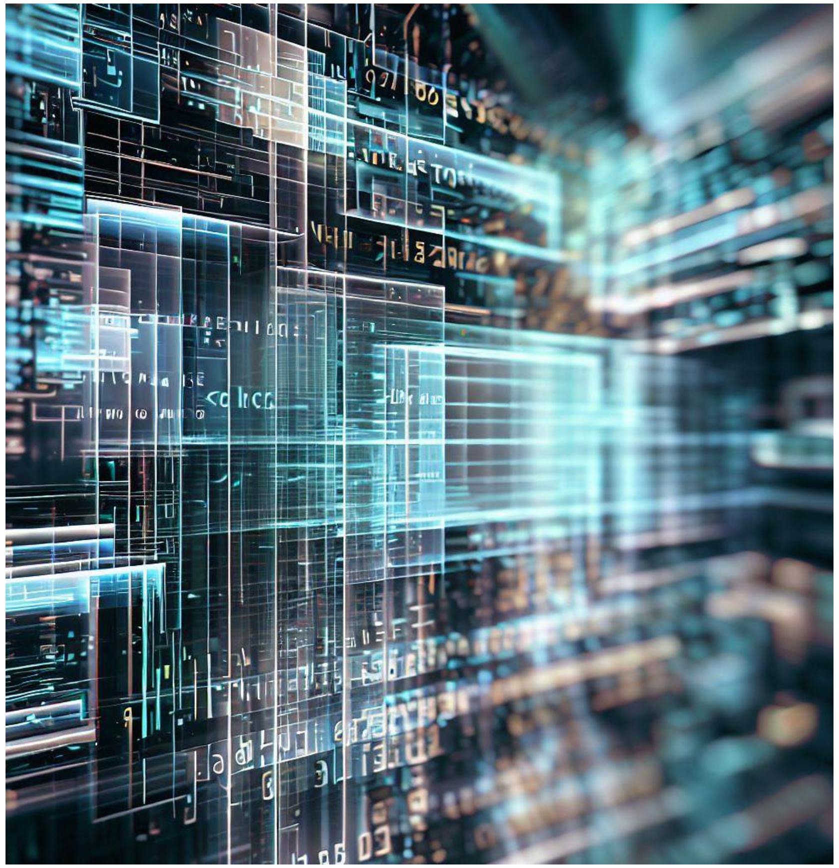
CTD Intro Week 1

Programming Fundamentals:

JavaScript Basics

Programming Fundamentals

- Write programs with future maintainers in mind
  - Clarity, simplicity, comments
  - Most programming involves updating/fixing code someone else wrote
- Understand the problem
  - Inputs, outputs, user interface, steps from inputs to outputs
- High-level steps
  - Pseudocode
- Divide and conquer
  - Code and test simpler components which taken together solve the problem
- DRY – Don’t Repeat Yourself
  - Capture reused components in one place
    - usually a function or method
  - Don’t copy and paste



JavaScript

- Javascript is not java!
- Invented in 1995 at Netscape
    - Called javascript because java was new, popular and exciting
        - But it is unrelated to java
- Standardized as ECMAScript (ECMA-262) in 1997
    - European Computer Manufacturers Association (ECMA)
- Important revisions
    - ES5 (2009)
        - Var for non-global scope, function scope only
    - ES6 (2015)
        - Lexical (block) scope with let and const
        - class, module
        - Anonymous function shorthand, arrow notation (a, b) => { }
- Most popular computer language
- Lots of built-in capabilities
- Rich set of packages available
- Highly optimized, good performance
- In all browsers
- Node.js for servers and command line apps (Google javascript engine)

Some JavaScript facts

- First class functions
  - Functions can be assigned to variable (not true in every language!)
    - myObject.func returns the function
    - myObject.func() calls the function
- Very permissive
  - Doesn’t, by default report errors on many things which are probably wrong
  - Automatic conversions between types
  - Doesn’t check type or number of function arguments
    - Fills in with undefined values if necessary
  - Typescript was invented to fix this
- Convenient object model
  - Any mix of indexed arrays […] and associative arrays {…}
    - Mixed datatypes
  - So simple and useful it became a data exchange standard
    - JavaScript Object Notation (JSON)

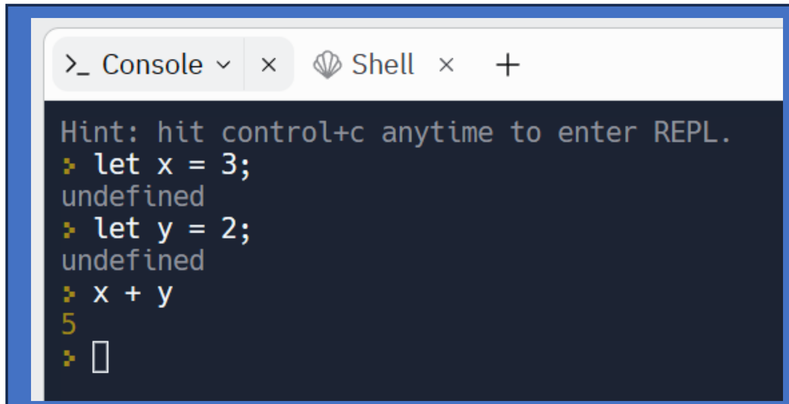
The 8 Data Types in JavaScript

- Number
  - Double precision floating point, also used to represent integers
- String
  - Characters (and anything representable by Unicode)
- Boolean
  - True/false
- Undefined
  - Its type is ‘undefined’, Lack of a value, never assigned, tests as false
- Null
  - Its type is ‘object’, absence of an object, tests as false
- Object
  - Combinations of indexed and associative arrays
- Symbol
  - Unique, immutable value to use as a key for objects
- BigInt
  - Integers with unlimited precision (subject to resource limitations)
- Use typeof(<name>) to find out what a variable’s datatype is
  - Where <name> is a placeholder for any variable name or literal

Behind the Scenes

- Don’t need to know details of compilation and data types for the intro class!
  - Context and background
- Different datatypes have varying internal representations
- Use explicit conversions between datatypes

Software to Hardware

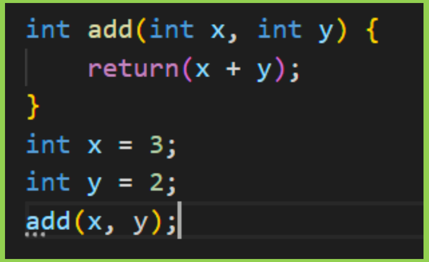


```
Hint: hit control+c anytime to enter REPL.
› let x = 3;
undefined
› let y = 2;
undefined
› x + y
5
›
```

Interpreter

```
// This function adds two numbers and returns the result
function add(x, y) {
  // Push the arguments to the stack
  LdaSmi x
  Star r0
  LdaSmi y
  Star r1
  // Add the two values and return the result
  Add r0, r1
  Return
}
```

Javascript Virtual Machine



```
int add(int x, int y) {
    return(x + y);
}
int x = 3;
int y = 2;
add(x, y);
```

Compiler – C++

```
define double @add(double %x, double %y) {
  ; Add the two values and return the result
  %1 = fadd double %x, %y
  ret double %1
}
```

LLVM – Low Level Virtual Machine

```
addsd %xmm1, %xmm0 ; add xmm1 to xmm0 and store in xmm0
```

Assembler

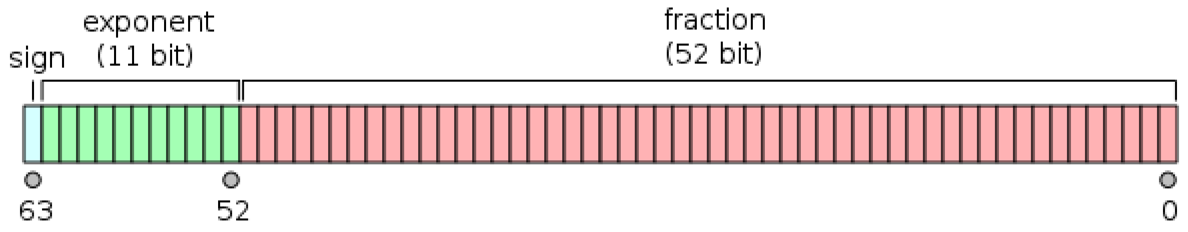
```
f2 0f 58 c1
```

Hexidecimal Machine Code

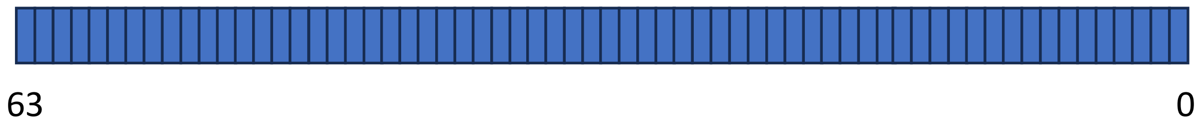
```
11110010 00001111 01011000 11000001
```

Binary Machine Code

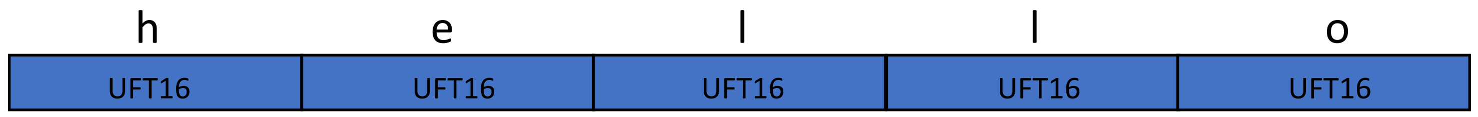
Data Types Generically and in JavaScript



Floating Point – Number in JavaScript

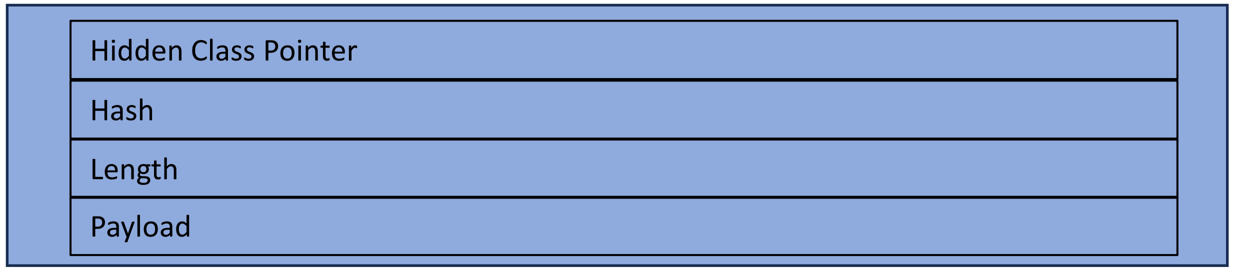


Integer – Not used for JavaScript Numbers



Can also be UTF8, UTF32…

String – Characters in memory



Object Header

Objects – combinations of indexed and associative arrays

Explicit Conversions

- Number()
- String()
- parseInt()
- parseFloat()
- Boolean()
- <num>.toFixed(<decimalPlaces>)
    - String with fixed formatting
- Math.floor()
- Math.round()

JavaScript Syntax

- [airbnb/javascript: JavaScript Style Guide (github.com)]
  - Common set of conventions for code
- Use lesson content for a comprehensive syntax guide
  - Just highlights in the presentation
- Variable naming
  - Use camelCase, longNameCamelCase
  - Use UPPERCASE constants to remember values (sometimes globally)
    - Const FINESTRUCTURECONSTANT = 1.0/137.0;
  - Case sensitive: thisVar is not the same as thisvar
  - Reserved words, can’t use parts of the language as variable names
    - let let = 5; // won’t work

JavaScript Syntax – Strings

- Literals
  - ‘a string’ and “a string” the same except
    - Need to escape quotes inside strings, so
      - “isn’t” works without escaping
      - Vs ‘isn\’t’
    - JSON only accepts “”
  - `allows interpolation of ${anyVar}` // string representation is inserted
    - Can also interpolate expressions
- Concatenate using ‘+’
  - Foobar = ‘foo’ + bar’;

More Syntax

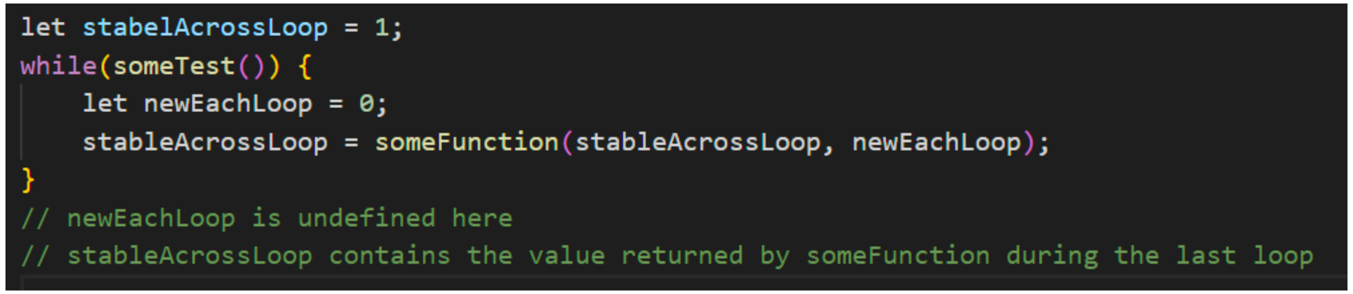
- Operator precedence
  - When in doubt, use ( ) to group
- ‘;’ at the end of every statement line
  - Not worth worrying about where they can be left out
- Functions and methods
  - More in a later lesson
  - Variables can contain functions (first class functions)
    - Call by adding (), can contain an argument list (arg1, arg2…)
    - aFunction is a variable, running it returns the function
    - Running aFunction() calls the function
  - Methods are functions which are part of an object and which act on it
    - Let num = 63;
    - Num.toFixed()

Variables and Scoping

- Variables give a name to a value
  - The value will have a data type
- Variable scoping
  - Defines the places a variable name is recognized
- Global scope
  - Dangerous and not advised
  - Valid everywhere, changing a global might impact code anywhere which references it
- Lexical scope
  - Value is local to a block { … }
  - Preferred
  - Defined using let or const
- Function scope
  - Defined using var
  - Value is defined anywhere in the enclosing function
  - Obsolete in most cases. Use let and const.

Let and Const

- If you don’t expect the value change, use const
  - For single values it can’t be changed
  - For objects, it is a constant reference, which means it’s the same data structure, but the content can be changed.
- Use let for everything else
  - Lexically (block) scoped
  - Scope the variable as narrowly as possible, but not too narrowly



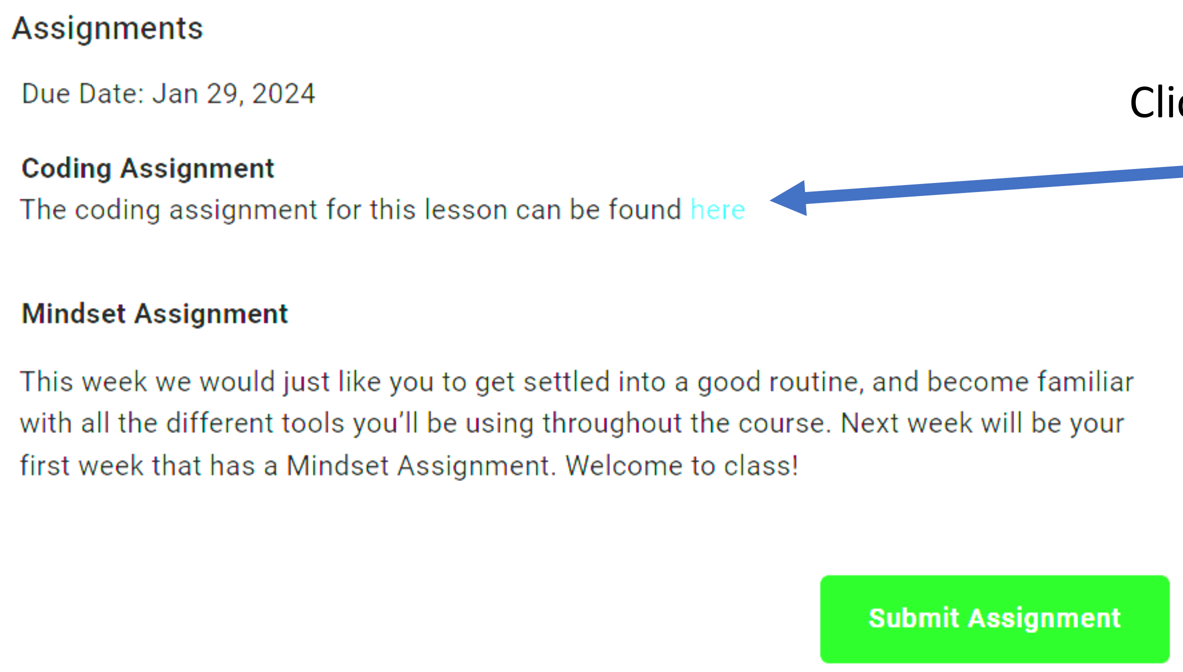
```
let stabelAcrossLoop = 1;
while(someTest()) {
    let newEachLoop = 0;
    stableAcrossLoop = someFunction(stableAcrossLoop, newEachLoop);
}
// newEachLoop is undefined here
// stableAcrossLoop contains the value returned by someFunction during the last loop
```

Truth and Equality

- Javascript does lots of automatic conversions
  - == can produce unexpected results
    - 0 == ‘’
    - null == undefined
    - [] == ![]
    - [2] == 2
    - \n == 0
  - Almost always, use ===, checks for same type and value
- What is truth?
  - true (false) – explicit
  - true: non-zero Number, non-empty string, any (even empty) object
- Logical operators &&, ||, ! To create logical expressions
  - Short circuit – only executes what is needed to determine the final value
    - false && ‘this never runs’
    - false || ‘this does run’

Coding Assignment

- The coding assignment is at the bottom of each lesson page



Assignments

Due Date: Jan 29, 2024

Coding Assignment
The coding assignment for this lesson can be found here

Mindset Assignment

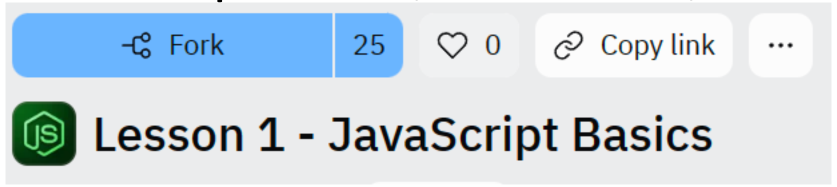
This week we would just like you to get settled into a good routine, and become familiar with all the different tools you'll be using throughout the course. Next week will be your first week that has a Mindset Assignment. Welcome to class!

Click here to go to the replit or github lesson

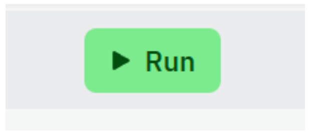
Click here for the assignment submission form

Using Replit

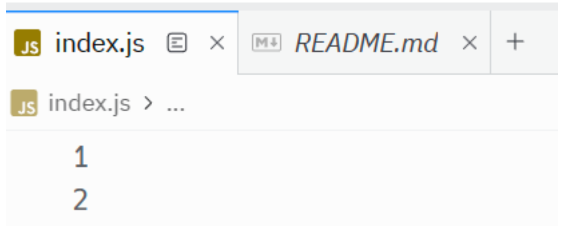
- REPL
  - Read Eval Print Loop – Read an expression, Evaluate it, Print the result
- Fork the lesson
  - Make your own copy



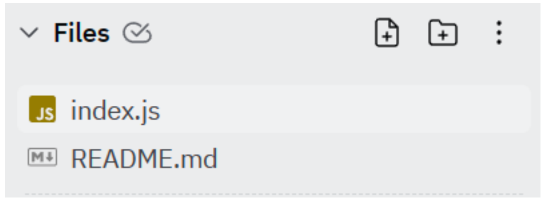
- Console tab
  - console.log(“message…”.) displayed after clicking the ‘Run’ button



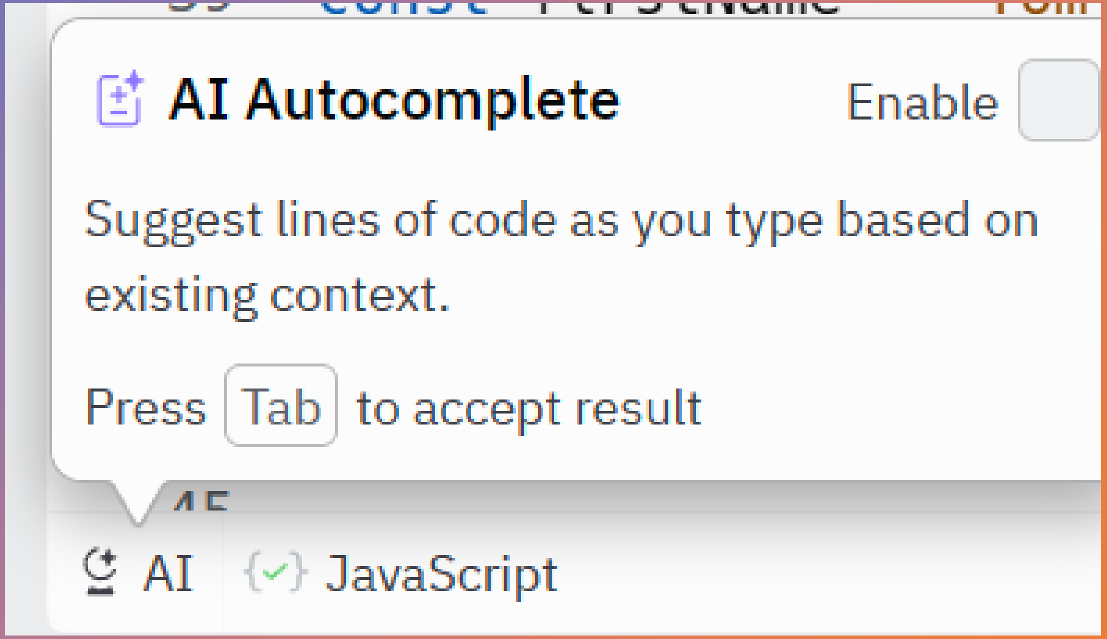
- Shell tab
  - Unix command line, not used for Haumea lessons
- Editing window
  - Central window where you write your code



- File browser



You will edit index.js for your lesson
README.md has lots of good information



AI Autocomplete Enable

Suggest lines of code as you type based on existing context.

Press Tab to accept result

Turn off the AI Assistant

- Please turn off the AI assistant
    - lower left corner of edit window
- Should be off by default
- Uncheck enable if it is on

Debugging

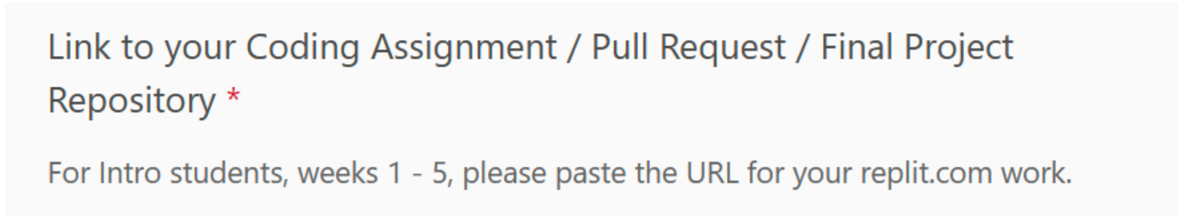
- Find out what the variables contain
- Test the functions and methods
- Console.log()
    - In replit, goes to the console tab/window
    - In a browser, goes to developer's tools console
- Developer's tools
    - Debugger, console
    - Lots of other goodies
        - Coming in the debugging lesson
    - Not needed for replit based lessons

Submitting your lesson

- Use this link: Submit Assignment
- Fill in the form
- Be sure to include your forked replit link

CODE THE DREAM SCHOOL

Assignment Submission Form


Link to your Coding Assignment / Pull Request / Final Project Repository *

For Intro students, weeks 1 - 5, please paste the URL for your replit.com work.

Demo and Q&A