

The JavaScript Language (Part 1)

Copyright © 2025 by
Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
 - A subset of JavaScript...
 - That is appropriate for COS 333...
 - Through example programs

Agenda

- **Overview**
- Setup
- Simple programs
- Functions
- Standard library
- Data types and operators
- Terminal I/O
- Exceptions
- Statements

Overview



Brendan
Eich

Overview: Motivation

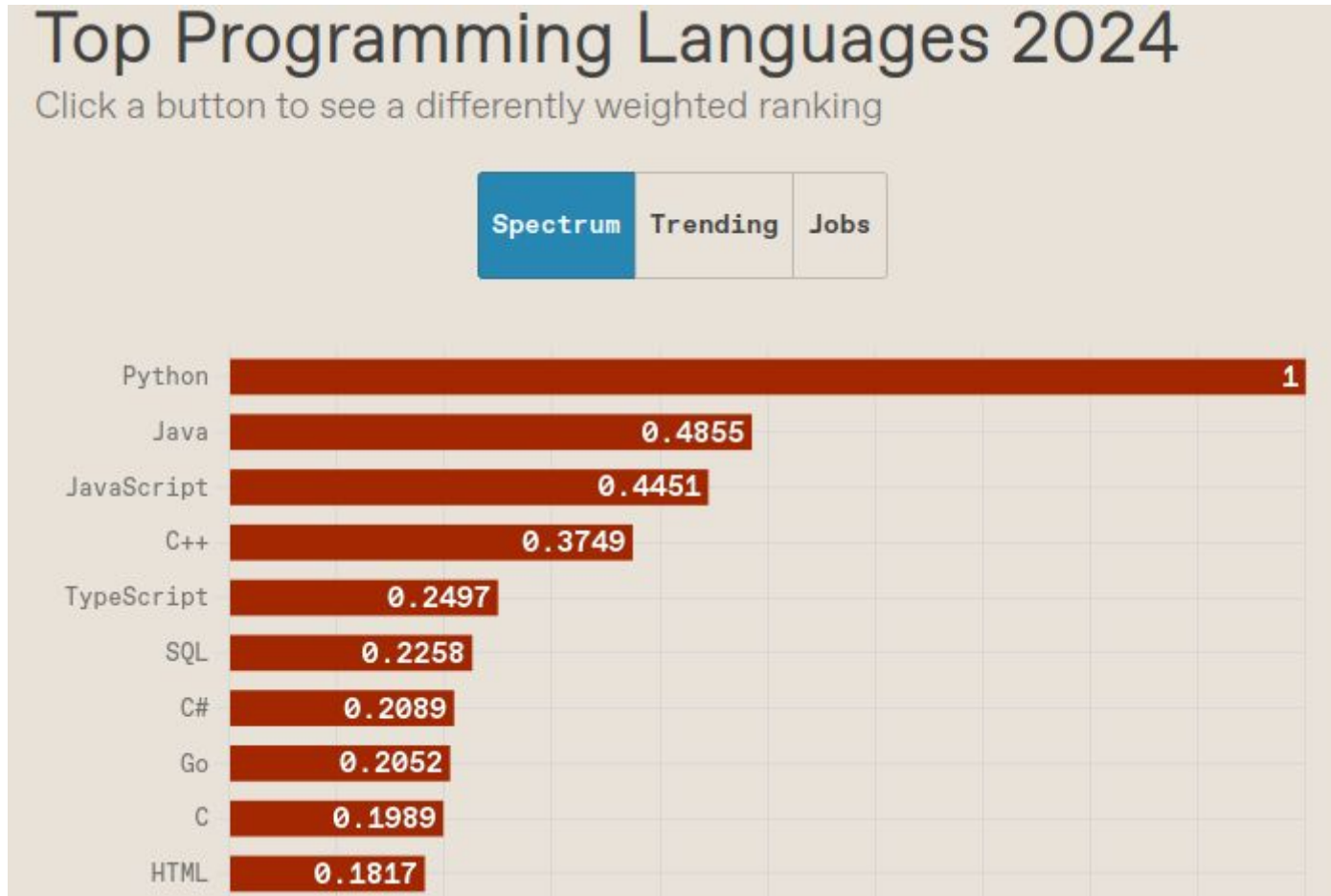
- **Question:**
 - Why study JavaScript?
- **Answer: ...**

Overview: Motivation

Rank	Language	Ratings
1	Python	21.9%
2	C++	11.6%
3	Java	10.5%
4	C	8.4%
5	C#	5.6%
6	JavaScript	3.5%
7	Visual Basic	2.4%
8	Go	2.0%
9	FORTRAN	1.8%
10	Delphi/Object Pascal	1.7%
11	SQL	1.6

<https://www.tiobe.com/tiobe-index/> on 10/20/24

Overview: Motivation



<https://spectrum.ieee.org/the-top-programming-languages-2024>

Overview: Motivation



<https://spectrum.ieee.org/the-top-programming-languages-2024>

Overview: Motivation



<https://spectrum.ieee.org/the-top-programming-languages-2024>

Overview: Motivation

Website	Front End	Back End
Google	JavaScript, TypeScript	C, C++, Go, Java, Python, Node
Facebook	JavaScript, TypeScript	Hack, Python, C++, Java, Erlang, D, Haskell
YouTube	JavaScript, TypeScript	Python, C, C++, Java, Go
Yahoo	JavaScript	PHP
Etsy	JavaScript	PHP
Amazon	JavaScript	Java, C++, Perl
Wikipedia	JavaScript	PHP
Fandom	JavaScript	PHP
X	JavaScript	C++, Java, Scala, Ruby
Bing	JavaScript	C++, C#
eBay	JavaScript	Java, JavaScript, Scala

<https://en.wikipedia.org/wiki/>

Programming_languages_used_in_most_popular_websites (10/20/24)

Overview: ECMAScript

- JavaScript is an implementation of...
- ***ECMAScript***
 - By **E**uropean **C**omputer **M**anufacturer's **A**ssociation
 - Latest specification:
 - <https://tc39.es/ecma262/>

Overview: ECMAScript

Year	Name	Description
1997	<i>ECMAScript1</i> <i>ES1</i>	First edition
1998	<i>ECMAScript2</i> <i>ES2</i>	Editorial changes only
1999	<i>ECMAScript3</i> <i>ES3</i>	Added regular expressions Added try/catch
	<i>ECMAScript4</i> <i>ES4</i>	Never released

https://www.w3schools.com/js/js_versions.asp

Overview: ECMAScript

Year	Name	Description
2009	<i>ECMAScript5</i> <i>ES5</i>	Added "strict mode" Added JSON support Added String.trim() Added Array.isArray() Added Array Iteration Methods
2011	<i>ECMAScript5.1</i> <i>ES5.1</i>	Editorial changes
2015	<i>ECMAScript6</i> <i>ECMAScript2015</i> <i>ES6</i>	Added let and const Added default parameter values Added Array.find() Added Array.findIndex() Added classes Added promises

https://www.w3schools.com/js/js_versions.asp

Overview: ECMAScript

Year	Name	Description
2016	<i>ECMAScript7</i> <i>ECMAScript2016</i> <i>ES7</i>	Added exponential operator (**) Added Array.prototype.includes
2017	<i>ECMAScript8</i> <i>ECMAScript2017</i> <i>ES8</i>	Added string padding Added new Object properties Added Async functions Added Shared Memory
2018	<i>ECMAScript9</i> <i>ECMAScript2018</i> <i>ES9</i>	Added rest / spread properties Added Asynchronous iteration Added Promise.finally() Additions to RegExp

https://www.w3schools.com/js/js_versions.asp

Overview: ECMAScript

Year	Name	Description
2019	<i>ECMAScript10</i> <i>ECMAScript 2019</i> <i>ES10</i>	Added Array.prototype.flat Added Array.prototype.flatMap Changed Array.sort and Object.fromEntries
2020	<i>ECMAScript11</i> <i>ECMAScript2020</i> <i>ES11</i>	Added BigInt Added null coalescing syntax
2021	<i>ECMAScript12</i> <i>ECMAScript2021</i> <i>ES12</i>	Added replaceAll for strings Added Promise.any Added AggregateError

https://www.w3schools.com/js/js_versions.asp

Overview: ECMAScript

Year	Name	Description
2022	<i>ECMAScript13</i> <i>ECMAScript2022</i> <i>ES13</i>	Added top-level await Added private class fields
2023	<i>ECMAScript14</i> <i>ECMAScript2023</i> <i>ES14</i>	Added functions to Array.prototype Added #! support
2024	<i>ECMAScript15</i> <i>ECMAScript2024</i> <i>ES15</i>	

https://www.w3schools.com/js/js_versions.asp

Overview: Transpiling

- **Problem:**
 - You create JavaScript code that uses some ES6 syntax
 - You want your code to be runnable on all browsers
 - Some old-but-still-widely-used browsers can interpret only ES5 syntax

Overview: Transpiling

- **Solution:** *Transpile*
 - You transpile your code from ES6 to ES5
 - You provide your code to browsers as ES5
 - Example transpiler: *Babel*

Overview: Polyfilling

- **Problem:**
 - You create some JavaScript code that uses ES5 syntax, but also uses some new modules – modules that are available only on new (post-ES5) browsers
 - You want your code to be runnable on ES5 browsers

Overview: Polyfilling

- **Solution: *Polyfill***
 - You polyfill those new modules
 - You compose your own versions of those new modules and include them in your code
 - Or better...
 - You include in your code an already vetted set of polyfills

Overview: Learning JavaScript

- **Commentary**

- To learn ES_n , you must also learn ES_{n-1} , ES_{n-2} , ..., ES_1
- JavaScript is a particularly difficult language to master

Overview: Running JavaScript

- Options for running JavaScript:
 - In browser (soon)
 - Via **Node.js** (now)

Overview: Node.js



Ryan
Dahl

Agenda

- Overview
- **Setup**
- Simple programs
- Functions
- Standard library
- Data types and operators
- Terminal I/O
- Exceptions
- Statements

Setup

- **Step 1: Install Node.js and npm**
 - Linux
 - Use your package manager to install the node and npm packages
 - Mac
 - Use Homebrew to install Node.js by issuing this command:
 - `brew install node`
 - MS Windows
 - Browse to <https://nodejs.org/en/download>
 - Download an appropriate .msi file
 - In Windows Explorer, double click on the .msi file
 - Use the installation defaults

Setup

- **Step 2: Install Node.js modules**
 - At shell prompt
 - cd to the directory that contains JavaScript code
 - **Install the `readline-sync` module**
 - `npm install readline-sync`

Agenda

- Overview
- Setup
- **Simple programs**
- Functions
- Standard library
- Data types and operators
- Terminal I/O
- Exceptions
- Statements

Simple Programs

- See **hello1.js**

```
$ node hello1.js  
hello, world  
$
```

Simple Programs

- See **hello2.js**

```
$ node hello2.js  
hello, world  
$
```

Agenda

- Overview
- Setup
- Simple programs
- **Functions**
- Standard library
- Data types and operators
- Terminal I/O
- Exceptions
- Statements

Functions

- See **square.js**

```
$ node square.js  
25  
25  
25  
25  
25  
$
```

Aside: “Never Fail” Design

- “Never fail”
 - Pervasive JavaScript design philosophy

```
function square1(i) {  
    return i * i;  
}  
...  
n = square1(5);      // What happens?  
n = square1(5, 6);  // What happens?  
n = square1();       // What happens?
```


Aside: The Arguments Array

- The arguments array

```
function sumall() {  
    let sum = 0;  
    for (let i = 0; i < arguments.length; i++)  
        sum += arguments[i];  
    return sum;  
}  
...  
n = sumall();           // 0  
n = sumall(5);         // 5  
n = sumall(5, 6, 7);   // 18
```

Agenda

- Overview
- Setup
- Simple programs
- Functions
- **Standard library**
- Data types and operators
- Terminal I/O
- Exceptions
- Statements

Standard Library

- See **squareroot.js**

```
$ node squareroot.js  
1.4142135623730951  
$
```

Full list of built-in objects in ES15:

<https://tc39.es/ecma262/>

Full list of built-in objects in Mozilla:

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

Agenda

- Overview
- Setup
- Simple programs
- Functions
- Standard library
- **Data types and operators**
- **Terminal I/O**
- Exceptions
- Statements

Data Types and Operators

- See **circle1.js**

```
$ node circle1.js
Enter the circle's radius:
5
A circle with radius 5 has diameter 10
and circumference 31.41592653589793.
$ node circle1.js
Enter the circle's radius:
1
A circle with radius 1 has diameter 2
and circumference 6.283185307179586.
$
```

Data Types and Operators

Data Type	Size	Example Literals
Boolean	1 byte	false, true
String	(varies)	"hi", 'hi'
Number	8 bytes	123, -123, 0.0, 1.23, 1.23e4

All numbers are stored as IEEE 754 floating point

Data Types and Operators

- Type conversion functions
 - `Number(x)`
 - `Boolean(x)`
 - `String(x)`

Data Types and Operators

- Strong suggestion
 - **Use the type conversion functions** to avoid mixed-type expressions!

Recommended viewing:

<https://www.destroyallsoftware.com/talks/wat>

Data Types and Operators

Operators	(Priority) Meaning
(...)	(1) Grouping
x.y	(2) Member access
x[y]	(2) Computed member access
new ...	(2) New
f(...)	(2) Function call
x?.y	(2) Optional chaining
new	(3) New without operator list
x++	(4) Postfix increment
x--	(4) Postfix decrement

Data Types and Operators

Operators	(Priority) Meaning
!x	(5) Logical NOT
~x	(5) Bitwise NOT
+x	(5) Unary plus
-x	(5) Unary negation
++x	(5) Prefix increment
--x	(5) Prefix decrement
typeof x	(5) typeof
void x	(5) void
delete x	(5) delete
await x	(5) await
x**y	(6) Exponentiation

Data Types and Operators

Operators	(Priority) Meaning
x*y	(7) Multiplication
x/y	(7) Division
x%y	(7) Remainder
x+y	(8) Addition
x-y	(8) Subtraction
x<<y	(9) Bitwise left shift
x>>y	(9) Bitwise right shift
x>>>y	(9) Bitwise unsigned right shift
x<y	(10) Less than
x<=y	(10) Less than or equal to
x>y	(10) Greater than
x>=y	(10) Greater than or equal to
x in y	(10) In
x instanceof y	(10) Instance of

Data Types and Operators

Operators	(Priority) Meaning
x==y	(11) Equality
x!=y	(11) Inequality
x===y	(11) Strict equality
x!==y	(11) Strict inequality
x&y	(12) Bitwise AND
x^y	(13) Bitwise exclusive OR
x y	(14) Bitwise OR
x&&y	(15) Logical AND
x y	(16) Logical OR
x??y	(17) Nullish coalescing operator
x?y:z	(18) Conditional

Data Types and Operators

Operators	(Priority) Meaning
<code>x=y, x+=y, x-=y, x**=y, x*=y, x/=y, x%=y, x<<=y, x>>=y, x>>>=y, x&=y, x^=y, x =y, x&&=y, x =y, x??=y</code>	(18) Assignment
<code>yield x</code>	(19) Yield
<code>yield* x</code>	(20) Yield
<code>x, y</code>	(21) Sequence

Data Types and Operators

Expression	Value
<code>123 == '123'</code>	true
<code>123 == '+123'</code>	true
<code>123 === '123'</code>	false
<code>123 === '+123'</code>	false

Equality operator (==) does type conversion

Strict equality operator (===) suppresses type conversion

Recommendation: Always use === instead of ==

Recommendation: Never mix types!!!

Terminal I/O

Reading from stdin:

```
const readlineSync = require('readline-sync');  
...  
str = readlineSync.question(str);  
str = readlineSync.question();
```

Terminal I/O

Writing to stdout:

```
process.stdout.write(strExpr) ;  
console.log(expr) ;
```

Writing to stderr:

```
process.stderr.write(strExpr) ;  
console.error(expr) ;
```


Agenda

- Overview
- Setup
- Simple programs
- Functions
- Standard library
- Data types and operators
- Terminal I/O
- **Exceptions**
- Statements

Exceptions

- Recall circle1.js

```
$ node circle1.js
Enter the circle's radius:
5
A circle with radius 5 has diameter 10
and circumference 31.41592653589793.
$ node circle1.js
Enter the circle's radius:
xyz
A circle with radius NaN has diameter NaN
and circumference NaN.
$ node circle1.js
Enter the circle's radius:
A circle with radius 0 has diameter 0
and circumference 0.
$
```

Exceptions

- See **circle2.js**

```
$ node circle2.js
Enter the circle's radius:
5
A circle with radius 5 has diameter 10
and circumference 31.41592653589793.
$ node circle2.js
Enter the circle's radius:
xyz
Error: Not a number
$ node circle2.js
Enter the circle's radius:
Error: Missing number
$
```

Exceptions

JavaScript standard exceptions:

```
Error  
  EvalError  
  RangeError  
  ReferenceError  
  SyntaxError  
  TypeError  
  URIError  
  AggregateError  
  InternalError
```

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

Agenda

- Overview
- Setup
- Simple programs
- Functions
- Standard library
- Data types and operators
- Terminal I/O
- Exceptions
- **Statements**

Statements

- See **euclidclient1.js**

```
$ node euclidclient1.js
Enter the first integer:
8
Enter the second integer:
12
gcd: 4
lcm: 24
$
```

Statements

Compound statement

```
{  
    statement1;  
    statement2;  
    ...  
}
```

Statements

Variable definition statements

```
let name = expr;  
const name = expr;
```

```
// Pre-ES6  
var name = expr; // name is "hoisted"
```


Aside: Hoisting

```
'use strict';  
...  
function f() {  
    ... // Not OK to use n here.  
    let n = 5;  
    ... // OK to use n here.  
}
```

```
'use strict';  
...  
function f() {  
    ... // OK to use n here.  
    ... // But its value is undefined.  
    var n = 5;  
    ... // OK to use n here.  
}
```

Statements

Function call statement

```
f(expr, expr, ...);
```

return statement

```
return;
```

```
return expr;
```

Statements

if statement

```
if (expr)  
    statement;  
else  
    statement;
```

false, 0, "", '', null, undefined,
NaN mean logical FALSE

Any other value indicates logical TRUE

Statements

while statement

```
while (expr)  
    statement;
```

false, 0, "", ', null, undefined,
NaN mean logical FALSE

Any other value indicates logical TRUE

Statements

do...while statement

```
do  
    statement;  
while (expr);
```

false, 0, "", '', null, undefined,
NaN mean logical FALSE

Any other value indicates logical TRUE

Statements

for statements (by example)

```
for (let i = 0; i < 10; i++)  
  statement;
```

```
for (let i = 0; i < someArray.length; i++)  
  ...someArray[i]...
```

```
for (let element of someArray)  
  ...element...
```

```
for (let property in someObject)  
  ...property...
```

Statements

break statement

```
break;
```

continue statement

```
continue;
```

Statements

try statement

```
try
    statement;
catch (exception)
    statement;
```

throw statement

```
throw object;
```


Summary

- We have covered:
 - Overview
 - Setup
 - Simple programs
 - Functions
 - Standard library
 - Data types and operators
 - Terminal I/O
 - Exceptions
 - Statements