JavaScript

Functions & Objects-PT5

COGS3
Objects - a collection of properties

- All dogs have a list of activities they are doing:
  - walking
  - fetching a ball
  - sleeping

- Dogs have a weight.

- Cutest dog object (ever!) have a name "Fido"
Object representing a dog

- name: “Fido”
- weight: 40
- breed: “Mixed”
- loves: [“walks”, “fetching balls”]
We assign the object to the variable fido

```javascript
var fido = {
    name: "Fido",
    weight: 40,
    breed: "mixed",
    loves: ["walks", "fetching balls"]
};
```
Access object properties with "dot" notation:

```javascript
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.
Which is more readable notation?

2. Access properties using a string with [] notation:

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

Use the object along with the property value wrapped in quotes and in brackets to access the value of that property.
3 Change a property's value:

```javascript
fido.weight = 27;
fido.breed = "Chawalla/Great Dane mix";
fido.loves.push("chewing bones");
```

changing Fido's weight

changing his breed...

push simply adds a new item to the end of an array.
Enumerate all an object’s properties:

```javascript
var prop;
for (prop in fido) {
    alert("Fido has a " + prop + " property ");
    if (prop == "name") {
        alert("This is " + fido[prop]);
    }
}
```

Note: the order of the properties is arbitrary. We are using the `[ ]` notation to access the value of the property. Each time through the loop, the variable `prop` gets the string value of the next property value.

this means to go through all of the properties of the object.
Have fun with an object's array:

```javascript
var likes = fido.loves;
var likesString = "Fido likes";

for (var i = 0; i < likes.length; i++) {
    likesString += " " + likes[i];
}
alert(likesString);
```

Here we are assigning the value of Fido's loves array to the variable likes.

We can loop through the likes array and create a likesString of all Fido's interest.

And we can alert the string!
Pass an object to a function:

```javascript
function bark(dog) {
    if (dog.weight > 25) {
        alert("WOOF");
    } else {
        alert("yip");
    }
}

bark(fido);
```

We can pass an object to a function just like any other variable.

In the function, we can access the object's properties like normal, using the parameter name for the object.

We are passing fido as our argument to the function `bark`, which expects a dog object.
The Dot Operator .

The dot operator (.) gives you access to an object’s properties. In general it’s easier to read than the [“string”] notation:

- `fido.weight` is the size of fido.
- `fido.breed` is the breed of fido.
- `fido.name` is the name of fido.
- `fido.loves` is an array containing fido’s interests.
Can we add properties to objects after we’ve defined them?

To add a property:
```
fido.age = 5;
```
(simply assign a new value)

To delete any property:
```
delete fido.age;
```
(use delete keyword)
When an object is assigned to a variable, the variable is given a reference to the object. It doesn't hold the object itself.

function bark(dog) {
  ... code here ....
}

When we call `bark` and pass `fido` as an argument, we get a copy of the reference to the dog object.
1. We've defined an object, `fido`, and we are passing that object into a function, `loseWeight`.

```javascript
fido = { 
  name: "Fido",
  weight: 48,
  breed: "Mixed",
  loves: ["walks", "fetching balls"]
};

fido.weight = 48;

loseWeight(fido);
```

`fido` is a reference to an object, which means the object does not live in `fido` variable, but is pointed to by the `fido` variable.

When we pass `fido` to a function, we are passing the reference to the object.
2. the dog parameter of the loseWeight function gets a copy of the reference to fido. So, any changes to the properties of the parameter affect the object that was passed in.

When we pass fido into loseWeight, what gets assigned to the dog parameter is a copy of the reference. *not a copy of the object.*

So fido and dog point to the same object. When we subtract 10 lbs from dog.weight, we are changing the value of fido.weight.