JavaScript.

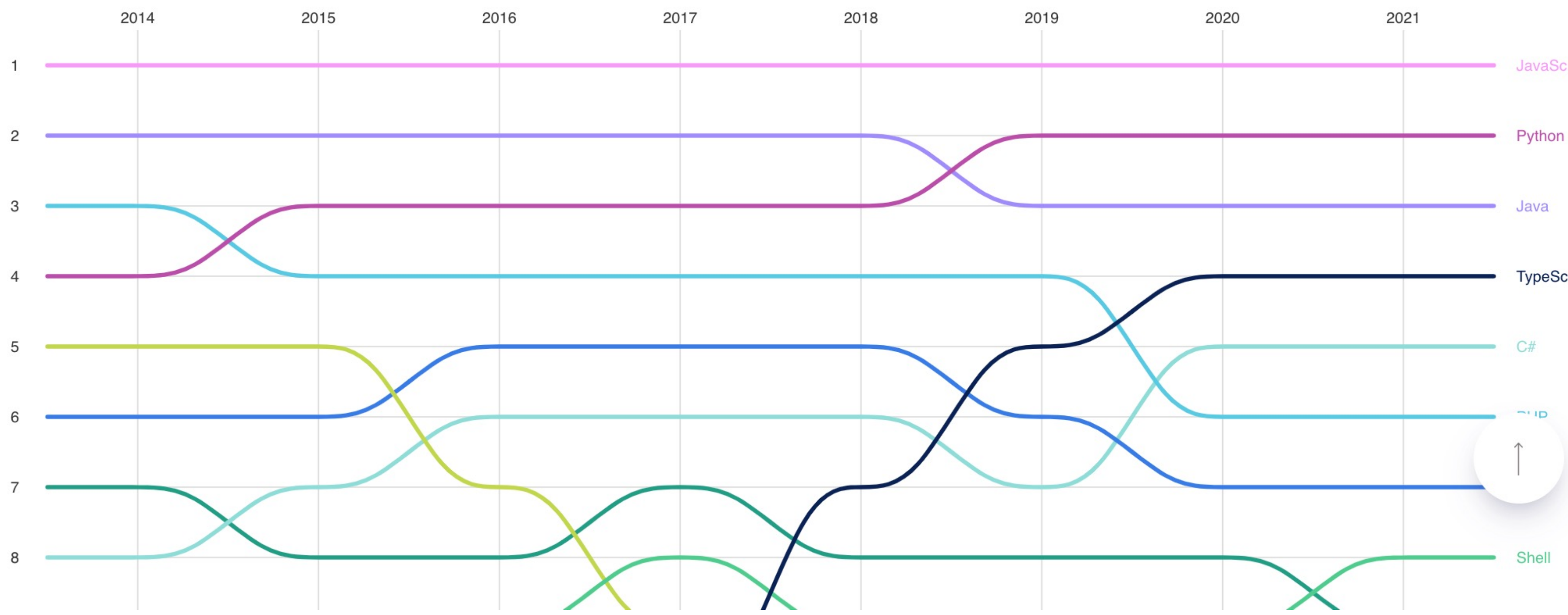
The Fundamentals

Topics

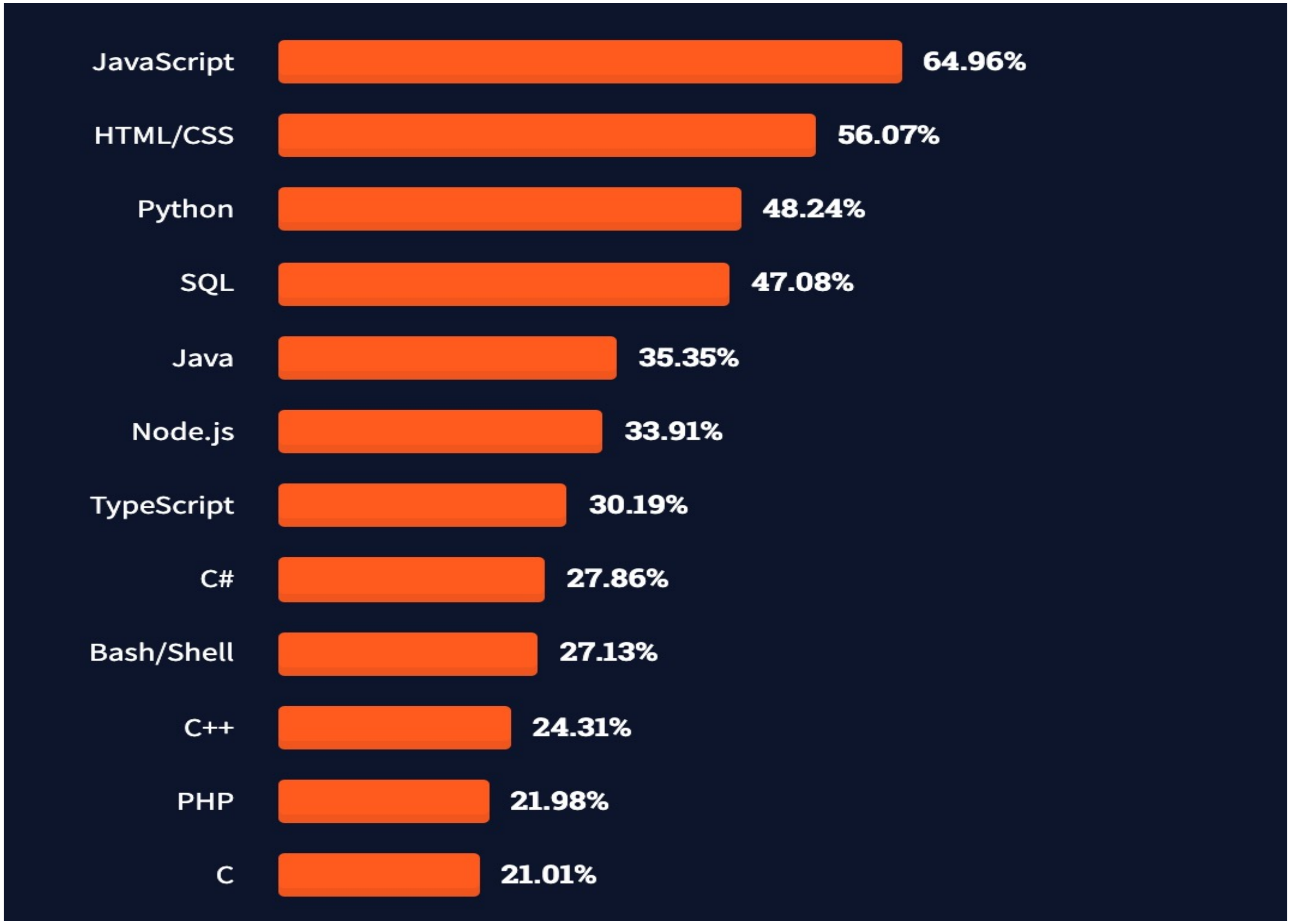
- **Background**
- **Data (State) representation**
  - **All about objects**
- **Behaviour (Logic) representation**
  - **All about functions**

Ref : https://octoverse.github.com/

Top languages over the years



Ref.: https://insights.stackoverflow.com/survey/2020#technology



Background.

- **Designed by Brendan Eich, at Netscape Corp. (early 1990s).**
  - **Influenced heavily by Java, Self and Scheme.**
- **Named JavaScript to capitalizing on Java's popularity.**
- **Netscape submitted JavaScript to ECMA for Standardization. (ECMA – European Computer Manufacturers Association. Organization that standardizes information)**
- **Resulted in new language standard, known as ECMAScript.**
  - **JavaScript is an implementation of ECMAScript standard.**
  - **ES1 - June 1997; ES2 - June 1998; ES3 - Dec. 1999; ES4 – Abandonned.**
  - **ES5 - 2009; ES6 - 2015 (ES2015); ES7 – 2016 (ES2016) ……**
- **The** node.js **platform (2009).**
  - **JavaScript on the server-side.**

Transpilation (using Babel)

- **Older Browsers cannot execute ES6+ JavaScript.**
  - **Must transpile code first.**

- **Newer browsers incrementally adopting ES6+.**
  - **Same for Node.js platform.**

- **The Babel tool suite.**
  - **One-stop shop for all transpilitation needs.**

The Semi-colon (;)

- Used to terminate a statement **statement.**
  - **Same as Java.**
- **Helps the runtime interpreter.**
- **Optional.**
- **Babel puts them back in - ASI.**
- **When omitted, be careful with multi-line expressions.**

JavaScript - Data representation.

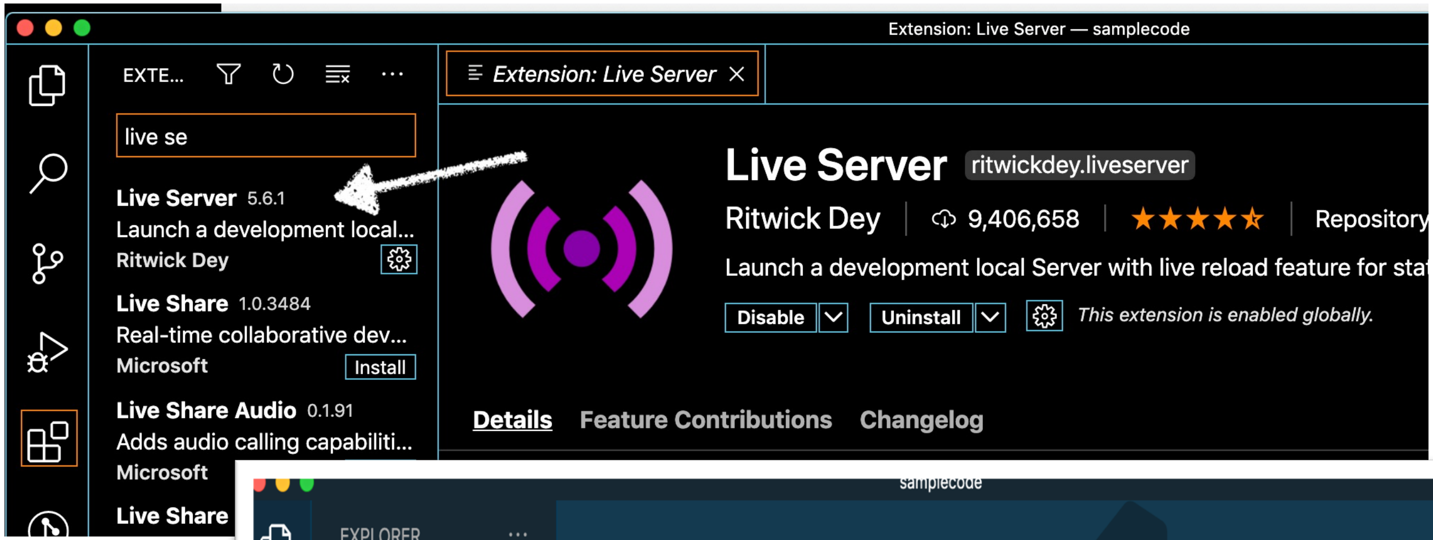
JavaScript Data Types.

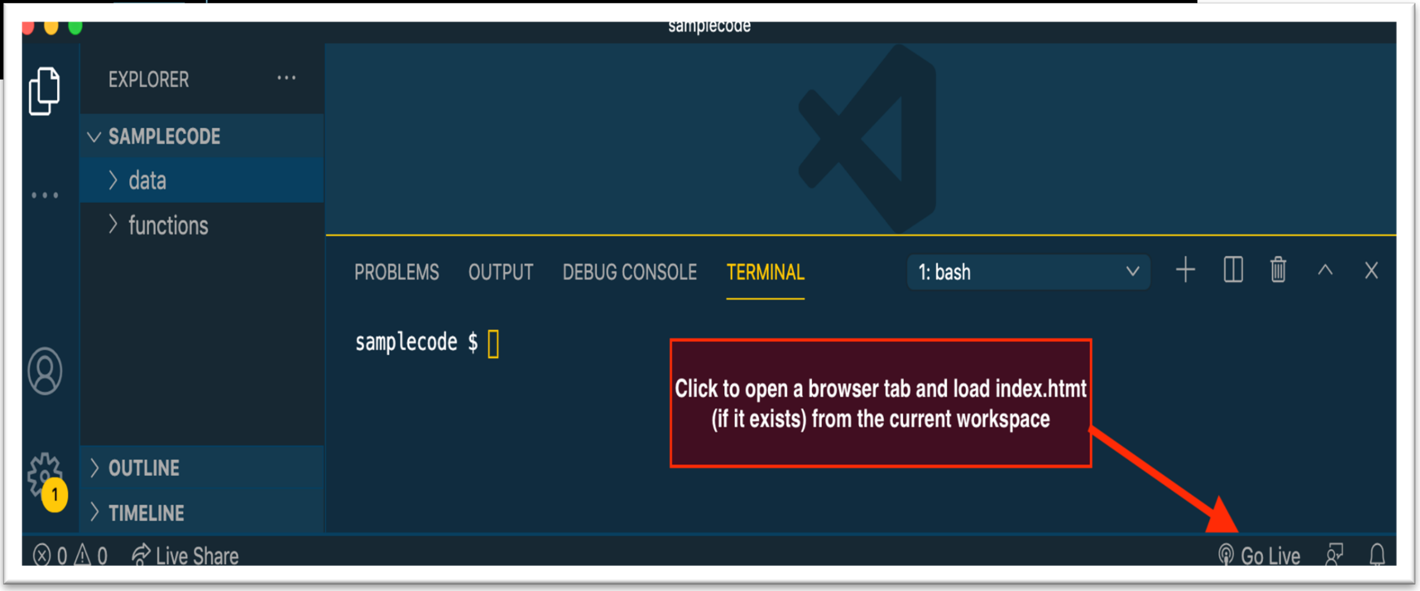
- **Data types:**
  1. **Primitives: number, string, boolean, null, undefined.**
     - *undefined* **– assigned to unitialized variable.**

  2. **Everything else is an <u>object</u>.**

- **JS is a <u>dynamically typed </u>language.**

Demo setup (VS Code)


Extension: Live Server — samplecode
EXTE...
live se
Live Server 5.6.1
Launch a development local...
Ritwick Dey
Live Share 1.0.3484
Real-time collaborative dev...
Microsoft
Install
Live Share Audio 0.1.91
Adds audio calling capabiliti...
Microsoft
Live Share
Extension: Live Server
Live Server ritwickdey.liveserver
Ritwick Dey
9,406,658
Repository
Launch a development local Server with live reload feature for sta
Disable
Uninstall
This extension is enabled globally.
Details
Feature Contributions
Changelog

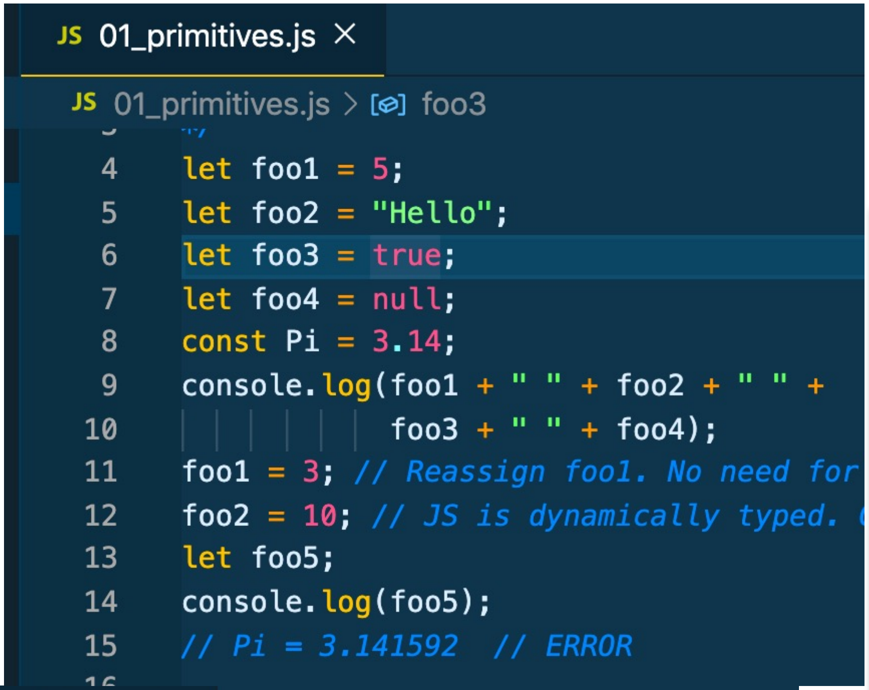


```
samplecode $ 
```

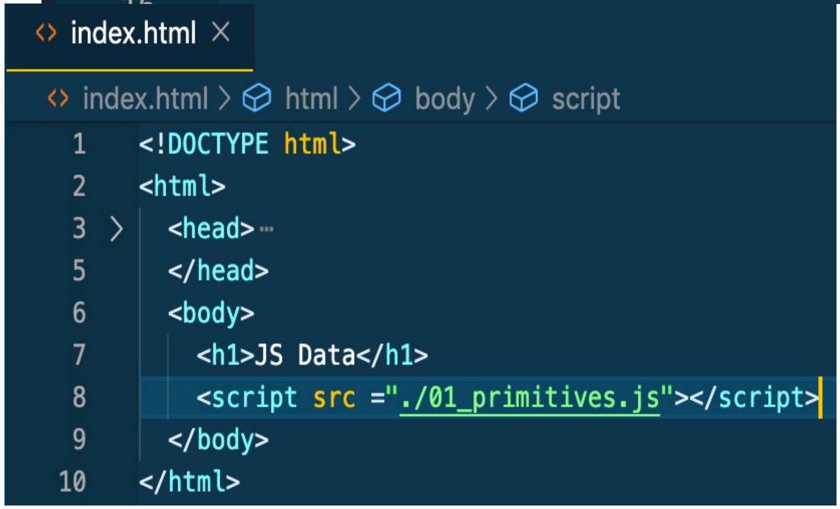
Click to open a browser tab and load index.htmt (if it exists) from the current workspace

- **Ref dataSamples/***01_primitives.js:*

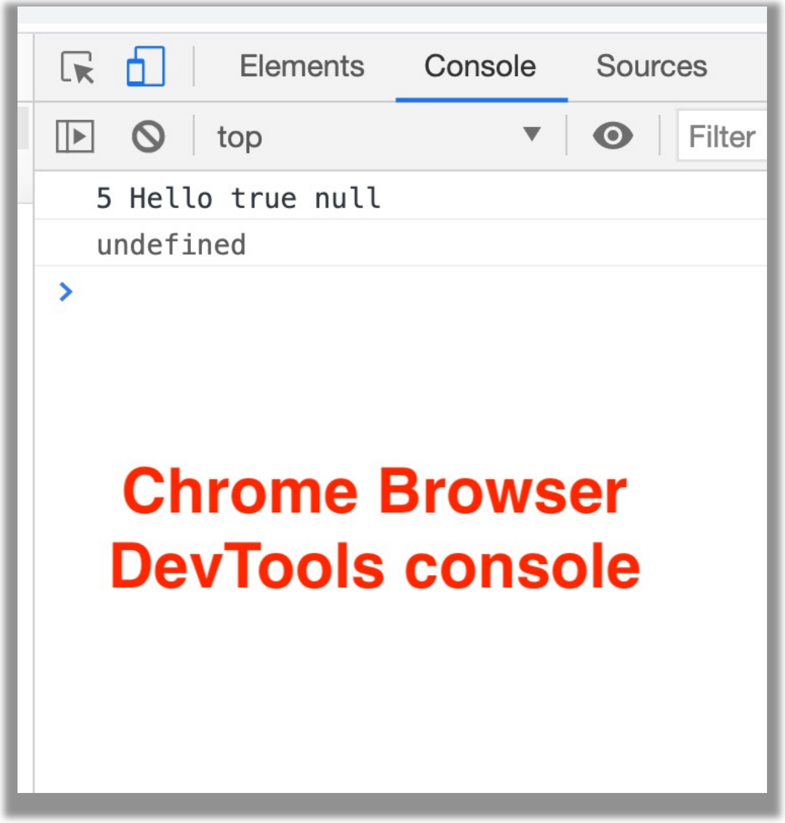
Primitive types.



```
let foo1 = 5;
let foo2 = "Hello";
let foo3 = true;
let foo4 = null;
const Pi = 3.14;
console.log(foo1 + " " + foo2 + " " +
            foo3 + " " + foo4);
foo1 = 3; // Reassign foo1. No need for
foo2 = 10; // JS is dynamically typed.
let foo5;
console.log(foo5);
// Pi = 3.141592  // ERROR
```



```
<!DOCTYPE html>
<html>
  <head>…
  </head>
  <body>
    <h1>JS Data</h1>
    <script src ="./01_primitives.js"></script>
  </body>
</html>
```



```
5 Hello true null
undefined
>
```

Chrome Browser DevTools console

Primitive types (Basic syntax).

```
let foo1 = 5 ;
```

- let **– keyword to indicate we are declaring 'something' (and assigning it a literal value in above case).**
  - **Use** const **when declaring constants (cannot reassign).**
- Identifier **– 'foo1' is an identifier for the thing being declared.**
  - **Lots of rules about valid format for identifiers (no spaces, don't start with numeric character, etc)**
- Operator **– e.g. +, =, * , –, [ ] (subscript) etc**
  - **Some rules about where they can appear in a statement.**

let & const

- let – **Declared variable CAN be reassigned**
- const – **Declared variable CANNOT be reassigned.**
  - **A Constant.**
  - **Use to clarify intent.**
  - **MUST be initialized on declaration.**

- **Both have block scope.**
  - **{ …. } encloses a block, e.g. for-loop, if, function, class**
  - **Same as Java**

Objects.

- **The fundamental structure for representing complex data.**
- **A unit of composition for data (or STATE).**
- **An object is a set of** key-value **pairs, termed** properties**.**

  { <key1> : <value1>, <key2> : <value2>, ... }

  - **Key (property name) - an identifier; must be unique within the object structure.**
  - **Value - can be a primitive value, object (nesting), array or function.**

e.g.

const me = { firstName: “Diarmuid”, lastName: “O’ Connor” } ;

Manipulating Object properties.

- **Two notations:**
  1. **Dot notation e.g** me.firstName ;
  2. **Subscript notation e.g.** me['firstName’] (Note quotes)
- **Same notations for** changing **a property value:**

  me.firstName = ‘Jeremiah’

  me[‘lastName’] = ‘O Conchubhair’
- **Objects declared with** *const* **ARE MUTABLE.**
  – *const* **cannot be reassigned, but its internal ‘value’ is mutable.**

- **Ref. dataSamples/02_objects.js**

Object characteristics.

- **Objects are dynamic.**
  - **Properties can be inserted and removed at run-time.**
  - **Ref. 03_dynamic_objects.js,**
- **Objects can be nested.**
  - **A property value may be an object structure.**
  - **Ref. 04_1_nested_objects.js**
- **A property value can be a variable reference.**
  - **Ref. 04_2_ nested_objects.js**

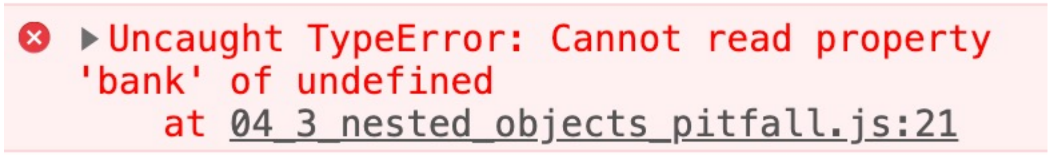
Object extras.

- **Object.keys(objRef) – get all keys in an object structure.**
- **Object.values(objRef) – get all values in an object structure.**
- **The 'in' operator – Does an object have a certain key? e.g.**
  'name' in me
- **Ref. 04_2_ nested_objects.js**
- **Internally JS stores object keys as** <u>strings</u>**.**
  - **Hence the subscript notation** – me['address'].

‘Cannot read property of undefined’ error

- **Access a invalid property of an object.**

  somObject.badProperty → undefined **(not fatal)**
- **Interpreting an invalid object property as an object.**
  - *someObject.badProperty.propertyX* **(Fatal)**



```
▶Uncaught TypeError: Cannot read property
'bank' of undefined
    at 04_3_nested_objects_pitfall.js:21
```

- **Ref. 04_3_ nested_objects_pitfall.js**
- **Other variations –**

  const var1 = null (or undefined)

  *var1.propert***yX** **ERROR**

Array data structure.

- **Dfn.: Array is an** ordered list **of values.**
  - **An object's properties are not ordered.**
- **Literal declaration syntax** :

  [<value1>, <value2>, ...]

- **Values can be of mixed type** (may reflect bad design!)**.**
- **Access elements with subscript notation.**
  - **Subscript termed an** index**.**
- **Ref 05_arrays.js**

Array data structure.

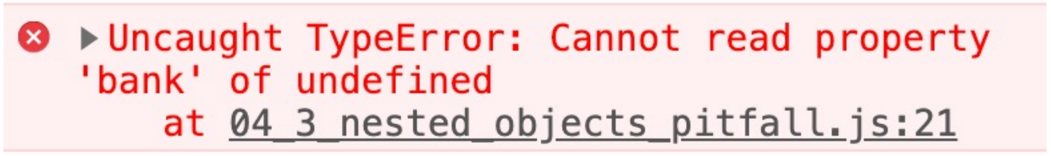
- **In JS, arrays are really just 'special' objects.**
  - **Index converted to a string for subscript notation:**

    nums[2] **becomes** nums['2']

- **An array 'object' has special properties built-in:**
  - **length property, e.g. const** len = nums.length
  - **Utility methods for manipulating elements e.g push, pop, shift, unshift, join, etc.**

Nested collections.

- **Arrays and objects can form nested data structures.**
- **Ex.:**
  - **An array where its elements are also arrays -** array_outer[3][2]
  - **An array of objects -** array_outer[1].propertyX.
  - **An object with a property whose value is an array -** objectY.propertyX[5]**].**
  - **etc.**
- **Accessing values inside a nested data structure can be a source of error.**



```
▶Uncaught TypeError: Cannot read property
'bank' of undefined
    at 04_3_nested_objects_pitfall.js:21
```

String templates (ES6)

- **String concatenation (ES5):**

```
console.log( foo1 + ' ' + foo2 + ' ' + foo3 + ' ' + foo4) ;
```

  - **Error prone and cumbersome.**

- **String template:**

```
console.log(` ${foo1} ${foo2} ${foo3} ${foo4} `) ;
```

  - **Use backquote (`) to enclose template, not single quote.**
  - **Interpolation: Embed variable / expressions using ${ …. }.**
  - **Expression is evaluated and result inserted into string**
- **Multi-line strings.**

- **Ref 06_string_templates.js**

Summary

- **Representing Data / State.**
  - **Primitives.**
  - **Objects.**
    - **Set of properties – key-value pairs.**
    - **Dynamic, nested.**
  - **Arrays.**
  - **String templates**

- **That runtime error**: *‘Cannot read property … of undefined*’