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

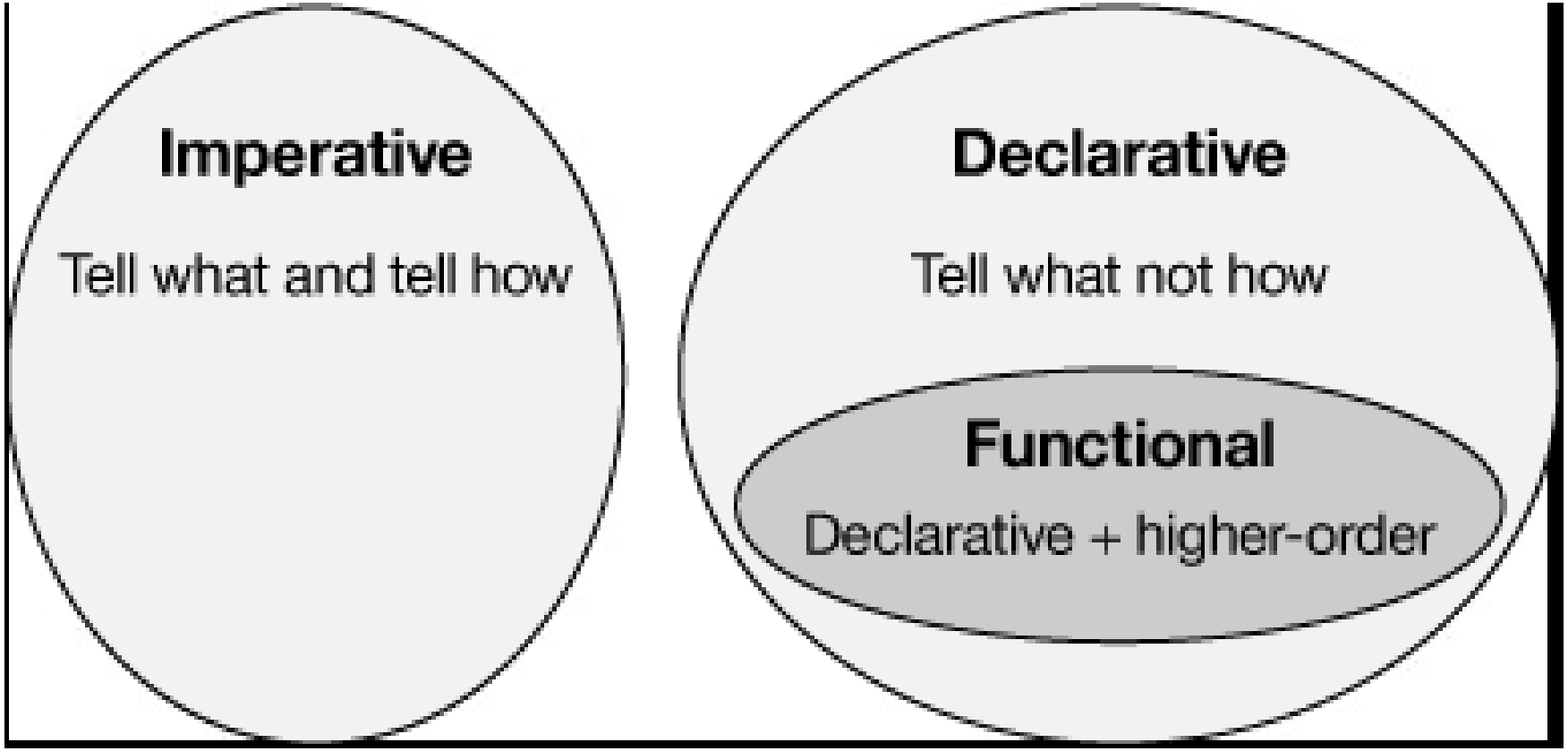
Any Questions?

In-Class Problem Solution

- 1-(8-24) In-Class Problem Solution.pptx

Any Questions?

Imperative vs. Declarative Programming Languages



EECS 368

- We are going to look at two languages in detail
  - JavaScript (Imperative)
  - Haskell (Declarative – Functional)
- We will also look at Cloud Computing from a language point of view.
- We will study several DSLs:
  - JavaScript Support Vector Graphics (SVG)
  - JavaScript Canvas
  - Haskell QuickCheck

What is JavaScript?

- JavaScript was introduced in 1995 as a way to add programs to web pages in the Netscape Navigator browser.
- The language has since been adopted by all other major graphical web browsers.
- It has made modern web applications possible—applications with which you can interact directly without doing a page reload for every action.

What is JavaScript?

- It is important to note that JavaScript has almost nothing to do with the programming language named Java.
- The similar name was inspired by marketing considerations rather than good judgment.
- When JavaScript was being introduced, the Java language was being heavily marketed and was gaining popularity.
- Someone thought it was a good idea to try to ride along on this success.
- Now we are stuck with the name!

What is JavaScript?

- After its adoption outside of Netscape, a standard document was written to describe the way the JavaScript language should work so that the various pieces of software that claimed to support JavaScript were actually talking about the same language.
- This is called the ECMAScript standard, after the Ecma International organization that did the standardization.
- In practice, the terms ECMAScript and JavaScript can be used interchangeably—they are two names for the same language.

What is JavaScript?

- Web browsers are not the only platforms on which JavaScript is used.
- Some databases, such as MongoDB and CouchDB, use JavaScript as their scripting and query language.
- Several platforms for desktop and server programming, most notably Node.js, provide an environment for programming JavaScript outside of the browser.
- We will discuss this more when we cover Cloud Computing.

What is JavaScript?

- JavaScript is very much alive as a language and books (including this one) get out-of-date quickly.
- www3 is a good resource for questions about JavaScript.
- Also just googling your question will result in a plethora of answers.

The console.log function

- The examples in my lectures and the book use console.log to output values.
- Most JavaScript systems (including all modern web browsers and Node.js) provide a console.log function that writes out its arguments to some text output device.
- In browsers, the output lands in the JavaScript console.
- This part of the browser interface is hidden by default, but most browsers open it when you press F12 or, on a Mac, command-option-I.
- If that does not work, search through the menus for an item named Developer Tools or similar.
- For example, using FireFox:

```
>> console.log("hello world")
hello world
```

- We will talk more about console.log later.

Using Textbook's Sandbox

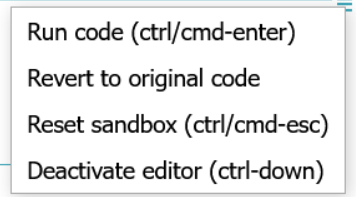
Highlight the code you want to run …

```
console.log(null == undefined);
// → true
console.log(null == 0);
// → false
```

and this will happen …

```
1 console.log(null == undefined);
2 // → true
3 console.log(null == 0);
4 // → false
```

Click on menu to right and "Run code" or type ctrl-cmd or ctrl-enter



and this will happen …

```
1 console.log(null == undefined);
2 // → true
3 console.log(null == 0);
4 // → false
true
false
```

Using Textbook's Sandbox

You can also modify the code and run the modified code.

```
1 console.log(null == undefined);
2 // → true
3 console.log(null == 0);
4 // → false
```

```
true
false
```

```
1 console.log(null == undefined);
2 // → true
3 console.log(null == undefined);
4 // → false
```

```
true
true
```

Any Questions?

JavaScript Basics

- Numbers
- Arithmetic
- Special numbers
- Strings
- Unary operators
- Boolean values and comparison operators
- Logical operators
- Empty values
- Automatic type conversion
- Short-circuiting of logical operators

Numbers

- Integers are written as follows:

  13

- Fractional numbers are written by using a dot.

  9.81

- For very big or very small numbers, you may also use scientific notation by adding an e (for exponent), followed by the exponent of the number.

  2.998e8

- That is $2.998 \times 10^8 = 299{,}800{,}000$.

Arithmetic

- Arithmetic operations such as addition or multiplication take two number values and produce a new number from them.
- For example:

100 + 4 * 11

- Operators:
  - Addition: +
  - Subtraction: -
  - Multiplication: *
  - Division: /
  - Exponentiation: **
  - Remainder (or Modulo): %
    - X % Y is the remainder of dividing X by Y.
    - For example, 314 % 100 produces 14, and 144 % 12 gives 0.

Arithmetic Precedence

- When operators appear together without parentheses, the order in which they are applied is determined by the precedence of the operators.
- ** has the highest precedence
- *, /, and % have the next highest precedence.
- + and – have the lowest precedence.
- When multiple operators with the same precedence appear next to each other, as in 1 - 2 + 1, they are applied left to right: (1 - 2) + 1.
- When in doubt, just add parentheses.

Special Numbers

- Infinity = ∞
  - For example: 1/0 = Infinity
- -Infinity = -∞
  - For example: -1/0 = -Infinity
- NaN = Not a Number
  - For example: 0/0 = NaN

Strings

- Strings are used to represent text.
- They are written by enclosing their content in quotes.
- You can use single quotes, double quotes, or backticks to mark strings, as long as the quotes at the start and the end of the string match.
- For example:

'Float on the ocean'

"Lie on the ocean"

`Down on the sea`

- Newlines (the characters you get when you press enter) can be included when the string is quoted with backticks (`).

Escaping Character

- Backslash (\) indicates that the character after it has a special meaning.
- Backslash (\) is called the escaping character.
- \n = newline
- \t = tab
- \" = " (not the end of a string)
- \\ = \
- For example:

  "A newline character is written like \"\\n\"."
- produces:

  A newline character is written like "\n".
- We will discuss other escape sequences later.

Concatenation of Strings

- Concatenation with +
- For example, "con" + "cat" + "e" + "nate" produces:

concatenate

Template Literal

- Strings written with single or double quotes behave very much the same—the only difference is in which type of quote you need to escape inside of them.

- Backtick-quoted strings, usually called template literals, can do a few more tricks.
- Apart from being able to span lines, they can also embed other values.
- When you write something inside ${} in a template literal, its result will be computed, converted to a string, and included at that position.
- For example, `half of 100 is ${100 / 2}` produces:

half of 100 is 50

Unary Operators

- The typeof operator, produces a string naming the type of the value you give it.

```
console.log(typeof 4.5)
// → number
console.log(typeof "x")
// → string
```

- The minus operator is both a binary operator and a unary operator.

```
console.log(- (10 - 2))
// → -8
```

Boolean Values and Comparison Operators

- JavaScript has a Boolean type, which has just two values, true and false, which are written as those words.
- Comparison operators:
  - < less than
  - > greater than
  - <= less than or equal
  - >= greater than or equal
  - == equal
  - != not equal
- Examples:

```
console.log(3 > 2)
// → true
console.log(3 < 2)
// → false
console.log("Aardvark" < "Zoroaster")
// → true
```

- The way strings are ordered is roughly alphabetic but not really what you'd expect to see in a dictionary.
- Uppercase letters are always "less" than lowercase ones, so "Z" < "a"
- Nonalphabetic characters (!, -, and so on) are also included in the ordering.
- When comparing strings, JavaScript goes over the characters from left to right, comparing the Unicode codes one by one.

Logical Operators

In order of precedence:

- ! → logical NOT

```
console.log(!false)
// → true
```

- && → logical AND

```
console.log(true && false)
// → false
console.log(true && true)
// → true
```

- || → logical OR

```
console.log(false || true)
// → true
console.log(false || false)
// → false
```

conditional operator (or ternary operator)

```
console.log(true ? 1 : 2);
// → 1
console.log(false ? 1 : 2);
// → 2
```

- The value on the left of the question mark "picks" which of the other two values will come out.
- When it is true, it chooses the middle value
- When it is false, it chooses the value on the right.

Empty Values

- null and undefined
- Denote the absence of a meaningful value.
- They are themselves values, but they carry no information.
- The difference in meaning between undefined and null is an accident of JavaScript's design, and it doesn't matter most of the time.
- Treat them as mostly interchangeable.

Automatic Type Conversion

- JavaScript goes out of its way to accept almost any program you give it, even programs that do odd things.

```
console.log(8 * null)
// → 0
console.log("5" - 1)
// → 4
console.log("5" + 1)
// → 51
console.log("five" * 2)
// → NaN
console.log(false == 0)
// → true
```

- When an operator is applied to the "wrong" type of value, JavaScript will quietly convert that value to the type it needs, using a set of rules that often aren't what you want or expect.
- This is called type coercion.
- This is both good and bad!

Short-Circuiting of Logical Operators

- The logical operators && and || handle values of different types in a peculiar way.
- They will convert the value on their left side to Boolean type in order to decide what to do.
- But depending on the operator and the result of that conversion, they will return either the original left-hand value or the right-hand value.
- Using this “feature” of JavaScript is BAD coding.
- Don’t do it!

Any Questions?

In-Class Problem

Evaluate these expressions as JavaScript would:

1. 2.99e2 + 1
2. 100 + 4 * 11
3. 100 + 4 * 11 ** 2
4. "In" + '-' + `Problem`
5. -101/0
6. "one or two\\nlines"
7. `half of 50 is: ${50 / 2}`
8. -(10 - 21)
9. (2>3) || (4<=4) && (3!=3)
10. 3>2 ? 1 : 2

If you have access to a web browser console, try to figure out the answer before checking it with the console.

11. Write an expression that finds the remainder of 100/9.

- You may work together.
- You may ask me or one of the SIs for help.
- Submit your solution as a PDF to Canvas before 11:59 PM today.
- You may leave when you are done.