Understanding Function Scope and Closure

Dan Geabunea

Passionate Software Developer

@romaniancoder | www.dangeabunea.com

Overview

Defining scope

Walking through various JavaScript scopes

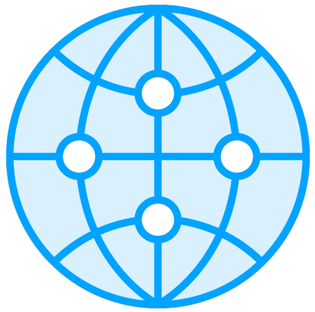
Understanding closures

Scope

The place (execution context) where variables and expressions are visible (can be referenced)

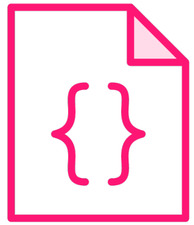
MDN Docs: https://tinyurl.com/3u72jtev

Scopes in JavaScript



Global

Function / Local



Block

```
let appName = ‘Cool app’;
```

Global Scope

A variable defined outside a function is a global variable
All functions and scripts on that web page can access it

```
{
  const aircraftModel = 'A380';
}
console.log(aircraftModel); // aircraftModel is not defined


for (let i = 0; i < 10; i++){
  console.log(i);
}
console.log(i); // i is not defined
```

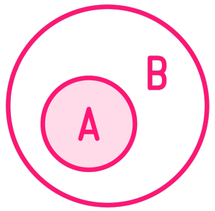
Block Scope

Introduced in ES6
Applies to variables declared using “let” and “const”
Variables defined within a block, denoted with { } are only accessible in that block

Function Scope

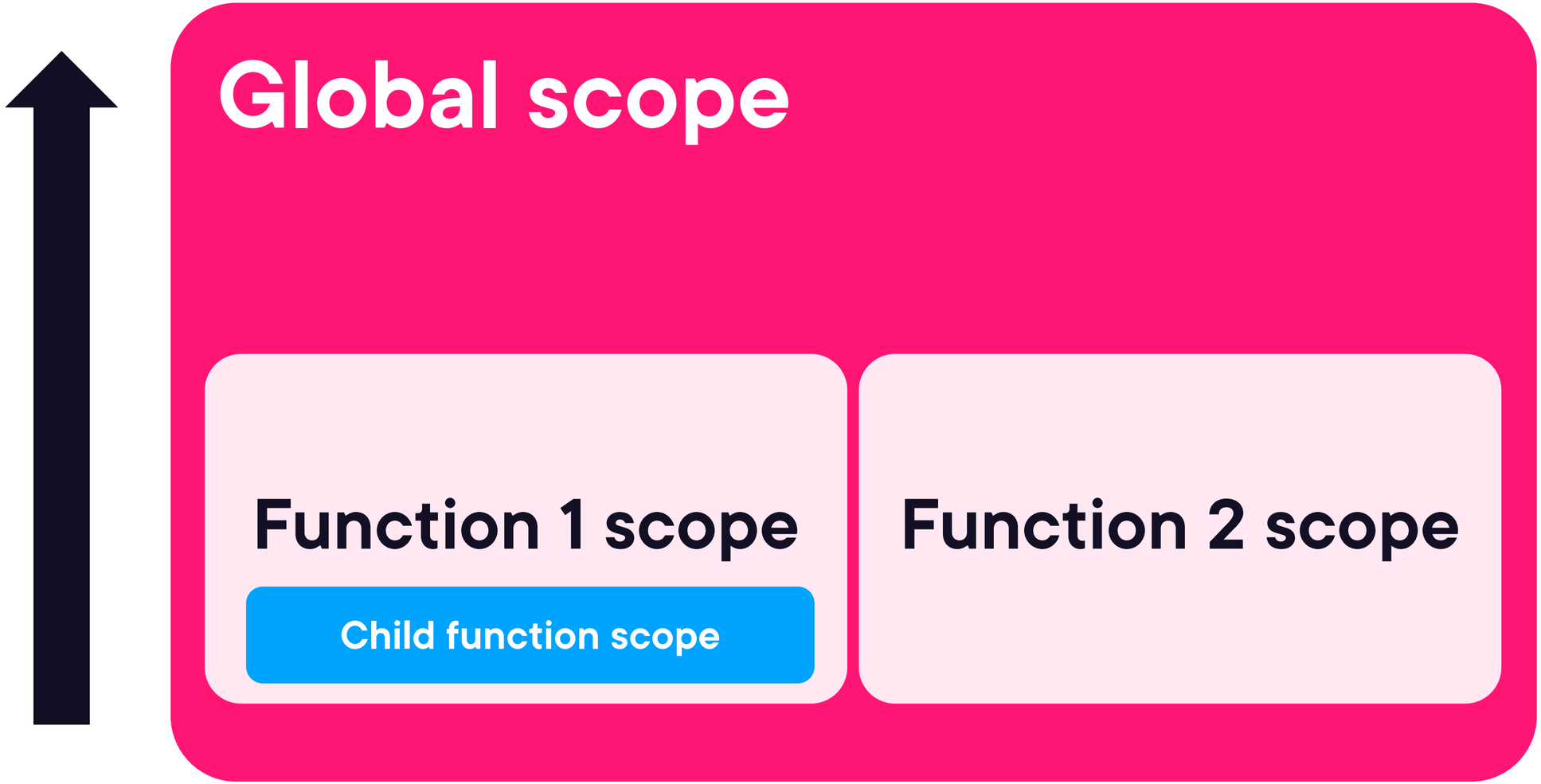
Each function has its own scope

Variables declared inside a function are not visible outside the function



A function can still access variables defined in its outer scope

Scope Layers



Function Scope

```
const airportAltitude = 100;

function land() {
  let currentAltitude = 2000;

  let altDiff = currentAltitude - airportAltitude;

  // logic to land
}

console.log(currentAltitude); // Not defined
```

Variables in Function Scope

```
function land() {
  var minSpeed = 350;
  let descentRate = 30;
  const runwayAltitude = 0;
}
```

Understanding Function Scope

Closure

Closure

The combination of a function and the lexical environment within which that function was declared

MDN Docs: https://tinyurl.com/mr3zutbr

Closure

Imagine you have an inner function defined in an outer function.

A closure is a JavaScript feature in which the inner function has access to the variables of the outer function...

...even after the outer function has returned.

MDN Docs: https://tinyurl.com/mr3zutbr

Closure Explained

```
function parent() {
  // variable in outer function
  let name = ‘John Doe';

  function child() {
    console.log(name);
  }

  return child;
}

let child = parent(); // outer function exited

child();
```

```
John Doe
```

```
function parent() {
  let name = ‘John Doe’;


  function child() {
    console.log(name);
  }
  return child;
}
```

Closure

A closure is the combination of a function and the lexical environment within which that function was declared. – MDN Docs

This environment consists of any local variables that were in-scope at the time the closure was created. – MDN Docs

Implementing Function Closure

Summary

Scope represents the execution context where variables are visible

Global scope

Function scope

Block scope

Closure is the combination of a function bundled together with reference to its surrounding variables and statements

Up Next:

Understanding “this”