Introduction to Javascript

the annotated edition

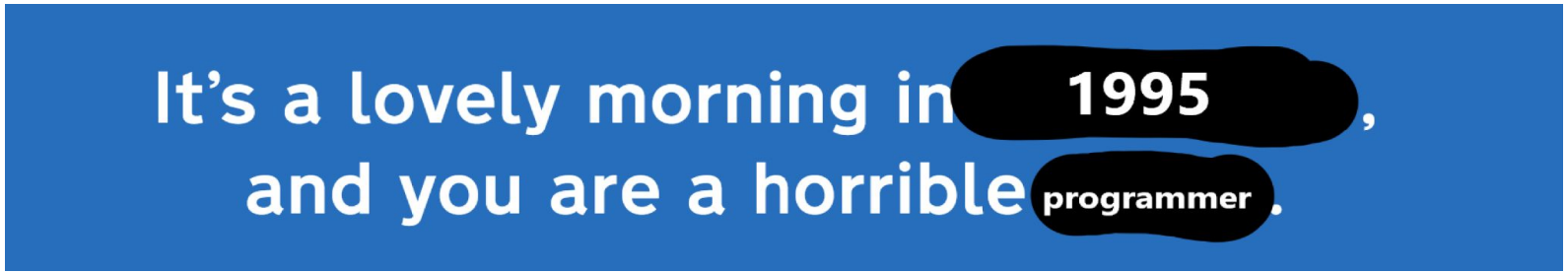
notes

- The final-ish speaker notes from this presentation are here
- I am always happy to help with js (or typescript) because I might as well put the cursed knowledge to use, message me in the groupme or wherever
- I'll also include the js code I put up on the screen in here

Outline

- History & Design
- Syntax
- Running Javascript
- Resources & Citations

History & Design



Design Requirements

- Like Java, but not too like Java
- Easy and accessible
- Lightweight
- Not compiled
- Deadline in 10 days :)

Result: JS is designed to *work*. It tries to fix your mistakes (nice for new programmers) and avoids throwing errors wherever it can, because if a tiny bit of your webpage doesn't work you don't want the rest to die on you.

Syntax

Basics

- Java-like (brackets, semicolons)
- Operators: **+**, **-**, *****, **/**, **%**, ******
- Logical: **&&** and **||**
- Comments are **//**
- Print: **console.log()**

```
console.log(1 - 2); // -1
console.log('a' + "b"); // "ab"
console.log("${5 % 3}, ${0/0}"); // "2, NaN"
console.log(typeof true); // boolean
```

Types

- Weakly typed
- Number: 64-bit double, NaN
- Strings: double or single quotes
- Boolean: **true**, **false**
- **undefined** and **null** (**undefined** is not-yet-initialized, **null** is deliberate non-value)

Variable declaration

- **`let`**: block scoped
- **`const`**: block scoped
- **`var`**: function scoped
- no keyword: globally scoped

```
let v1;
const v2 = 1;
var v3 = 2;
v4 = 3;
```

Variable declaration

- **`let`**: block scoped
- **`const`**: block scoped

```
let v1;
const v2 = 1;
```

NO.

ONLY USE LET AND CONST.

You will see `var` if you look for answers on the internet, because `let` and `const` are relatively new additions. Don't use `var`.

Arrays

- ordered list of elements
- mutable length
- elements can be any type
- indexing starts at 0
- access via `arr[i]`
- accessing an index past the end will return `undefined`, not an error

```
let arr = [12, "hey", 18.3, false];

arr.push(null);
arr; // [12, "hey", 18.3, false, null]
arr.pop(); // null
arr.length; // 4
arr[6]; // undefined
arr.slice(2,4); // [18.3, false]
arr.join(","); // "12,hello,18.3,false"
```

Dictionaries (“Objects”)

- called “objects” in JS
- unordered key/value pairs
- keys are strings (immutable)
- vals can be any type
- values can change
- access via `dict[“key”]` or `dict.key`

```
let dict = {
     "hello": "world",
     51: "stuff",
     thisworks: true,
     andthis: 50.3
};

dict.hello = "world!!";
dict["newthing"] = "newval";
newthing in dict; // true
```

Control Flow

- if/else
- for
- while
- do/while, switch

```
if (7 <= 12) {
    // do something
} else if (0 > 1) {
    // do a different thing
} else {
    // do something else
}
```

```
while (true) {
    // do stuff
    break;
}
```

```
let arr = [5,6,7]
for (let i = 0; i < 4; i++) {
    console.log(arr[i]);
} // 5, 6, 7, undefined
```

standard for loop

```
for (let i = 0; i < 4; i++) {
    console.log(arr[i]);
} // 5, 6, 7, undefined
```

```
let arr = [5,6,7]
let dict = {"key1": "val1", "key2": 2}
```

for/in

```
for (const k in arr) {
    console.log(k);
} // "0", "1", "2"

for (const m in dict) {
    console.log(m);
} // "key1", "key2"
```

for/of

```
for (const j of arr) {
    console.log(j);
} // 5, 6, 7

for (const [key, val] of Object.entries(dict)) {
  console.log(key, val);
} // key1 val1  key2 2
```

Functions

- Pass by value
- First class

```
function square(num) {
     // other stuff
    return num * num;
}
```

```
const factorial = function fac(n) {
    if (n < 2) {
        return 1
    }
    return n * fac(n - 1) ;
}
factorial(5) // 120
```

Extras: arrow function & `.map()`

```
const square = (num) => {num * num}


arr.map(square);
arr.map(n => n*n);
// both return [25, 36, 49]
```

Variable Scoping

- **`let`**: block scoped
- **`const`**: block scoped
- **`var`**: function scoped
- no keyword: globally scoped

```
function myFun() {
  let deptt = 0;
  // do other stuff
  depttt = deptt + 1;
}
```

```
function scoping() {
  if (true) {
    v1 = "a";
    var v2 = "b";
    let v3 = "c";
    const v4 = "d";
  }
  console.log(v1, v2); // works
}
console.log(v1); // also works
```

Miscellaneous Nonsense

Semicolons

```
let b = 1, c = 2, d = '3', e = '4';

let a = b + c
(d + e).length;
// JS reads a = b + c(d+e).length;
// and fails since c() isn't a function


function square(num) {
    return
     {num * num}
} // returns nothing because JS
inserts ; after return
```

Semicolons are inserted at the end of a line whenever JS thinks they should be, but sometimes it gets it wrong, so just always use them.

Sort

```
> [5, 12, 9, 2, 18, 1, 25].sort();
< [1, 12, 18, 2, 25, 5, 9]
```

(it sorts as if they're strings, bafflingly)

Types, Redux

Equality, == vs ===

```
> 0 == "0"
< true

> 0 === "0"
< false
```

```
if (b == 1 && c == 2) {
    console.log(b + c);
}

if (b === 1 && c === 2) {
    console.log(b + c);
}
```

All numbers are floats

```
> 0.1 + 0.2
< 0.30000000000000004

> 9999999999999999
< 10000000000000000
```

For very big integers, use the BigInt type.

Unary +

```
> 9 + "1"
< "91"

> 9 - "1"
< 8

> +"9" + 1
< 10
```

More typing nonsense, for fun

```
> [] + []
< ""

> {} + []
< 0

> [] + {}
< "[object Object]"

> {} + {}
< NaN

> (!+[]+[]+![]).length
< 9

> (!+[]+![]).length
< undefined
```

```
> j = "1"
> j += 1
< "11"

> ++j
< 12


> i = "1"
> i++
< 1
```

```
> true == 1
< true

> true === 1
< false

> true + true === 2
< true
```

TypeScript

- Not a distinct language - compiles down to JS
- Basically makes JS strongly typed (and also doesn't let you declare variables without a `let`/`const`/`var` keyword)
- Download: `npm install -g typescript`
- Compile: `tsc yourfile.ts`

```
let tsArr: number[] = [5, 6, 7]; // trying to add a string would now give an error
// (you can also do number|null|boolean[] for multiple possible types)
for (let j: number = 0; j < 10; j++) {
  console.log(tsArr[j]);
}
```

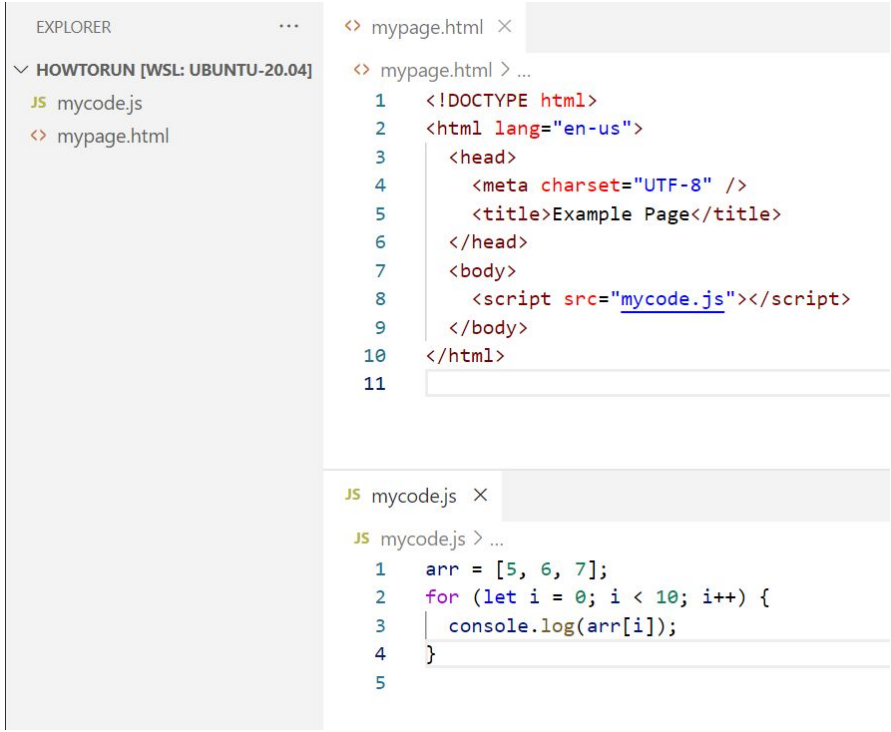
Running JavaScript

Basic HTML page

```
<!DOCTYPE html>
<html lang="en-us">
  <head>
    <meta charset="UTF-8" />
    <title>Example Page</title>
  </head>
  <body>
    <-- anything else goes here -->
    <script src="mycode.js"></script>
  </body>
</html>
```

mycode.js

```
arr = [5,6,7]
for (let i = 0; i < 10; i++) {
    console.log(arr[i]);
}
```



Instructions on Running JS

- Make a html file and a js file
- Include `<script src="pathtojsfile"></script>` at the bottom of the body of the html file. (If using multiple scripts, put them in the order you want them to run.)
- Open your html page in the browser of your choice.
- Press **`ctrl+shift+I`** or **`cmd+shift+I`** (or go to your browser settings > More Tools > Developer Tools) to open the console.
- Refreshing the page will update to the latest saved version of your js file.

Resources

- Documentation: https://developer.mozilla.org/en-US/docs/Web/JavaScript
- Quick reference: https://learnxinyminutes.com/docs/javascript/
- Typescript: https://www.typescriptlang.org/
- Extensions for VSC
  - Prettier - Code formatter v9.2.0 (will auto-format your code)
  - Babel JavaScript v0.0.36 (provides syntax highlighting for JS)

Sources

History: computer.org/csdl/magazine/co/2012/02/mco2012020007/13rRUy08MzA

Syntax: developer.mozilla.org/en-US/docs/Web/JavaScript

Problems: medium.com/javascript-non-grata/the-top-10-things-wrong-with-javascript-58f440d6b3d8