EECS 368
Programming Language Paradigms

David O. Johnson

Fall 2022

Reminders

- Assignment 1 due: 11:59 PM, Wednesday, September 7
- Assignment 2 due: 11:59 PM, Monday, September 19

Any Questions?

In-Class Problem Solution

- 4-(8-31) In-Class Problem Solution.pptx

Any Questions?

JavaScript Objects

- Remember when I said JavaScript followed the function-oriented design paradigm and not the object-orient design paradigm?
- Well, that was only partly true.
- JavaScript actually supports both design paradigms!

Defining Objects

- A JavaScript object is an arbitrary collections of properties.
- One way to create an object is by using braces as an expression.
- Inside the braces, there is a list of properties separated by commas.
- Each property has a name followed by a colon and a value.

```
let day1 = {
  squirrel: false,
  events: ["work", "touched tree", "pizza", "running"]
};
console.log(day1.squirrel);
// → false
console.log(day1.events);
// → Array(4) [ "work", "touched tree", "pizza", "running" ]
```

Properties

- The two main ways to access properties in JavaScript are with a dot and with square brackets.
- When using a dot, the word after the dot is the literal name of the property.
  - value.x fetches the property of value named “x”
  - For example, if you know that the property you are interested in is called color, you say value.color.
- When using square brackets, the expression between the brackets is evaluated to get the property name.
  - Property names are strings.
  - They can be any string, but the dot notation works only with names that look like valid binding names.
  - So, if you want to access a property named 2 or John Doe, you must use square brackets: value[2] or value["John Doe"].

Methods

- Properties that contain functions are called methods.

- For example, every string has a toUpperCase method.
- When called, it will return a copy of the string in which all letters have been converted to uppercase.

```
let doh = "Doh";
console.log(typeof doh.toUpperCase);
// → function
console.log(doh.toUpperCase());
// → DOH
```

Methods

- This is an example of a push method that adds values to the end of an array:

```
let sequence = [1, 2, 3];
sequence.push(4);
sequence.push(5);
console.log(sequence);
// → [1, 2, 3, 4, 5]
```

Assigning Properties & Values to Objects

- It is possible to assign a value to a property expression with the = operator.
- This will replace the property's value if it already existed.
- Or create a new property on the object if it didn't.

```
let day1 = {
  squirrel: false,
  events: ["work", "touched tree", "pizza", "running"]
};
console.log(day1.wolf);
// → undefined
day1.wolf = false;
console.log(day1.wolf);
// → false
```

- Object.assign function that copies all properties from one object into another.

```
let objectA = {a: 1, b: 2};
Object.assign(objectA, {b: 3, c: 4});
console.log(objectA);
// → {a: 1, b: 3, c: 4}
```

Deleting Properties from Objects

- The delete operator removes the named property from the object.
- This is not a common thing to do, but it is possible.
- The difference between setting a property to undefined and actually deleting it is that:
  - in the first case of setting a property to undefined, the object still has the property (it just doesn't have a very interesting value)
  - in the second case of deleting it, the property is no longer present, or undefined.

```
let anObject = {left: 1, right: 2};
console.log(anObject.left);
// → 1
delete anObject.left;
console.log(anObject.left);
// → undefined
```

Binary in Operator

- The binary in operator, when applied to a string and an object, tells you whether that object has a property with that name.

```
let anObject = {left: 1, right: 2};
console.log(anObject.left);
// → 1
delete anObject.left;
console.log(anObject.left);
// → undefined
console.log("left" in anObject);
// → false
console.log("right" in anObject);
// → true
```

Object.keys Function

- To find out what properties an object has, you can use the Object.keys function.
- You give it an object, and it returns an array of strings—the object's property names.

```
console.log(Object.keys({x: 0, y: 0, z: 2}));
// → ["x", "y", "z"]
```

Any Questions?

Arrays

- Arrays are a kind of object specialized for storing sequences of things.
- Arrays are defined as a list of values between square brackets, separated by commas.
- The notation for getting at the elements inside an array also uses square brackets.
- The first index of an array is zero, not one.
- So the first element is retrieved with listOfNumbers[0].
- Zero-based counting has a long tradition in technology and in certain ways makes a lot of sense, but it takes some getting used to.
- Think of the index as the amount of items to skip, counting from the start of the array.

```
let listOfNumbers = [2, 3, 5, 7, 11];
console.log(listOfNumbers[2]);
// → 5
console.log(listOfNumbers[0]);
// → 2
console.log(listOfNumbers[2 - 1]);
// → 3
```

- The length property of an array tells us how many elements it has.

```
let listOfNumbers = [2, 3, 5, 7, 11];
console.log(listOfNumbers.length);
// → 5
```

Array Iteration

- Going over arrays one element at a time is something that comes up a lot.
- To do that you'd run a counter over the length of the array and pick out each element in turn.

```
for (let i = 0; i < JOURNAL.length; i++) {
  let entry = JOURNAL[i];
  // Do something with entry
}
```

- There is a simpler way to write such loops in modern JavaScript.

```
for (let entry of JOURNAL) {
  console.log(`${entry.events.length} events.`);
}
```

- When a for-loop looks like this, with the word of after a variable definition, it will loop over the elements of the value given after of.
- This works not only for arrays but also for strings and some other data structures.

Any Questions?

Mutability

- The types of values discussed in earlier lectures, such as numbers, strings, and Booleans, are all immutable—it is impossible to change values of those types.
- You can combine them and derive new values from them, but when you take a specific string value, that value will always remain the same.
- The text inside it cannot be changed.
- If you have a string that contains "cat", it is not possible for other code to change a character in your string to make it spell "rat".

Mutability

- Objects work differently.
- Object values can be modified and are thus mutable.
- You can change their properties, causing a single object value to have different content at different times.
- When we have two numbers, 120 and 120, we can consider them precisely the same number, whether or not they refer to the same physical bits.
- With objects, there is a difference between:
  - having two references to the same object
  - having two different objects that contain the same properties

Mutability

Consider this code:

- The object1 and object3 bindings grasp the same object.
- Which is why changing object1 also changes the value of object3.
- They are said to have the same identity.
- The binding object2 points to a different object, which initially contains the same properties as object1 but lives a separate life.

```
let object1 = {value: 13};
let object2 = {value: 13};
let object3 = object1

object1.value = 26;
console.log(object3.value);
// → 26
console.log(object2.value);
// → 13
```

Mutability

Consider this code:

- When you compare objects with JavaScript's == operator, it compares by identity.
- It will produce true only if both objects are precisely the same value.
- Comparing different objects will return false, even if they have identical properties.

```
let object1 = {value: 13};
let object2 = {value: 13};
let object3 = object1

console.log(object1 == object3);
// → true
console.log(object1 == object2);
// → false
```

Any Questions?

JSON

- JSON (pronounced “Jay Sawn” or “Jason”), which stands for JavaScript Object Notation is widely used as a data storage and communication format on the Web, even in languages other than JavaScript.
- JSON looks similar to JavaScript’s way of writing arrays and objects, with a few restrictions.
- All property names have to be surrounded by double quotes, and only simple data expressions are allowed—no function calls, bindings, or anything that involves actual computation.
- Comments are not allowed in JSON.

JSON

For example:

- A journal entry might look like this when represented as JSON data:

```
{
  "squirrel": false,
  "events": ["work", "touched tree", "pizza", "running"]
}
```

- JavaScript gives us two functions to convert data to and from this format.
- JSON.stringify takes a JavaScript value and returns a JSON-encoded string.
- JSON.parse takes such a string and converts it to the object it encodes.

```
let string = JSON.stringify({squirrel: false,
                             events: ["weekend"]});
console.log(string);
// → {"squirrel":false,"events":["weekend"]}
console.log(JSON.parse(string).events);
// → ["weekend"]
```

Any Questions?

The Math Object

The Math Object is a grab bag of number-related utility functions.

- Math.max (maximum)
- Math.min (minimum)
- Math.sqrt (square root)
- Math.cos (cosine)
- Math.sin (sine)
- Math.tan (tangent)
- Math.acos (inverse cosine)
- Math.asin (inverse sine)
- Math.atan (inverse tangent)
- Math.PI (π)
- Math.random (generates pseudorandom number ≥ 0 and < 1)
- Math.floor (floor: 1.1 → 1)
- Math.ceil (ceiling: 1.1 → 2)
- Math.round (to the nearest whole number)
- Math.abs (absolute value)

Any Questions?

Summary

- Objects and arrays (which are a specific kind of object) provide ways to group several values into a single value.
- Conceptually, this allows us to put a bunch of related things in a bag and run around with the bag, instead of wrapping our arms around all of the individual things and trying to hold on to them separately.
- Most values in JavaScript have properties, the exceptions being null and undefined.
- Properties are accessed using value.prop or value["prop"].
- Objects tend to use names for their properties and store more or less a fixed set of them.
- Arrays, on the other hand, usually contain varying amounts of conceptually identical values and use numbers (starting from 0) as the names of their properties.
- There are some named properties in arrays, such as length and a number of methods.
- Methods are functions that live in properties and (usually) act on the value they are a property of.
- You can iterate over arrays using a special kind of for-loop—for (let element of array).
- JavaScript Object Notation (JSON) is widely used as a data storage and communication format on the Web, even in languages other than JavaScript.
- The Math Object is a grab bag of number-related utility functions.

Any Questions?

In-Class Problem

Show the console output (?) for each of these code snippets:

```
let listOfNumbers = [2, 3, 5, 7, 11];
console.log(listOfNumbers[2]);
// → ?
console.log(listOfNumbers[0]);
// → ?
console.log(listOfNumbers[2 - 1]);
// → ?
```

```
let object1 = {value: 10};
let object2 = object1;
let object3 = {value: 10};
console.log(object1 == object2);
// → ?
console.log(object1 == object3);
// → ?
object1.value = 15;
console.log(object2.value);
// → ?
console.log(object3.value);
// → ?
```