Functions

Lesson Objective

- Understand idea and uses of functions
- Learn how to use inbuilt functions (methods)
- Learn how to write functions in JavaScript
- Understand function call and return in relation to stack frames
- Understand scope and scope chain

Function

- Modular organization of related set of codes, to perform a specific task.
- A function can be a program by itself, but usually a program is composed of number of functions. i.e. in most cases, a function is a "subprogram"

Function declaration (function statement)

```
function [functionName]([param1[, param2[, …paramN]]]){
      statements;
      [return value;]
}
```

- The first line of function is called the header or signature, and it includes the keyword `function`, the function name and the optional parameter list.
- The statements inside a function are called the body of a function.
  - Function returns undefined, when return is not explicit.
- See examples:
  - lecture_codes/lesson5/func_say_hi.js
  - Lecutre_codes/lesson5/func_test_odd.js

Calling a function

- A function by itself won't do anything, unless you call/invoke it.
- How a function is called depends on functions header/signature
  - To call a function, you simply write a function name followed by a set of parentheses; optionally passing matching arguments for the corresponding parameters.

```
functionName([arg1[, arg2[, …argN]]])
```

Pencil Exercises

- Lesson 9 - examples 2 and 4
- Lesson 9 – examples 6 and 8

Function call and stack frame

```javascript
// Output?

function A(){
    console.log("A is called");
    console.log("Before B is called");
    B();
    console.log("After B is called")
}

function B(){
    console.log("B is called");
    console.log("Before C is called");
    C();
    console.log("After C is called");
}

function C(){
    console.log("C is called");
}
A();
console.log("After A is called");
```

Example: Lets draw a stack

```
function funA(a,n){
    let something;
    something = "something.";
    funB(something, n);
}

function funB(a,b){
    let thing;
    thing = "a thing.";
    console.log("What is on the stack when we're here?");
}

function main(){
    let test;
    let n;
    test = "Hello";
    n = 5;
    funA(n, 10);
}

main();
```

Exercise: Draw the stack

```
function funX(a, b) {
    let c;
    c = 5;
    funY(a * c, "yes");
}

function funY(x, y) {
    let z;
    z = "I can see the sea";
    console.log("What is on the stack here?");
}

function main() {
    let a;
    let b;
    a = "Hello";
    funX(3, a);
    b = "World";
}

main();
```

Function expression & Anonymous Function

- A function keyword can be used to define a function inside an expression

```
// function expression
let sayHi = function(){console.log("Hi");}
sayHi();
```

- Function without a name is called anonymous function.

- How would you write above function expression as a statement/declaration?

Arrow function

- New syntax introduced in ES6 to write a function in concise way

```
let isEven = (a) => {return a%2===0;}
console.log(isEven(4));

let isOdd = (a) => a%2 !== 0;
console.log(isOdd(7));

let sayHello = () => console.log('HI');
sayHello();
```

```
(arguments) => { return statement } // general syntax
  argument => { return statement } // one parameter
      argument => statement // implicit return
            () => statement // no input
```

Main point

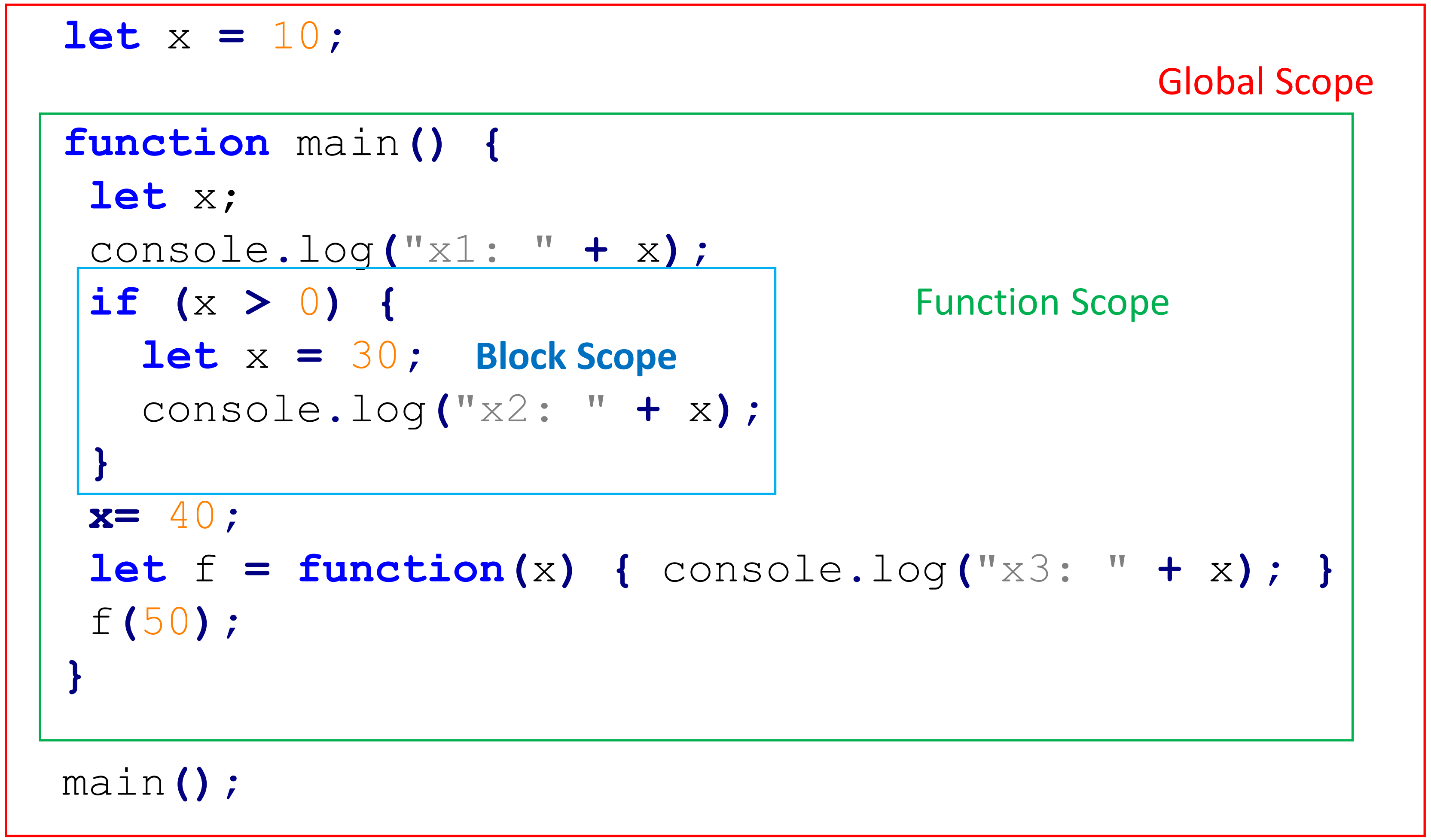
- Functions are subprograms and a computer program usually is composed of number of smaller functions. Functions makes programming modular, reusable and easier to understand. When a program starts to get complex, we must break it into smaller functions in order to handle it better. To be a better programmer we should not only be able to solve a problem at hand, but also need to be able to break it into smaller, meaningful, reusable functions. *Science of consciousness, With the regular experience of pure consciousness through practice of TM, one develop ability to fine focus on smaller details without missing the big picture*.

Scope revisited

- The scope of a variable determines how long and where a variable can be used.
- There are two level of scopes in JavaScript, i.e. local inside of a function and global outside of a function.
  - Parameters are local to a function.
- With the let keyword from ES6, JavaScript also have block scope.
- See example: *lecture_codes/lesson5/scopes.js*

Lexical scope in JavaScript (ES6+)

- From ES6, in JavaScript every block ({}) defines a scope.



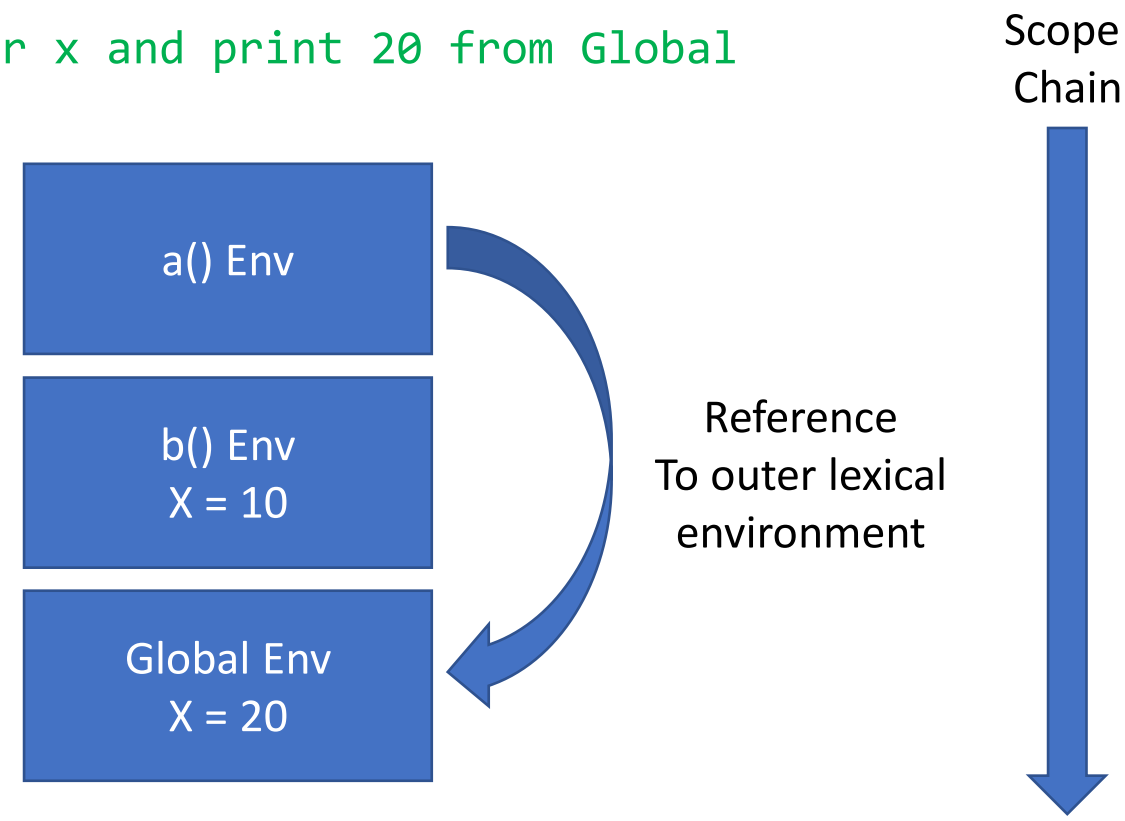
```
let x = 10;

function main() {
 let x;
 console.log("x1: " + x);
 if (x > 0) {
   let x = 30;
   console.log("x2: " + x);
 }
 x= 40;
 let f = function(x) { console.log("x3: " + x); }
 f(50);
}

main();
```

Scope chain

- When we refer to a variable in a program, JS engine will look for that variable in the current scope. If it doesn’t find it, it will consult its outer scope until it reach the global scope.

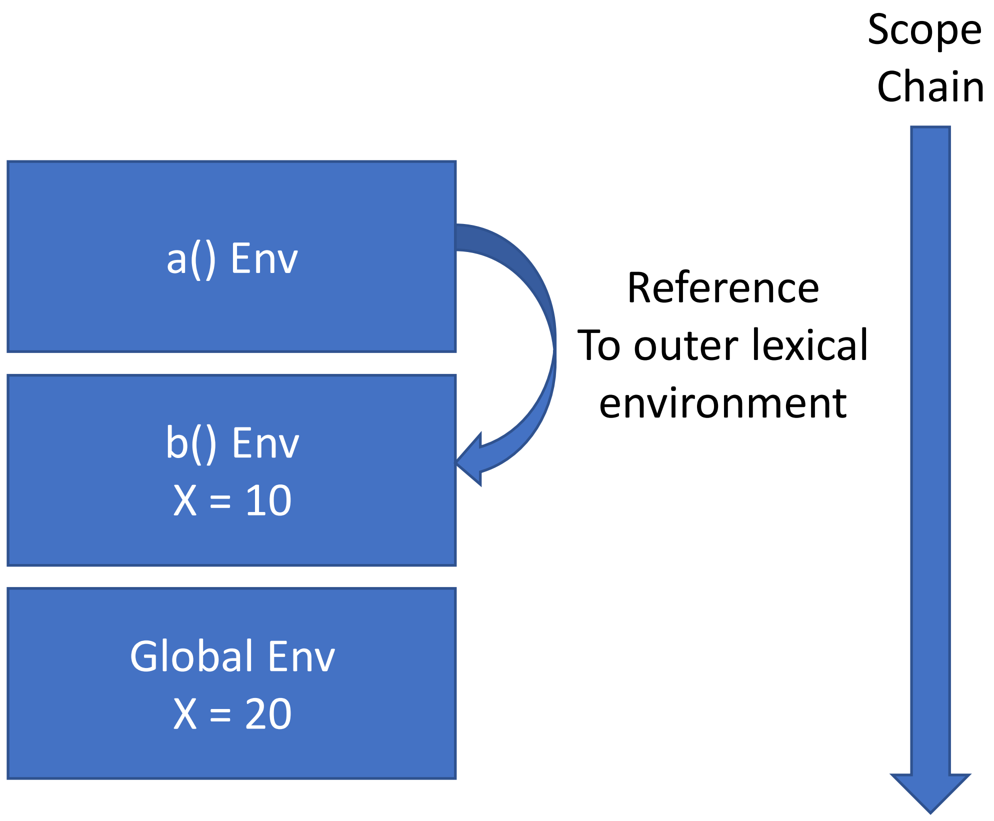
Scope Example

```
function a(){
        console.log(x); // consult Global for x and print 20 from Global
}

function b(){
        let x = 10;
        a(); // consult Global for a
        console.log(x);
}

let x = 20;
b();
```



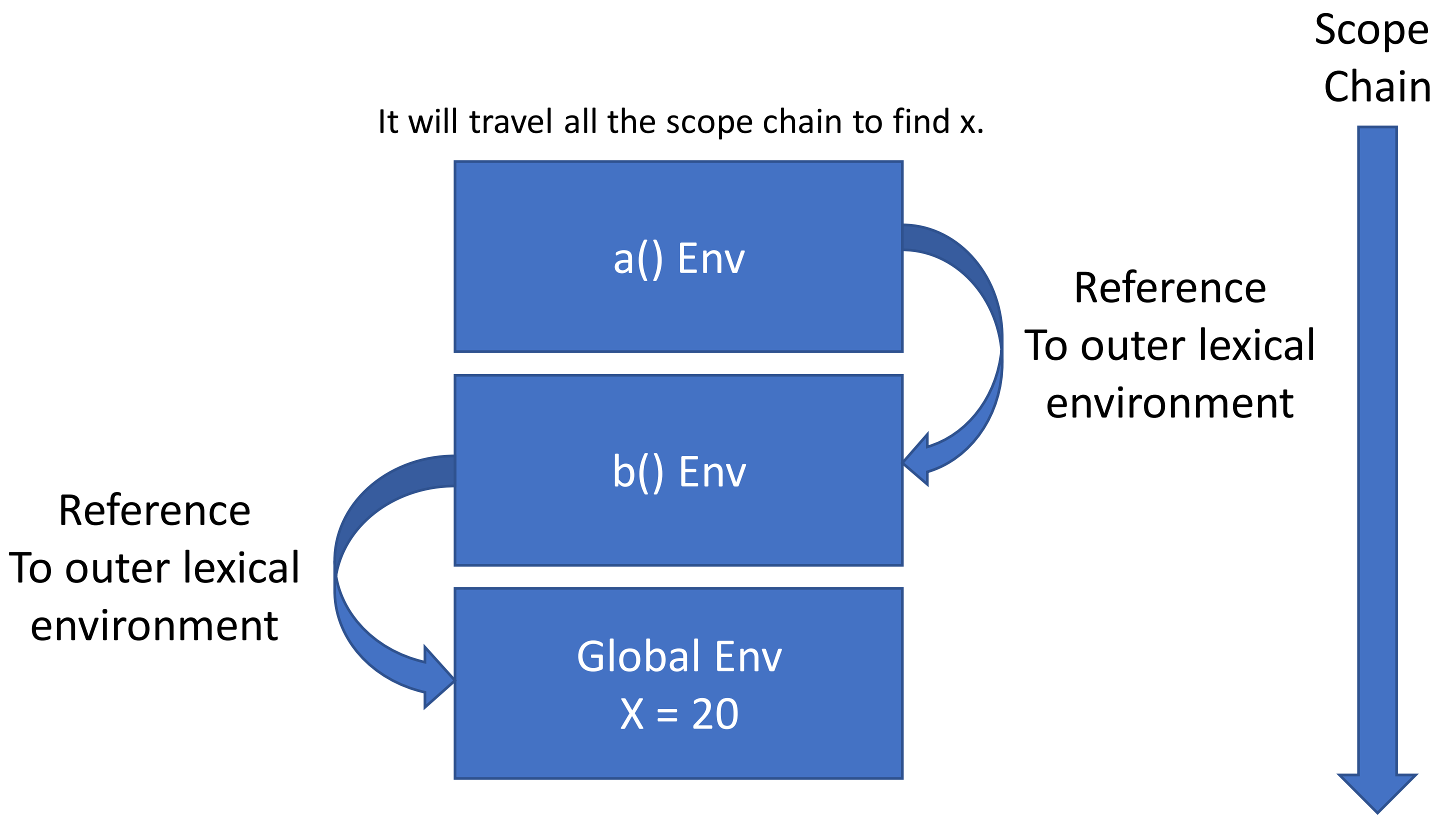
Scope Example

```
function b(){
        function a(){
                console.log(x);
        }
        let x = 10;
        a();
        console.log(x);
}

let x = 20;
b(); // 10
```



Scope Example

```
function b(){
        function a(){
                console.log(x);
        }
        a();
        console.log(x);
}

let x = 20;
b(); // 20
```


Scope Chain
It will travel all the scope chain to find x.
a() Env
Reference To outer lexical environment
b() Env
Reference To outer lexical environment
Global Env X = 20

Scope Example

```
function f() {
        let a = 1, b = 20, c;
        console.log(a + " " + b + " " + c); // 1 20 undefined

        function g() {
           let b = 300, c = 4000;
           console.log(a + " " + b + " " + c); // 1 300 4000
           a = a + b + c;
           console.log(a + " " + b + " " + c); // 4301 300 4000
        }

        console.log(a + " " + b + " " + c); // 1 20 undefined
        g();
        console.log(a + " " + b + " " + c); // 4301 20 undefined
}
f();
```

Exercise

```
let x = 10;
function main() {
    let x = 0;
    console.log("x1 is " + x);
    x = 20;
    console.log("x2 is " + x);

    if (x > 0) {
        x = 30;
        console.log("x3 is " + x);
    }

    console.log("x4 is " + x);

    function f(x) {
        console.log("x5 is " + x);
    }

    f(50);
    console.log("x6 is " + x);
}
main();
console.log("x7 is " + x);
```

Main Point
Scope chain and execution context

- When we refer a variable in a program, JS engine will look for that variable in the current scope. If it doesn't find it, it will consult its outer scopes until it reach the global scope. *Science of consciousness, During the process of transcending we naturally proceed from local awareness to more subtle levels of awareness to the unbounded awareness*.

Exercise (make examples running)

- Lesson 9 - example 13, "Area of a Triangle".
- Lesson9 - example 15, "How long to Invest"

Assignments

- Reading chapter 9
- **See next slides for the programming assignments**
- We will revisit programming assignment for chapter 9 after we cover HTML, DOM and events.

Programming assignments

1. Write a function named `checkPrime` that accepts a parameter and returns true if the argument is a prime number otherwise returns false.
   - Now write a program that prompts user to input a number and calls the function `checkPrime` to see if the entered number is a prime number or not.
2. Write a function `farhToCels` that accepts a parameter for temperature in degree Fahrenheit and returns temperature in degree Celsius.
   - Now, write a program that prompts user to enter a temperature value in degree Fahrenheit and returns result in degree Celsius by using the function `farhToCels`

Programming assignment

- Lesson 9 - Programming assignment 6
  - ~~function doInputOutput()~~
  - function `houseVolume(width, depth, height, sweep)`
  - function `livingVolume(width, depth, height)`
  - function `roofVolume(width, depth, sweep)`
  - function `triangleArea(a, b, c)`

  - Note: `houseVolume` depends on `livingVolume` and `roofVolume`
- Write a program to get inputs for width, depth, height and sweep for a house and compute and display volume of the house.