JavaScript

Objects

Objects

- Suppose I wanted to model a single person: name, age, and city
- I could use an array like this:

```
var person = ['Cindy', 32, "Missoula"];
```

- To retrieve the person's hometown

```
person[2];
```

- What if I accidentally reversed the order?

```
var person2 = ['Travis', 'Los Angeles', 21];
```

Objects cont.

- The previous example is a perfect use case for an OBJECT.

```
var person = {
    name: 'Cindy',
    age: 32,
    city: 'Missoula'
};
```

Objects cont.

- Objects store data in key-value pairs.

```
var person = {
    name: 'Travis',
    age: 32,
    city: 'LA'
};
```

- Unlike arrays, objects have no order.

Retrieving Data

- You have two choices: bracket and dot notation.
- Bracket notation is similar to arrays.

```
var person = {
    name: 'Travis',
    age: 32,
    city: 'LA'
};

console.log(person['name']);
```

- Dot notation (most common):

```
var person = {
    name: 'Travis',
    age: 32,
    city: 'LA'
};

console.log(person.name);
```

Bracket vs Dot notation

- There are a few differences between the 2 notations.
- You cannot use dot notation if the property starts with a number

```
someObject.1blah    //INVALID
someObject['1blah'] //VALID
```

- You can lookup using a variable with bracket notation.

```
var str = 'name';
someObject.str //doesn't look for 'name'
someObject[str] //does evaluate str and looks for 'name'
```

- You cannot use dot notation for property names with spaces.

```
someObject.fav color    //INVALID
someObject['fav color'] //Valid
```

Updating Data

- Just like an array: access a property and re-assign it.

```
var person = {
    name: 'Travis',
    age: 32,
    city: 'LA'
};

//to update age
person['age'] += 1;
//to update city
person.city = 'London';
```

Creating Objects

- Like arrays, there are a few methods of initializing objects.
- Make an empty object and then add to it.

```
var person = {};
person.name = 'Travis';
person.age = 21;
person.city = "LA";
```

- All at once

```
var person = {
    name: 'Travis',
    age: 32,
    city: 'LA'
};
```

- Another way of initializing an Object.

```
var person = new Object();
person.name = 'Travis';
person.age = 21;
person.city = "LA";
```

Data

- Objects can hold all sorts of data.

```
var junkObject = {
    age: 57,
    color: 'purple',
    isHungry: true;
    friends: ['Horatio', 'Hamlet'],
    pet: {
        name: 'Hunter',
        species: 'Dog',
        age: 2
    }
};
```

Nesting Objects and Arrays

- It is common to nest Objects within Arrays. Like this:

```
var posts = [
    {
        title: 'Cats are mediocre',
        author: 'Blake',
        comments: ['Awesome post', 'terrible post']
    },
    {
        title: 'Cats are mediocre',
        author: 'Blake',
        comments: ['<3', 'This is great!']
    }
]
```

- It is also common to next Arrays within Objects.

Nesting cont.

- Nesting can get very complex with objects inside of an array inside of an object inside of an array.

Assignment 10.1: Movie Database Exercise

- Create an array of movie objects. Each movie should have a title, rating, and hasWatched properties.
- Iterate through the array and print out something that looks like this:

```
You have watched 'In Bruges' - 5 stars
You have not seen 'Frozen' - 4.5 stars
You have seen 'Max Max Fury Road' - 5 stars
You have not seen 'Les Miserables' - 3.5 stars

USE YOU OWN MOVIES!
```

Adding Methods to Objects

- A function declared in an object is considered a method.

```
var person = {
    name: 'Blake',
    age: function(x, y) {
        return x + y;
    },
    city: 'Jacksonville'
};
```

- To call the method, you call it like a normal function.

```
person.age(10, 16);
```