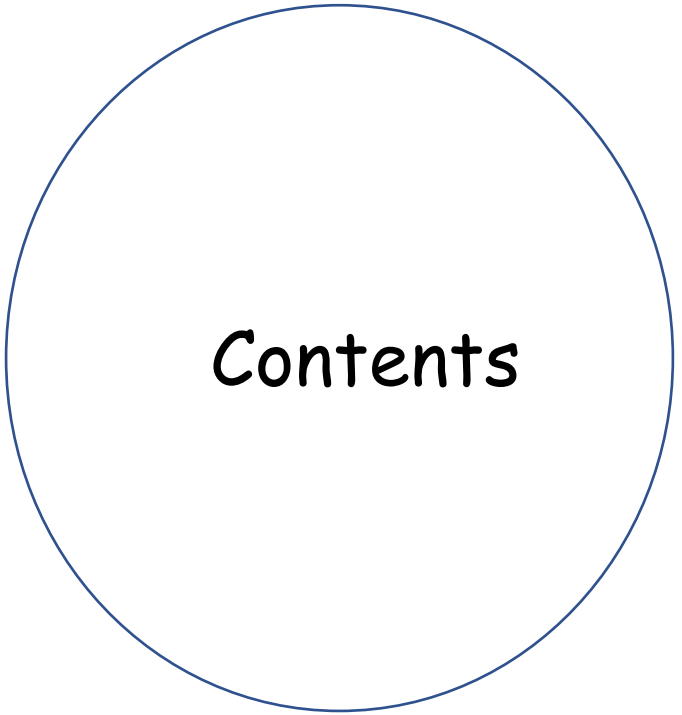
JavaScript

Partha Das Chowdhury

- Basics
  - How to create a js program?
  - Hello World
- Execution Context
- Hoisting
- Scope in JS
- Call Stack
- Syntax
- Arrays
  - For loop
- Anonymous Functions
- Arrow Functions
- Events



JavaScript is not Java

- A scripting language. We write it and run it.
- We saw it since mid 90s.
  - Brandon Eich of Netscape/Mozilla
- JavaScript, HTML & CSS is what we experience on the Internet.
- JavaScript runs in the client side
  - Say we want to check if the user enters passwords match during registration.

The Basics

- Create a file with .js
- Do not need code delimiters
- We include them in the script tag

```
1. <script src="your JavaScript.js"></script>
2. <script> Write the code here </script>
```

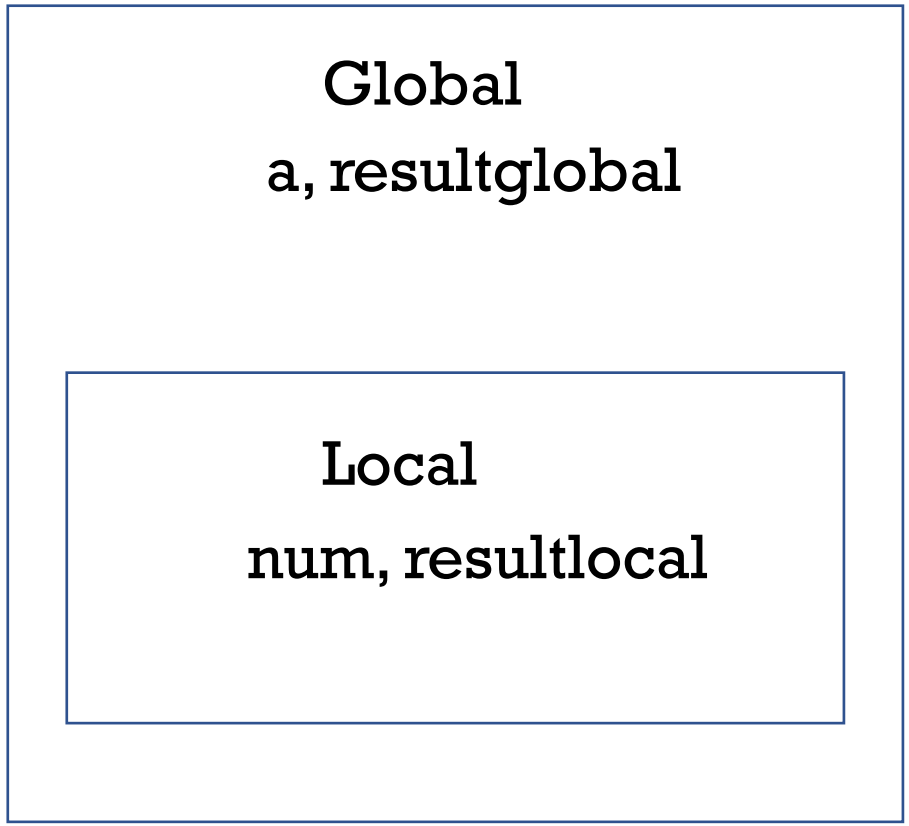
- We put the script in the same file or a different file.
- Put them in a separate file and link it using the script tag.
- Doing so looks good and is efficient.

```
<html>

<head> <title> Hello World</title> </head>

<body>



    <script>
            function doAnything(){
                alert('Hello World');
              }

          doAnything();
      </script>

</body>
</html>
```

Execution Context

- The browser sees our js code….
- Creates a box to run our code --- Execution context
  - That has our variables
  - That has our code
- Global & Local execution context
  - Functions are local and reside within the global context

Execution Context




```
var a = 2;

function add (num) {

  var resultlocal = num + 10;

}

var resultglobal = add(a);
```

Hoisting

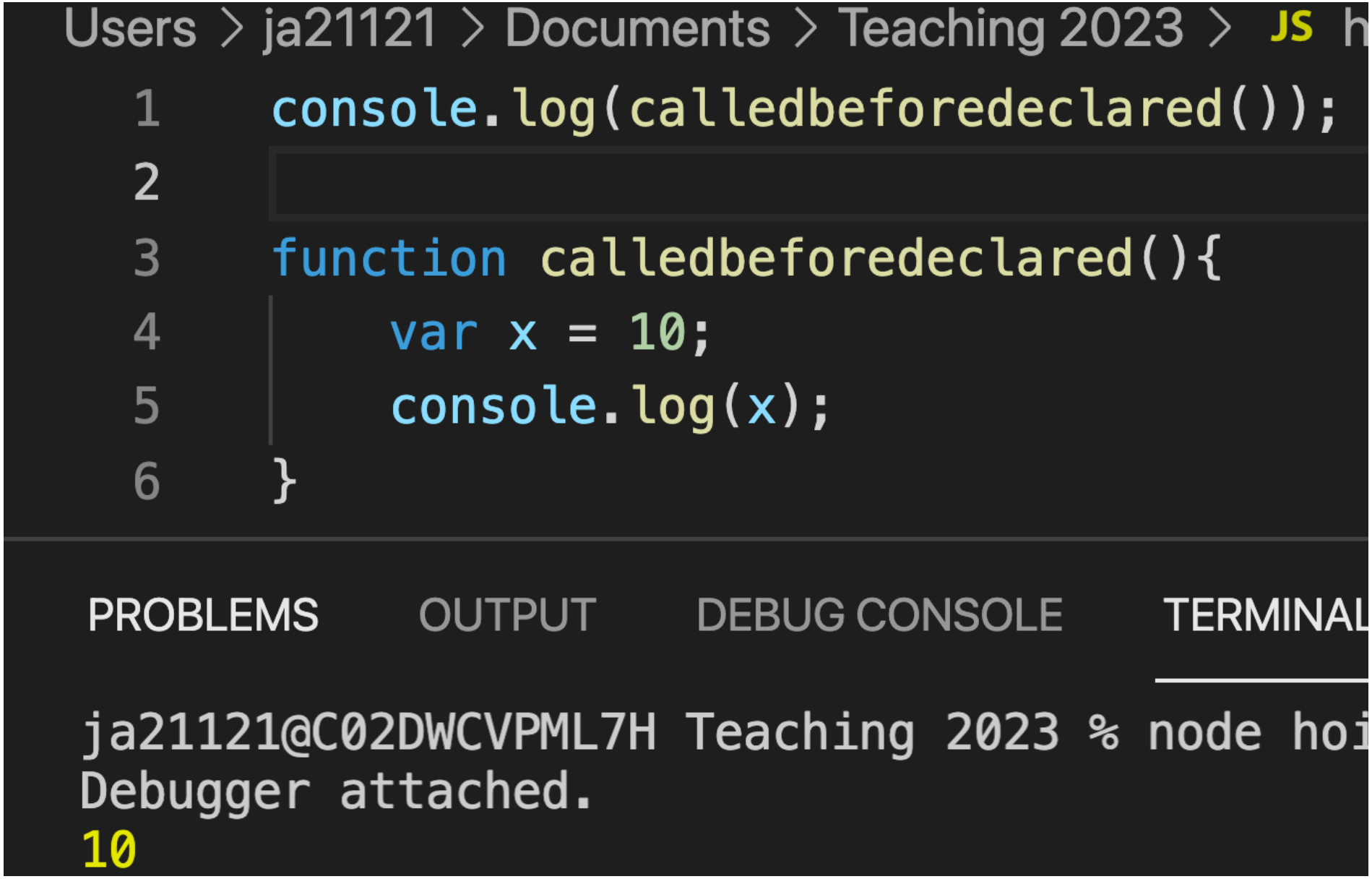
```
1    console.log(x);
2
3    var x = 10;
```

```
PROBLEMS    OUTPUT    DE

ja21121@C02DWCVPML7H Te
Debugger attached.
undefined
```

```
1    console.log(x);
2
3    //var x = 10;
```

```
PROBLEMS    OUTPUT    DEBUG CONSOL

    at Function.executeUserEntryP
ja21121@C02DWCVPML7H Teaching 202
Debugger attached.
Waiting for the debugger to disco
/Users/ja21121/Documents/Teaching
console.log(x);
            ^


ReferenceError: x is not defined
```

Hoisting


Users > ja21121 > Documents > Teaching 2023 > JS h
1 console.log(calledbeforedeclared());
2
3 function calledbeforedeclared(){
4 var x = 10;
5 console.log(x);
6 }
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
ja21121@C02DWCVPML7H Teaching 2023 % node hoi
Debugger attached.
10

Hoisting

- Left it says undefined and right it says not defined.
- Let's remember execution context
- Context has all the variables & functions.
- That's is called hoisting.
- In case of var it is declaration hoisting.
- In case of functions its value hoisting.

Scope

Users > ja21121 > Documents > Teach

```
1  function first(){
2      console.log(a);
3      function second(){
4
5      }
6  }
7  var a = 10;
8  first();
```

PROBLEMS OUTPUT DEBUG CONS

```
ja21121@C02DWCVPML7H Teaching 2
10
```

Users > ja21121 > Documents > Teaching

```
1  function first(){
2      console.log(a);
3      function second(){
4          var a = 10;
5      }
6  }
7
8  first();
```

PROBLEMS OUTPUT DEBUG CONSOL

```
    console.log(a);
                ^

ReferenceError: a is not defined
```

Scope

- Scope in JavaScript is lexical
- Means where a variable is created and its parents'
- But not the child.

- In our example
- First() has access to the global execution context
- First does not have access to its child which is Second()
- So, we have the error on the right picture.

As one moves up to parents and grand parents that defines the scope chain.

Call Stack

- Push and pop as contexts are created
- Known by various names

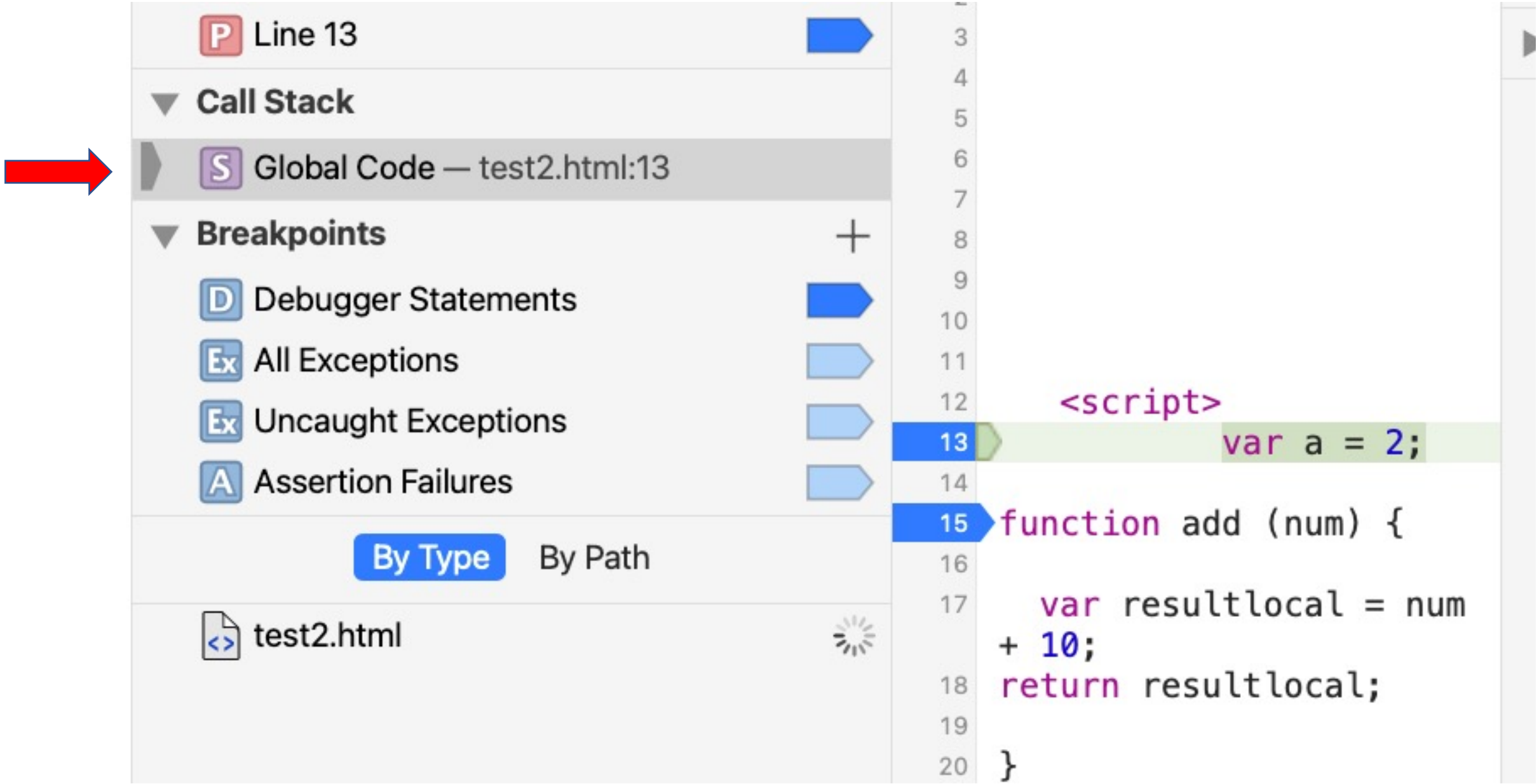
Global

a, resultglobal

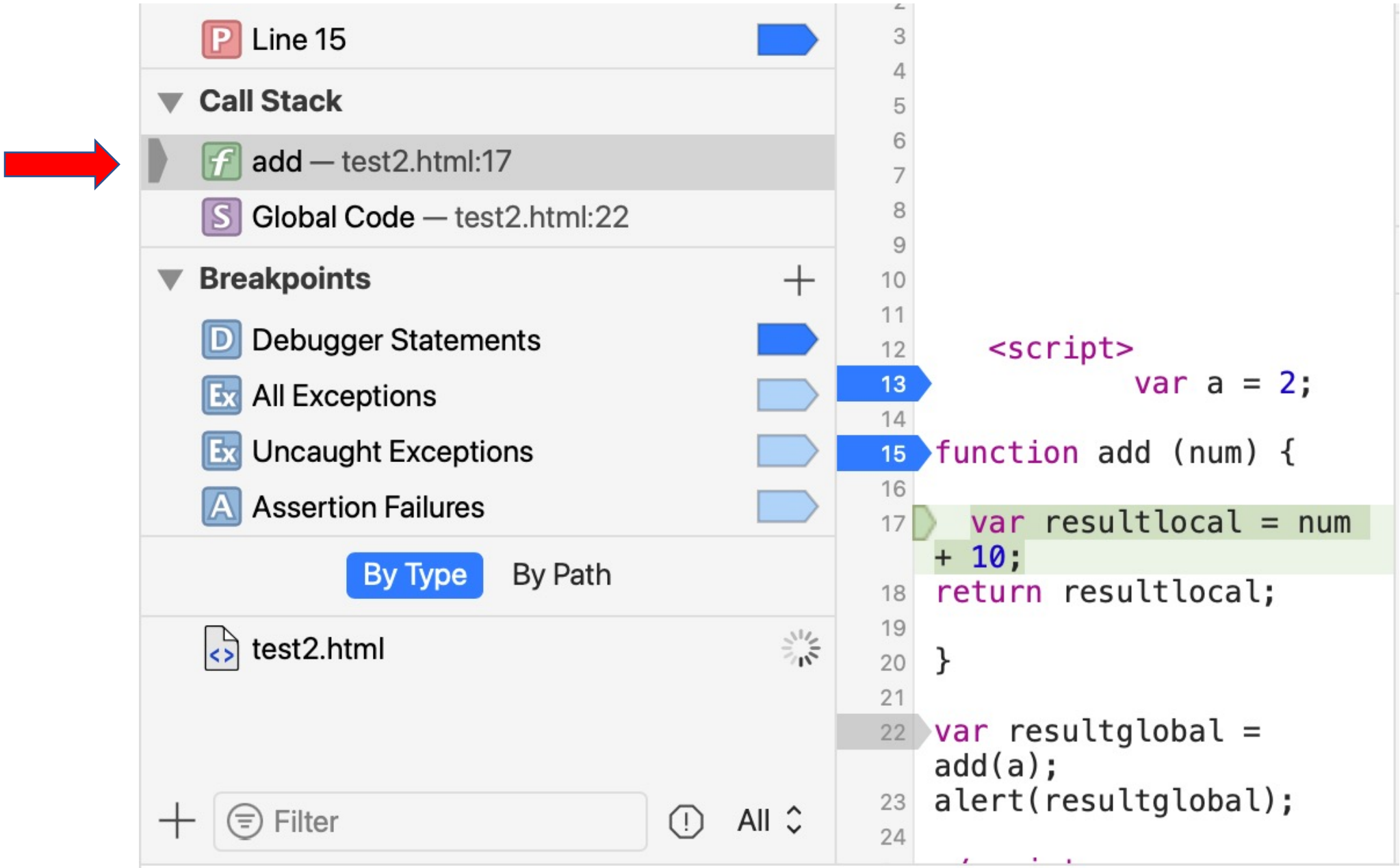
Local

num, resultlocal

Call Stack


Line 13
Call Stack
Global Code — test2.html:13
Breakpoints
Debugger Statements
All Exceptions
Uncaught Exceptions
Assertion Failures
By Type
By Path
test2.html
<script>
var a = 2;
function add (num) {
var resultlocal = num + 10;
return resultlocal;
}

Call Stack


Line 15
Call Stack
add — test2.html:17
Global Code — test2.html:22
Breakpoints
Debugger Statements
All Exceptions
Uncaught Exceptions
Assertion Failures
By Type
By Path
test2.html
Filter
All
<script>
var a = 2;
function add (num) {
var resultlocal = num + 10;
return resultlocal;
}
var resultglobal = add(a);
alert(resultglobal);

Syntax

We have all of these, we are perhaps familiar with

- break
- If…else
- for….
- function
- do….while
- var, let & const
- return
- switch
- throw
- try…catch
- ……
- arrays (You can have mixed arrays)
  - var example = [1, joe, 5.555]

Syntax

And these too….

- Comparison operators : <,<=,>,>=, ==, ===
- Logical Operators: &&, ||, !
- ("3.0"=== 3) ("3.0"== 3)

We don't need to declare types

- Language makes the best guess
- It mostly gets it right, when it doesn't use parseInt for Integers.
- Many operators automatically convert types so happens in many operations.
- var name = prompt("What's your name?");
- var age = 20;

Arrays

```
var name = ``johndoe";

console.log(name.length);   = 7


console.log(name[0]); = 'j'
console.log(name[name.length]); = undefined
console.log(name[name.length-1]); = 'e'
```

Arrays & for loop

```
var example = [1, joe, 5.555]
```

A mixed array

```
for var element in
example {
    console.log(element);
    }
```

index

```
for var element of
example {
    console.log(element);
    }
```

value

- Please see the functions that you can apply to arrays like map(),pop() etc.

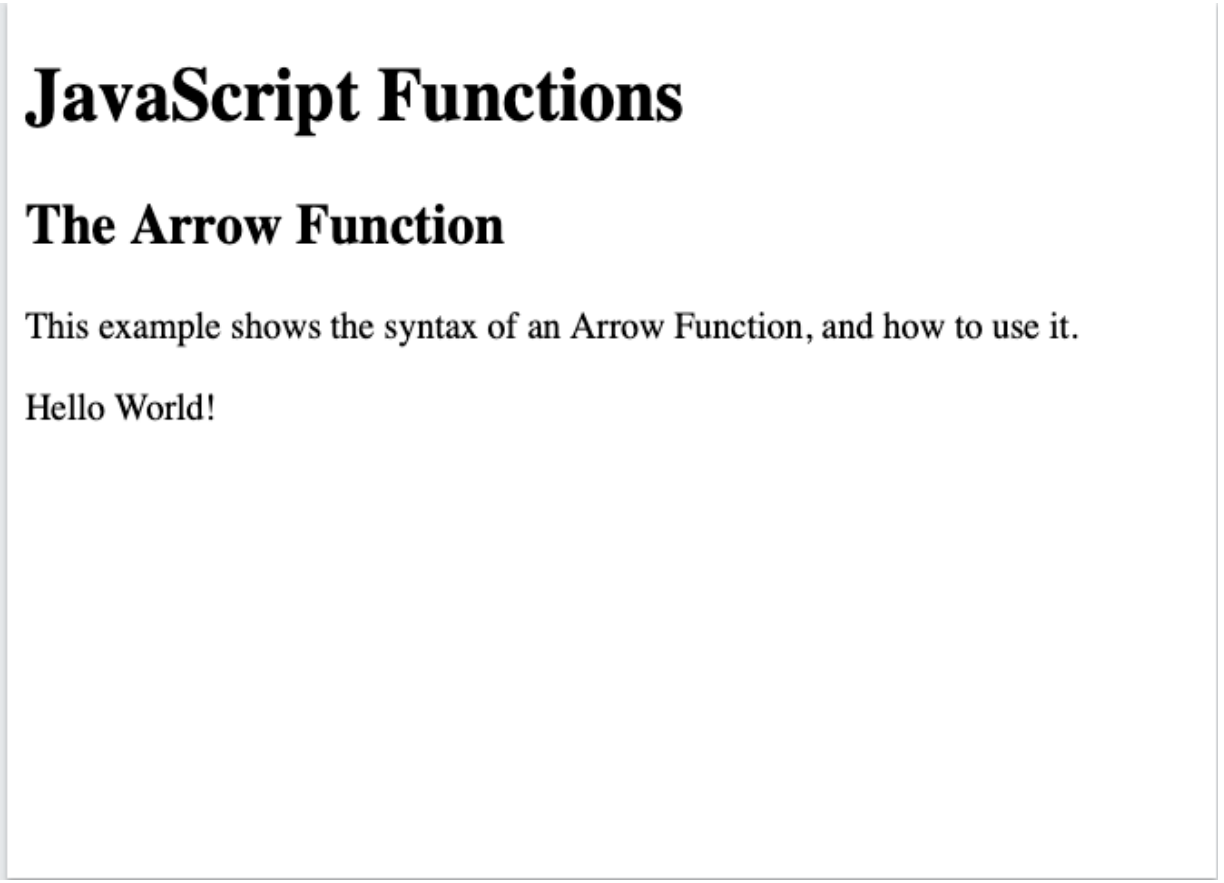
Anonymous Functions

- All functions do not need to have a name. Let's use map() and an array.
- map() allows you to apply a function to all the elements of an array.
  - var age = [20,25,30,35]
  - Let's use map to add 10 years to each of the values in the array.
  - age = age.map(function(any)
    {
        return age + 10;
    })

Arrow Functions

- https://www.w3schools.com/js
- Write shorter functions

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Functions</h1>
<h2>The Arrow Function</h2>

<p>This example shows the syntax of an Arrow Function,
and how to use it.</p>

<p id="demo"></p>

<script>
let hello = "";

hello = () => {
  return "Hello World!";
}

document.getElementById("demo").innerHTML = hello();
</script>
```



Events in JavaScript

- Anything we do on a web page is an event
  - Click on buttons, enter a text, hover our mouse
  - HTML pages can capture the event which can be passed to an event handler in JavaScript.

```
<html>

<head> <title> Event Handlers </title> </head>

<body>

        <button onclick="alertName(event)">Button 1 </button>
        <button onclick="alertName(event)">Button 2 </button>


  <script>
        function alertName(event)
          {
            var trigger = event.srcElement;
            alert('You clicked on ' + trigger.innerHTML);
           }

   </script>

</body>

</html>
```

Thank You