The Building Blocks of Programming

CSCI 185: Intro to programming with JavaScript

Announcements

- Quiz on Friday (during class time)
- UNCA withdrawal deadline this Friday
- If you haven’t turned in HW4-5, make it a priority.
    - Last day to submit HW4 (late): March 20 (next Monday)
    - Last day to submit HW5 (late): April 3
- Most of Quiz 2 (this Friday will cover HW4-HW5)

Questions about the practice quiz?

Before we continue our exploration of JavaScript….

Would it be useful to go over any of the practice quiz questions?

Warmup: Switch Between Classes

1. Download the lecture files: https://github.com/csci-185/spring2023/raw/main/course-files/lectures/lecture16.zip
2. Open them in VS Code
3. See if you can figure out how to switch the class that's attached to the body tag when the user clicks one of the buttons:
   a. Clicking the fall button should update the body element's class to "fall"
   b. Clicking the winter button should update the body element's class to "winter."
   c. Clicking the spring button should update the body element's class to "spring."

Hint: target the body element using one of the JavaScript selection methods (e.g., document.querySelector(???), and then set the className property of the targeted element.

Outline

In this class, we're going to do a high-level overview of programming, and the fundamental constructs of JavaScript

1. **Data**
   data types and variables
2. **Statements & Expressions**
   operators and functions (part 1)
3. **Events**
4. **Control**
   conditionals, iteration, and looping

For today

JavaScript Data Types

There are 7 basic data types in JavaScript:

number	For numbers of any kind: integer or floating-point	1.4, 33, 99999999
string	For strings (text). A string may have one or more characters, there’s no separate single-character type	‘hello world!’
boolean	for true/false.	true, false
null	for unknown values – has a single value null	null
undefined	for unassigned values – has a single value undefined	undefined
object	for more complex data structures.	{ name: ‘ralph’, species: ‘dog’ }
symbol	for unique identifiers (we won’t be using this one)	

Why do data types matter?

Each data type has its own abilities. For instance:

- **numbers** are good for mathematical calculations
- **strings** are good for doing text manipulations, e.g.:
    - convert to lowercase
    - check if a word is in a sentence
    - calculate the number of characters
- **booleans** are good for figuring out whether to execute a code block (for conditional statements a and loops).
- **Objects** are good for grouping related data and/or functions together

Why do data types matter?

Try executing these two statements in the console and check out what's different:

```
"2" + "2"
2 + 2
```

Takeaway: different data types have different powers

- Numbers are good for doing math
- Strings have functionality that allows you to format / manipulate text:
    - e.g., "hello".toUpperCase();
- Booleans are good for logic (we'll get into that more soon)!

Figuring out what data type you have

```
typeof 'hello world!';
typeof true;
typeof false;
typeof null;
typeof undefined;
typeof 23.4;
typeof 4500;
typeof [1, 3, 4, 6];
typeof { name: 'Lester', species: 'dog', age: 15};
```

Functions that convert between data types

```
// String(), Number(), Boolean()
console.log(123, typeof 123, String(123), typeof String(123));
console.log('123', typeof '123', Number('123'), typeof Number('123'));
console.log('true', typeof 'true', Boolean('true'), typeof Boolean('true'));
```

Takeaway

It’s important to know **what data type you have** in order produce the result you want

Variables

Variables

Used for ***temporary data storage*** and making things easier to read. Like ‘scratch paper.’ Consider the following code. Visualize it

```
// copy this code and run it in your browser console:
let x = 1;
let y = 6;
let z = 4;
x = 2;
z = x + y;
x = 4;
z = z + x;

console.log(x);
console.log(y);
console.log(z);
```

03_variables.js

Variables

1. Variables are containers for storing and / or referencing data
2. Variables can also be used to alias data, functions, variables, objects, functions, etc.
3. You assign values to variables using the **assignment operator** (equal sign)

=

Naming Variables: Case Sensitivity

JavaScript is case-sensitive.

```
let a = 3;
let A = 33;
console.log(a);
console.log(A);
```

Naming Variables

Although JavaScript doesn’t care what you name your variables (beyond the rules specified in the previous slide), there are some conventions that most people follow:

1. Ensure your variable names are mnemonic
2. “camelCase” for functions and variable names
   a. let firstName = 'Erica';
   b. let lastName = 'Hernandez';
3. Hyphens for file names: hello-world.js

Naming Variables: Camelcase

Examples of camelcase variables and functions:

Variables

```
let timeLeftTilEndOfClass = 35;
let firstName = 'Jazmin';
let lastName = 'Soltani';
```

Functions

```
const divideTwoNums = (num, denom) => {
  return num / denom;
};

const moveAvatarLeft = (dog) => {
  dog.x -= 1;
  dog.redraw();
}
```

Naming Variables: Mnemonic

Which program is easier to read and reason about?

```
let x = 65;
let y = 3;
let z = x * y;
console.log(z);
```

```
let rate = 65;
let time = 3;
let distance = rate * time;
console.log(distance);
```

Naming Variables: Mnemonic

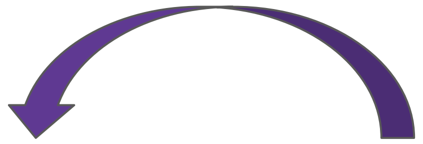
Which program is easier to detect errors?

```
let x = 65;
let y = 3;
let z = x * x;
console.log(z);
```

```
let rate = 65;
let time = 3;
let distance = rate * rate;
console.log(distance);
```

Variables

Turn the following statement into a sentence:



```
let speed = 65;
```

Create a new container called **speed** and *put* the number **65** into it

- or -

Create a new container called **speed** and *assign* the number **65** to it

Declaring versus assigning

In JavaScript, you need to declare your variables before you assign values to them. This can either be done all on one line...

```
let a = 3;    // declares and assigns the value 3 to the variable a
```

or in two steps...

```
let a;        // declares the variable a
a = 3;        // assigns the value 3 to the variable a
```

visualize it

Declaration keywords

In JavaScript, there are three keywords for declaring variables: let, const, and var

- **let**
  Means that the variable may be reassigned. It also signals that the variable will be used only in the block it's defined in (we'll talk about this more when we get to functions).
- **const**
  Means that the variable won't be reassigned. It's immutable. Used in cases where you declare and assign once, but don't change it afterwards.
- **var**
  Avoid. Old way to do things prior to ES6.
  "The weakest signal available" — "The variable may or may not be reassigned, and the variable may or may not be used for an entire function."

Source: https://medium.com/javascript-scene/javascript-es6-var-let-or-const-ba58b8dcde75

const and let demo (console)

Expressions

constants, operators, functions

Constants ~ Values

Constants (i.e. raw data values) don’t need to be evaluated…they just are

3

2.5

‘Hello World’

true

Operators ~ a way to perform a simple computation

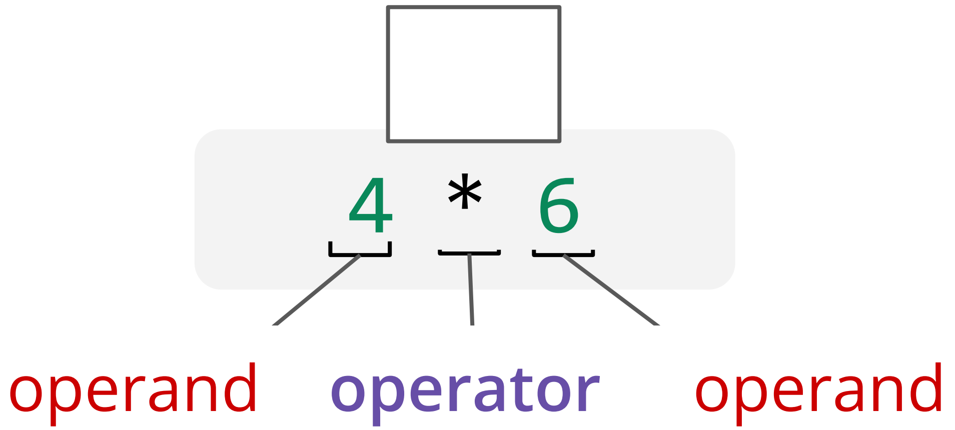
Operators, when used in conjunction with their required operands, evaluate to a value. Also an expression:

```
2 + 1                   // expression that evaluates to 3
 5 / 2                  // expression that evaluates to 2.5
'Hello ' + 'World'      // expression that evaluates to 'Hello World'
2 + 1 === 3             // expression that evaluates to true
```

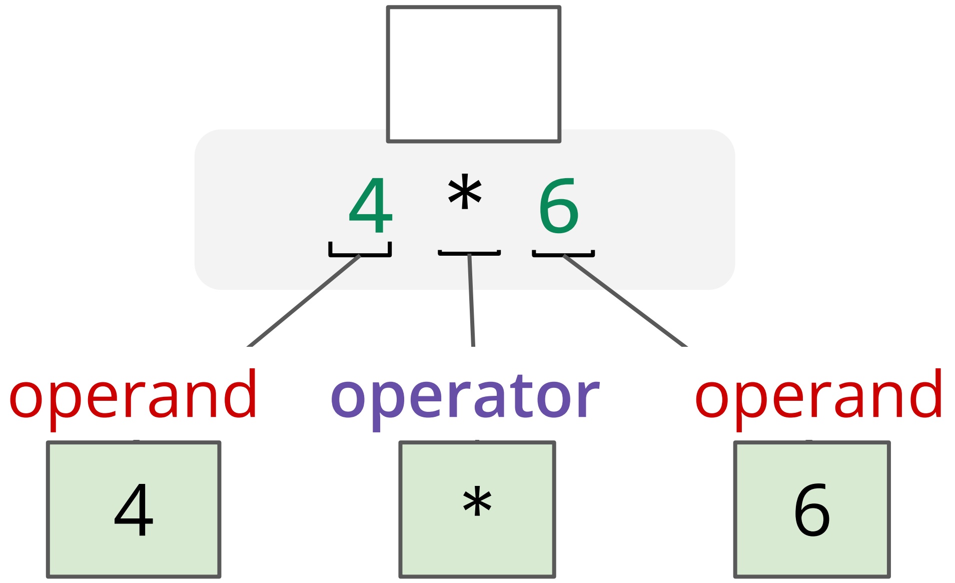
Example 1: Evaluate this expression (operator)

```
4 * 6
```

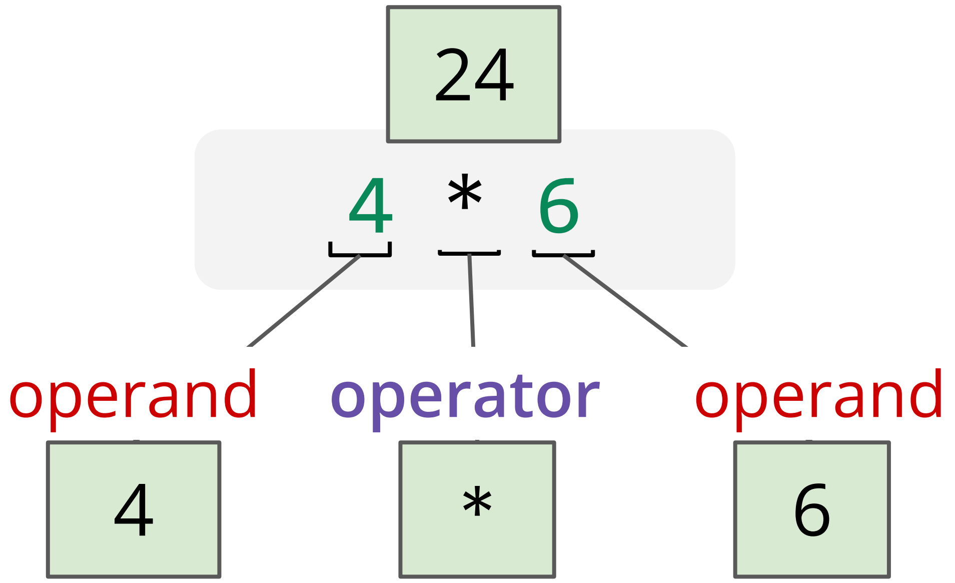
Example 1: Evaluate this expression (operator)



Example 1: Evaluate this expression (operator)



Example 1: Evaluate this expression (operator)



Arithmetic Operators

+	Addition	Adds values on either side of the operator
-	Subtraction	Subtracts right hand operand from left hand operand
*	Multiplication	Multiplies values on either side of the operator
/	Division	Divides left hand operand by right hand operand
**	Exponent	Performs exponential (power) calculation on operators
%	Modulus	Divides left hand operand by right hand operand; returns remainder
++	Increment	Adds 1 to the number
--	Decrement	Subtracts 1 from the number

Operator Precedence

Python evaluates expressions as is done in mathematics (PEMDAS). After precedence rules, expressions are evaluated left to right.

1	()	Parenthesis
2	**	Exponentiation
3	*, /, %	Multiplication, division, remainder
4	+, -	Addition, subtraction

Order of Operations

```
let result = 16 - 2 * 5 / 3 + 1;
```

Order of Operations

```
let result = 16 - 2 * 5 / 3 + 1;
```

Order of Operations

```
let result = 16 - 2 * 5 / 3 + 1;
                  16 - 10 / 3 + 1;
```

Order of Operations

```
let result = 16 - 2 * 5 / 3 + 1;
                  16 - 10 / 3 + 1;
                  16 - 3.33 + 1;
```

Order of Operations

```
let result = 16 - 2 * 5 / 3 + 1;
             16 - 10 / 3 + 1;
             16 - 3.333 + 1;
             12.666 + 1;
```

Order of Operations

```
let result = 16 - 2 * 5 / 3 + 1;
             16 - 10 / 3 + 1;
             16 - 3.333 + 1;
             12.666 + 1;
    result = 13.666;
```

Templates

- Templates, or "template literals" are "smart strings" that allow you to embed expressions
- They're convenient for generating larger chunks of HTML that depend on variables or other data.
- Uses the "backtick" character (instead of regular single or double quotes) to indicate that you are specifying a template (above the tab key):

  ` <template goes here> `

- Within the template, expressions represented like this:

  ${ my_expression }

Source: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals

Rewriting previous HTML using template syntax

```
const name = "Jane";
const pic = "http://website.com/avatar.png";
const score = 300;
const html = `
    <div class="card">
        <img src="${pic1}">
        <p>
            ${name}'s high score is:
            ${score}
        </p>
    </div>`;
```

More on templates

```
const name = 'Walter';
console.log(` A template
    can be multiple lines.
    It can also evaluate expressions like:
    ${2 + 2} or
    ${name} or
    ${getTodaysDate()}
` );
```