note down what the following statements will return. Try to figure this out *before* putting the commands in the console.

```
2 == "2";
2 === 2;
10 % 3;
10 % 3 === 1;
true && false;
false || true;
true || false;
```

https://www.rithmschool.com/courses/javascript/introduction-to-javascript-boolean-exercises

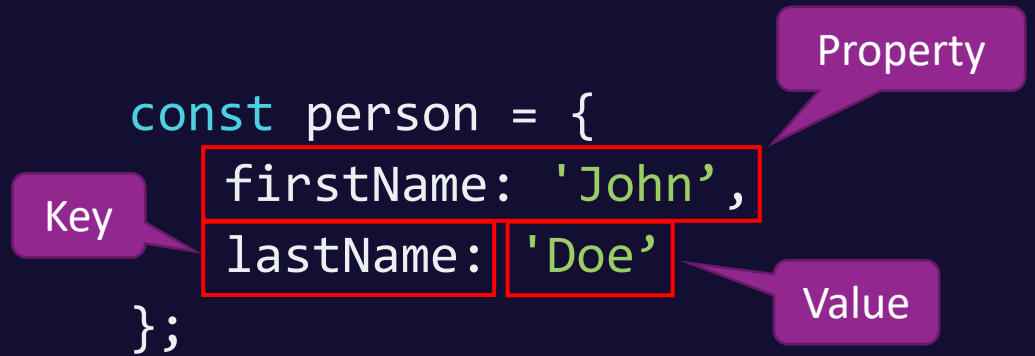
Non-Primitive Data Types

Non-Primitive Data-Type	
1. Object	Used for denoting complex data structure with a collection of properties and methods
1a) Array (a type of Object)	A data structure whereby you can store a collection of elements

- All JavaScript values, except primitives, are objects.

Objects

- Objects are an ***unordered*** collection of **key/value pairs**, where the
  - **Keys** are usually strings
  - **Values** can be any type, even other objects.
- Objects are defined by the list of key: value pairs, comma-separated and enclosed by curly braces.

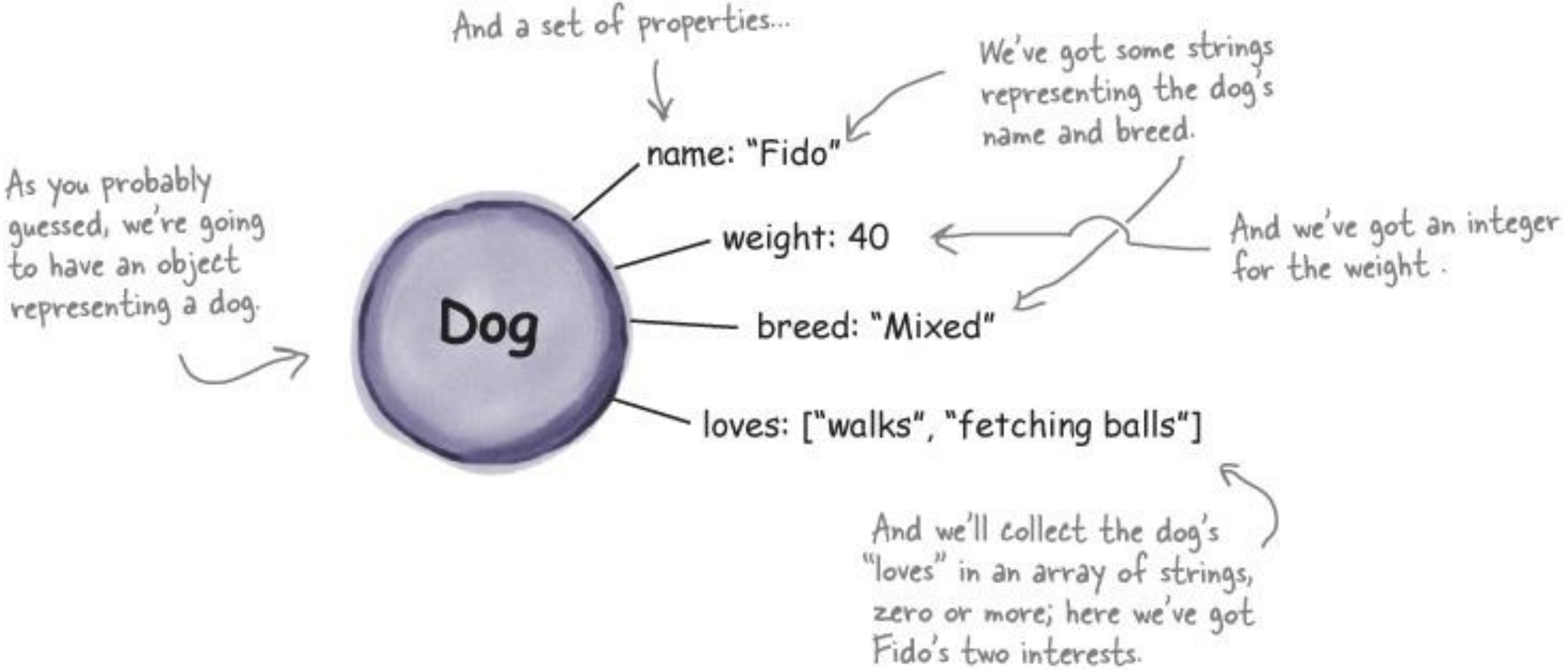


```
const person = {
   firstName: 'John',
   lastName: 'Doe'
};
```

- The person object has two properties firstName and lastName with the corresponding values 'John' and 'Doe'.
- When an object has multiple properties, you use a comma (,) to separate them like the above example.

Note: *Spaces and line breaks are not important. An object definition can span multiple lines.*


Most dogs have names, like Fido here.
All dogs have a weight
All dogs have a list of activities they enjoy, like walking and fetching balls.
And a breed. In this case we'd call Fido a mixed breed.
Dog


And a set of properties...
We've got some strings representing the dog's name and breed.
name: "Fido"
As you probably guessed, we're going to have an object representing a dog.
weight: 40
And we've got an integer for the weight.
Dog
breed: "Mixed"
loves: ["walks", "fetching balls"]
And we'll collect the dog's "loves" in an array of strings, zero or more; here we've got Fido's two interests.

We're going to assign our object to the variable fido.

Start an object with just the left curly brace, then all the properties are going to go inside.

Notice that each property is separated by a comma. NOT a semicolon!

```
const fido = {
      name: "Fido",
      weight: 40,
      breed: "Mixed",
      loves: ["walks", "fetching balls"]
};
```

This object has four properties, name, weight, breed and loves.

Notice that the value of weight is a number, 40, and the values of breed and name are strings.

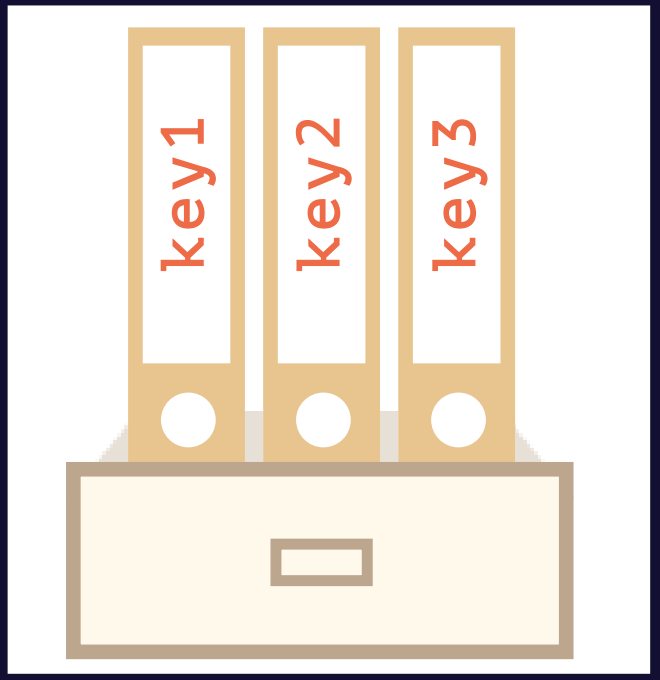
And of course we have an array to hold the dog's loves.

Exercise 2

1. Create a *cat* object containing a first name and last name property, where the values are strings.

Accessing object properties

- **Property accessors** provide access to an object's properties by using the dot notation or the bracket notation.
- Specifically, values can be accessed from objects in two ways:
  - Using the dot operator
  - Using the square brackets



Dot Operator

```
objectName.propertyName
```

The dot notation can be used to access the property of an object.

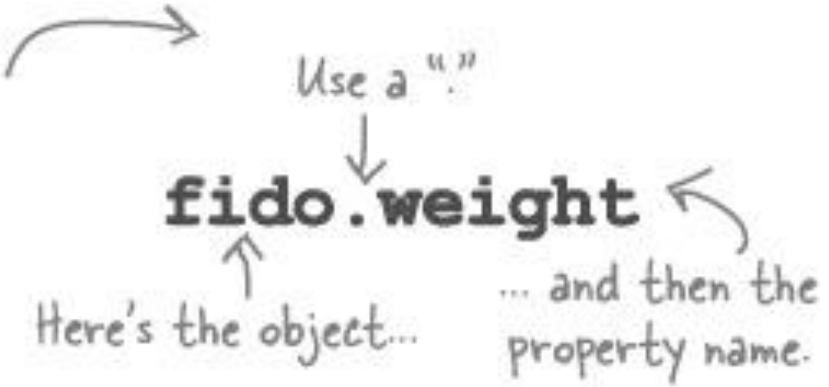
For example, to access the firstName property of the cat object, you use the following expression:

```
cat.firstName
```

*An expression is **any valid set of literals, variables, operators, and expressions that evaluates to a single value**. The value may be a number, a string, or a logical value.*

```
if (fido.weight > 25) {
    alert("WOOF");
} else {
    alert("yip");
}
```

Use the object along with a "." and a property name to access the value of that property.



Exercise 3

1. Add a favouriteColour property to your *cat* object
2. Log a string to the console: "This is firstName lastName, and their favourite colour is favouriteColour." Use the dot operator.

Square bracket notation - [ ]

The square brackets ***property accessor*** has the following syntax.

```
objectName["propertyName"];
```
string

To access the value of an object's property via the array-like/square bracket notation, we use:

```
cat["firstName"];
```

```
const breed = fido["breed"];
  if (breed == "mixed") {
      alert("Best in show");
  }
```

Use the object along with the property name wrapped in quotes and in brackets to access the value of that property.

Now we use [ ] around the property name.

```
fido["weight"]
```

Here's the object...

... and the property name in quotes.

We find dot notation the more readable of the two.

- When a property name contains spaces, you need to place it inside quotes. For example, the following address object has the "street no" as a property:

```
let address = {
  "street no": 23,
  street: “O'Connell Street",
  suburb: “Auckland CBD",
  city: "Auckland",
};
```

The property name could also be a variable that evaluates to a string denoting the property name.

```
const property = "name";
const hero = {
  name: "Batman",
};

hero["name"]; // => 'Batman'
hero[property]; // => 'Batman
```

Exercise 4

1. Create an object called `rectangle` with two properties, `length` and `width` with number values of 10 and 50, respectively.
2. Log the area of the rectangle to the console
    1. The formula to finding the area of the rectangle:

       Area of a rectangle = Length x Width

Note: Access the object's properties using the *square bracket* notation.

Modifying the value of a property

- Like normal variables, to change the value of a property, you use the assignment operator (=).

```
let person = {
  firstName: "John",
  lastName: "Doe",
};
person.firstName = "Jane";
console.log(person); // Output: Jane
```


Can we add properties to objects after we've defined them?

Adding/Removing an object property

- A JavaScript object is a collection of unordered properties.
  - Properties are the values associated with a JavaScript object.
- You can ***add new properties*** to an existing object by simply giving it a value.

```
cat.favouriteColour = "Purple";
```

- The `delete` keyword deletes a property from an object.
  - The delete keyword deletes both the value of the property and the property itself.

```
delete cat.lastName;
```

Exercise 5

1. Creates a `newPerson` object containing a `firstName`, `lastName`, `favouriteNumber`, `favouriteDay` properties.
2. `Log` the object to the console
3. Add a property called `favouriteFood` to the object.
4. `Log` the object to the console
5. Remove the `favouriteDay` property from the object. Change the value of the `favouriteNumber` property by doubling the current number value.
6. `Log` the object to the console

MISSION READY

www.missionreadyhq.com

DARE TO DEVELOP

Thank you

Reuben Simpson