Check-In Code: javascript!=java

TypeScript

Basically, JavaScript that doesn't suck

acmurl.com/typescript-slides

hack ACM Hack

JavaScript Crash Course

OK, it's not THAT bad actually 🤔

hack ACM Hack

What is JavaScript?

```
console.log("Hello World!");
```

- NOT Java
- Web client language (originally)
  - Make things interactive on websites!
- Now: also use server-side and natively
  - Node.js, Electron apps, React Native
- Interpreted, imperative language
  - But it also has elements of OOP and functional programming

Basic Syntax - Variables

- Declare variables with `var`, `let`, or `const`

```
var globalVar = 5; // can be accessed anywhere!
{
    let localVar = 420; // can only be accessed inside this block
}
console.log(localVar) // will not work
// also semi-colons are optional, but recommended
const constant = 100; // constant and cannot be reassigned
constant = 3847; // will not work
```

hack ACM Hack

Basic Syntax – Variables

- 6 primitive types: undefined, Boolean, Number, String, BigInt, Symbol
- Other types: objects, arrays, functions
- However, variables are not restricted to be a single type!

```
var foo = false;
foo = 5; // foo changes from a boolean to an int
console.log(foo); // still works
foo = () => {console.log("bar")}; // foo is now a function we can call
foo(); // also works
```

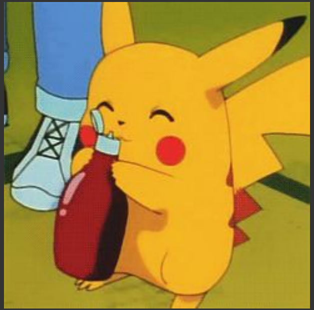
- This can lead to some WEIRD stuff which we'll see later

hack ACM Hack

Basic Syntax - Objects

- JS objects are more like hashmaps instead of traditional OOP objects
- A collection of Key-value pairs

```
const pokemon = {
    name: "Pikachu", // key is name, value is "Pikachu"
    type: "Electric",
    level: 20,
    likes: "ketchup",
    evolution: {
        pokemon: "Raichu",
        method: "Thunderstone"
    } // objects can be nested
};
```



Basic Syntax - Logic

- If-statements are pretty much identical to C and Java
- Use else if and else to handle different branches of logic

```
if (condition) {
    doSomething();
} else if (anotherCondition) {
    doSomethingElse();
} else {
    doAnotherThing();
}
```

Basic Syntax - Logic

- 3 delicious flavors of for-loops

```
for (let i = 0; i < 4; i++) {
    // C-style for-loop
}

for (let index in list) {
    // for-in loop - loop through keys/indices
    // no guarantee that the items are in order
}

for (let property of object) {
    // for-of loop - loop through the values
    // goes in order unlike for-in
}
```

- There's also the interesting `forEach` function that we won't go into today

hack ACM Hack

Basic Syntax - Logic

- While and do-while loops are also very similar to C and Java

```
while (condition) {
    // do something until condition is false
}

do {
    // do something at least once,
    // then repeat until condition is false
} while (condition);
```

ACM Hack

Functions

- 3 ways to declare, and they are *generally* interchangeable

```
function sum(x, y) {
    return x + y;
}

// Anonymous function
let multiply = (x, y) => {
    return x * y;
};

// Alternative way to declare
let multiply = function(x, y) => {
    return x * y;
};
```

ACM Hack

Applying our knowledge

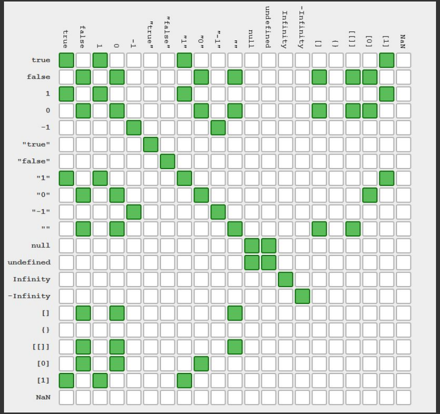
Let’s try an easy leetcode problem!

leetcode.com/problems/two-sum/

Now for the problems of JS...

Problems with JS

- Weakly typed means anything is possible
  - Completely ok to reassign a variable as a completely different type
  - Your objects can have any structure, and this structure can change at any time
- Type Coercion
  - If the types are wrong, JS will try to do the conversion itself to make it work
  - See jsfuck.com for an extreme example of this



	true	false	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	null	undefined	Infinity	-Infinity	[]	{}	[[]]	[0]	[1]	NaN
true	🟩		🟩					🟩												🟩	
false		🟩		🟩					🟩		🟩					🟩		🟩	🟩		
1	🟩		🟩					🟩												🟩	
0		🟩		🟩					🟩		🟩					🟩		🟩	🟩		
-1					🟩					🟩											
"true"						🟩															
"false"							🟩														
"1"	🟩		🟩					🟩												🟩	
"0"		🟩		🟩					🟩										🟩		
"-1"					🟩					🟩											
""		🟩		🟩							🟩					🟩		🟩			
null												🟩	🟩								
undefined												🟩	🟩								
Infinity														🟩							
-Infinity															🟩						
[]		🟩		🟩							🟩										
{}																					
[[]]		🟩		🟩							🟩										
[0]		🟩		🟩					🟩												
[1]	🟩		🟩					🟩													
NaN																					

Hello world with JSFuck

```
[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!
+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[
]]][([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!
+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])
[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+([![]]+[
][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]
+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[+!+[]+[+[]]]+([][[]]
+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[]
)[+!+[]]+([][[]]+[])[+[]]+([][(![]+[])[+[]]+([![]]+[][[]])[+
!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[
]+!+[]+!+[]]+(!![]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]+[])
[+[]]+(!![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![
]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!
```

ACM Hack

Enter TypeScript!

hack ACM Hack

What the heck is TypeScript?

- JavaScript but with static types
- A strict **superset** of JS – all JS code can still run in TS
- Compiled into JavaScript, which is then run by the browser

Why TypeScript?

- Type safety for everything
  - Know exactly what goes in and out of your functions and what types variables can be
  - Catch bugs before they show up in runtime
- Smart code completion
  - Know exactly how your objects are structured
- Newer JS features in older browsers
  - TS can be compiled to any version of JS
  - Your website can (probably) still function in Internet Explorer 6

Setup (optional)

- Only if you want to compile TypeScript locally
- Install **Node** and **npm**: nodejs.org/en/download/
- Install TypeScript with command:

```
npm install -g typescript
```

- Compiling a .ts file:

```
npx tsc index.ts
```

- Alternatively, just use the TS playground: typescriptlang.org/play
  - This workshop: get familiar with the language itself, without worrying about the tooling

ACM Hack

Setup (optional)

- tsconfig.json - specify files to compile and compiler options
- The tsc command will look at this file to know what to do
- Example (copied from TS website)

```json
{
  "compilerOptions": {
    "module": "system",
    "noImplicitAny": true,
    "removeComments": true,
    "preserveConstEnums": true,
    "outFile": "../../built/local/tsc.js",
    "sourceMap": true
  },
  "include": ["src/**/*"],
  "exclude": ["node_modules", "**/*.spec.ts"]
}
```

Basic Types

- Types are declared with a colon (:) after the variable name

```
let foo: string = "Hello world!";
let bar: string | number; // bar can be a string OR number
```

- Here’s a list of some basic/primitive types
  - any – special type, means the variable can be “any” type like in JS
  - unknown – used if you don’t know the type at compile time
  - boolean – true/false
  - string – a sequence of characters
  - number – any number, can be integer or float (decimal)

More types

- Arrays – declare with name of type and brackets []

```
let nums: number[] = [420, 69, 1337]; // an array of numbers
```

- Tuples – fixed-length array with known types

```
let event: [string, number] = ["TS workshop", 10];
```

- Enums – a fixed set of possible values

```
enum Community {
    Hack,
    Innovate,
    Cyber,
    Design
}
let heck: Community = Community.Hack;
```

ACM Hack

Objects

- We can use the object type to define anything that's not a primitive type

```
let foo: object = {bar: 384};
foo = {somethingElse: true}; // still legit
```

- Notice how in this example, foo can still be any shape/structure it wants
- There's a better way to define structure: **interfaces**

Interfaces

- Kinda like Java interfaces, but simpler
- Define the name and types of properties in your object

```
interface AcmMember {
    name: string;
    points: number;
    community?: Community; // optional property
    likesToCode: boolean;
}

let garrett: AcmMember = {
    name: "Garrett",
    points: 420,
    community: Community.Hack,
    likesToCode: true
};
```

ACM Hack

Functions

- We can specify the types of the parameters and the return type
    - A new type appears! void specifies that nothing is returned

```
function failClasses(classesToFail: string[]): void {
    // try not to fail but fail anyway :(
}

let isEven: (num: number) => boolean = (num: number) => {
    if (num === 1)
        return false;
    else if (num === 2)
        return true;
    else if (num === 3)
        return false;
    // not gonna write the rest...
}
```

More Leetcoding

Now let's code this in TypeScript!
leetcode.com/problems/two-sum/

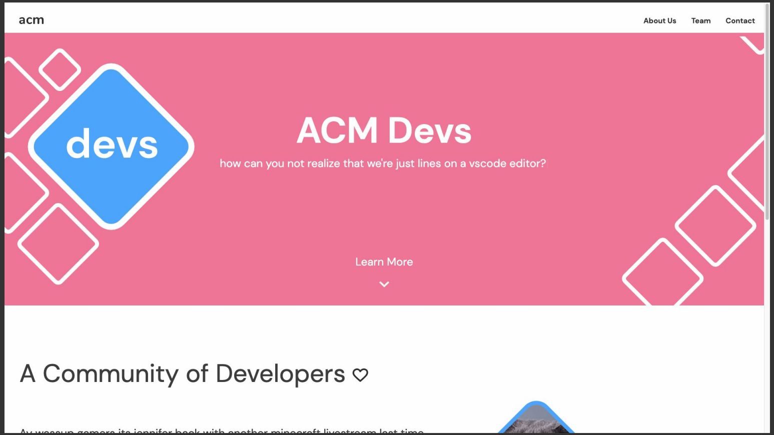
Applications of TypeScript

What can you actually do?

ACM Hack

Client-side Example

- ACM Static Template: github.com/acmucsd/static-template
- Uses React with TypeScript
  - We know the exact type of each prop in each component



ACM Devs

how can you not realize that we're just lines on a vscode editor?

A Community of Developers ♡

ACM Hack

What about Server-side?

- ts-node allows you to compile and run TypeScript
  - npmjs.com/package/ts-node
- express-generator-typescript - template for Node/Express projects in TS
  - npmjs.com/package/express-generator-typescript

Additional Resources

TypeScript Handbook:

typescriptlang.org/docs/handbook/intro.html

hack ACM Hack