Upright Education

Advanced Data Types: Objects

Software Development Bootcamp

Understanding Objects in JavaScript

Why We Use Objects

- Objects allow us to model real-world entities in code
- They provide a way to group related data and functionality together
- Objects make it easier to organize and manage complex data structures
- They enable more intuitive and readable code by using descriptive property names
- Objects are versatile and can represent almost any concept or entity in your program

Objects

- An Object is a collection of related data and functions (methods), organized into key-value pairs.
- Objects allow you to group and manage information and behavior related to a specific entity or concept.

An Object Contains Properties

- An object is made up of properties. Each property has a key and a value
- **Key-Value Pairs**: Each property has a **key** (name) and a **value**. The key is a string, and the value can be any data type

Object Example

- `let tulip = {...}` is an assignment, setting the variable `tulip` to point to the object we define

```
let tulip = {
 species: "dog",
 color: "grey",
 spayed: true,
 breed: "terrier",
 weight: 14,
 favoriteActivity: "chasing
squirrels",
};
```

Keys Are Strings

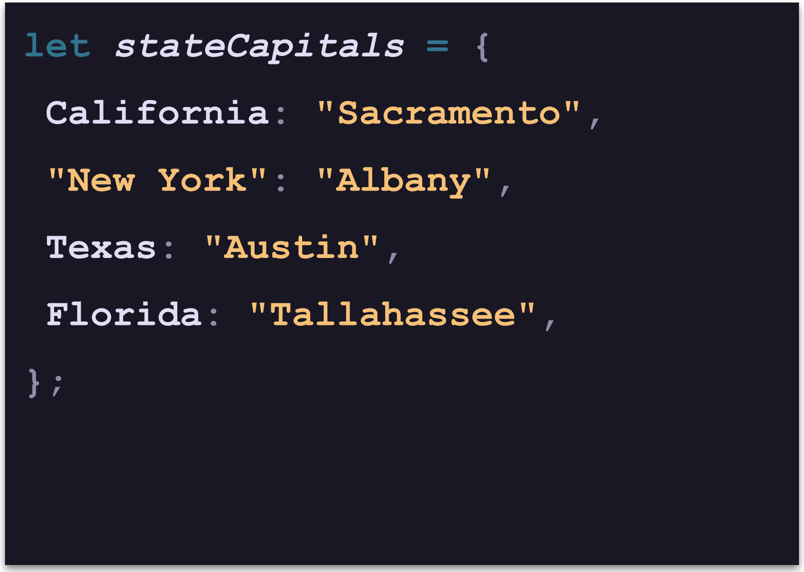
- All of the keys of an object are strings. Our **`tulip`** object has 6 keys:
  - **`species`**
  - **`color`**
  - **`spayed`**
  - **`breed`**
  - **`weight`**
  - **`favoriteActivity`**
- As long as the keys have no spaces, we don't need to wrap them in quotation marks

Objects As Dictionaries

- An object is useful for putting many similar things together.
- Just like a dictionary, objects let you organize and quickly access information

Simple Dictionary of US State Capitals

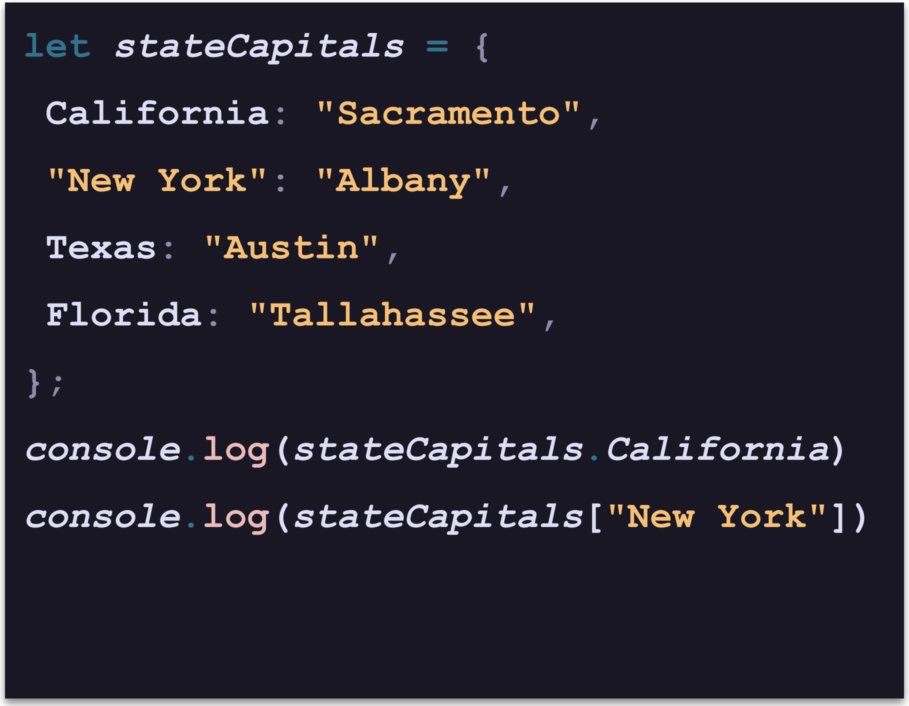
- To find a capital, you "look up" the state name



```
let stateCapitals = {
 California: "Sacramento",
 "New York": "Albany",
 Texas: "Austin",
 Florida: "Tallahassee",
};
```

Accessing values

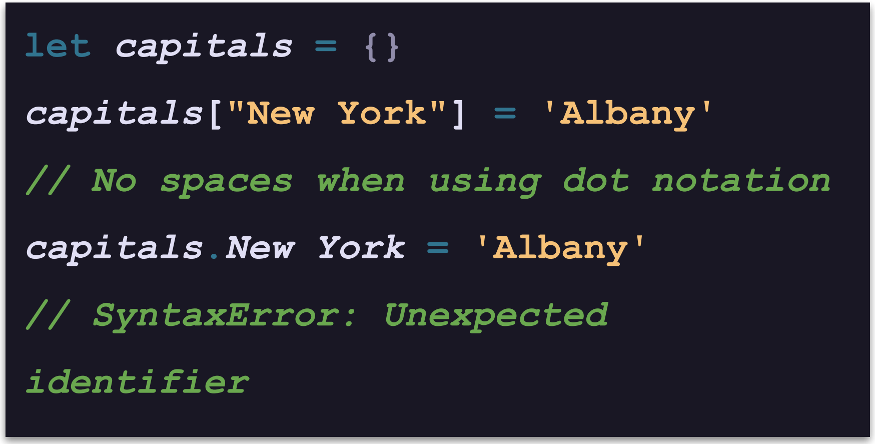
- You can access the value of a property using **dot notation** or **bracket notation**



```
let stateCapitals = {
 California: "Sacramento",
 "New York": "Albany",
 Texas: "Austin",
 Florida: "Tallahassee",
};
console.log(stateCapitals.California)
console.log(stateCapitals["New York"])
```

Dot Notation VS. Bracket Notation

- Use dot notation when you know the exact property name, and it's a simple word (no spaces)
- Use bracket notation when
  - The property name has spaces or special characters
  - The property name is stored in a variable



```
let capitals = {}
capitals["New York"] = 'Albany'
// No spaces when using dot notation
capitals.New York = 'Albany'
// SyntaxError: Unexpected
identifier
```

Setting Object Properties

- You can set the properties of an object with either dots or brackets followed by a = (single equal sign)
  - Adding properties works even after the object has been created
  - Setting a property with a key that already exists replaces the original value.

```
let stateCapitals = {
 California: "Sacramento",
 "New York": "Albany",
 Texas: "Austin",
 Florida: "Tallahassee",
};
stateCapitals.Wyoming = "Cheyenne"
stateCapitals["North Dakota"] =
"Bismarck"
```

Objects Can Contain Complex Information

- Objects allow us to design data structures (tools for organizing and storing many related values).

```
let alice = {
 name: "Alice Abrams",
 homeAddress: {
   street: "12 Maple St.",
   location: {
     latitude: 44.4759,
     longitude: -73.2121,
   },
 },
 pets: ["Mittens", "Fido"],
};
```

Object Methods: `Object.keys()`

- **Parameter**: The object whose keys are to be returned
- **Return value**: An array of strings representing the object's keys

```javascript
let alice = {
 name: "Alice Abrams",
 homeAddress: {
   street: "12 Maple St.",
   location: {
     latitude: 44.4759,
     longitude: -73.2121,
   },
 },
 pets: ["Mittens", "Fido"],
};
Object.keys(alice)
// ['name', 'homeAddress', 'pets']
```

Object Methods: `Object.values()`

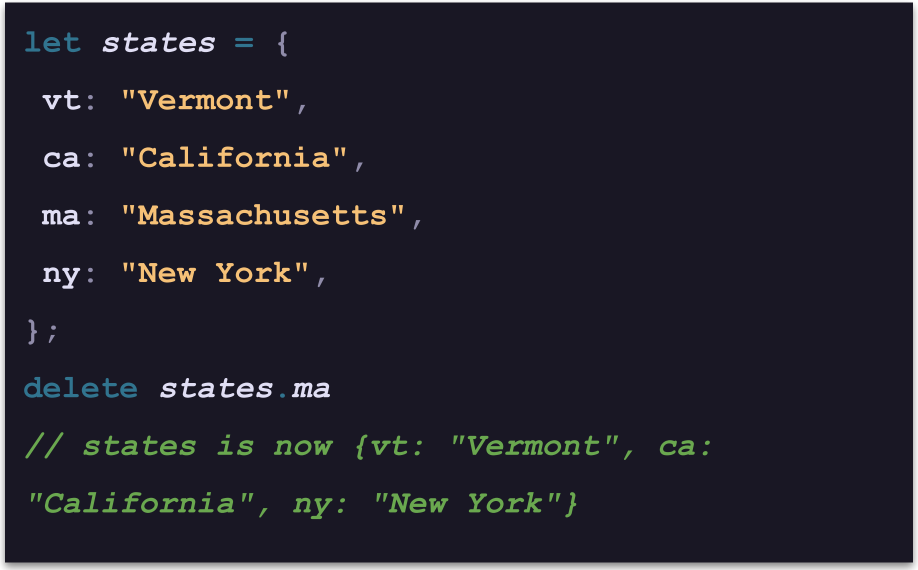
- **Parameter:** The object whose values are to be returned
- **Return value:** An array of strings representing the object's values

```
let alice = {
 name: "Alice Abrams",
 homeAddress: {
   street: "12 Maple St.",
   location: {
     latitude: 44.4759,
     longitude: -73.2121,
   },
 },
 pets: ["Mittens", "Fido"],
};
Object.keys(alice)
// ['name', 'homeAddress', 'pets']
```

Upright Education

Removing Key-Value Pairs From Objects

- Used to delete a key value pair from an object.
- Use delete when you're sure you don't need the property anymore



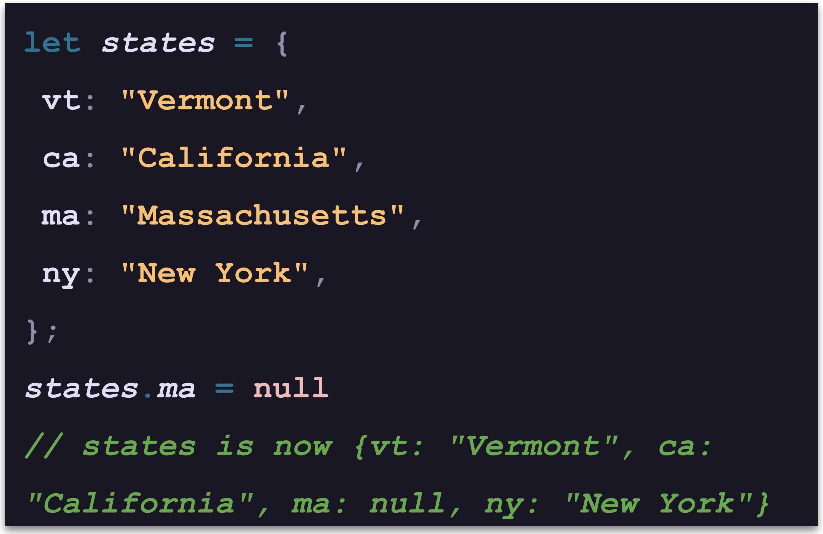
```
let states = {
 vt: "Vermont",
 ca: "California",
 ma: "Massachusetts",
 ny: "New York",
};
delete states.ma
// states is now {vt: "Vermont", ca:
"California", ny: "New York"}
```

Why Keep The Key?

- Maintain object structure for consistency
- Preserve property for future use
- Indicate that a value was intentionally removed or invalidated
- Keep track of what properties an object should have, even if some are currently empty

Alternative To Delete

- Set value to **`null`**
- Use when you want to explicitly indicate “no value”
- They key remains, but its value is **`null`**



```
let states = {
 vt: "Vermont",
 ca: "California",
 ma: "Massachusetts",
 ny: "New York",
};
states.ma = null
// states is now {vt: "Vermont", ca:
"California", ma: null, ny: "New York"}
```

Upright Education

Exercise

Presidents