Expressions, Fragments of code that produces a values, whether it by unary, binary or ternary operators, or just simply writing a value.

Expressions can contain other expressions, sought of how a sentence can contain subsentences.

If an expression is akin to a sentence fragment. Then a full sentence is akin to a statement

A program is a list of statements

Statements can have side effects.

At the end of every statement, place a semi colon, till you learn more about missing semicolons

Bindings

Also known as variables

Let var = 5;

// let indicates the statement is going to be a binding

Rather than thinking of bindings or variables as boxes, think of them as tentacles.

Two bindings can refer to the same value.

If there are no tentacles attached to a value, it’s inaccessible to the program, ?maybe forever?

//A let statement can define multiple bindings

let var1 = “sean”, var2 = “bond”

//other binding statements

var var1 = ”sean” // works similar to let with some confusing properties, findout more in next chapter

const pi = 3.14

binding names may contain “$”,”\_”, alphanumeric character. Must not start with number

also cannot used keywords and “reserved for use”(for future JavaScript versions)

The environments

The collection of bindings and their values that exist at a given time is called the “environment”.

When a program starts up, the environment is not empty, it always contains bindings that are part

of the language standard.

Most of the time, the language provides bindings that help interact with the surrounding system.

(i.e reading mouse and keyboard input)

Functions

Executing a function is called “invoking”, “calling”, or “applying”.

Values given to functions are called arguments, functions have parameters to the argument.

Note:”Btw prompt() function isn’t use much in modern programs, as you have no control over the look of it, but it’s still good for experiments or toy programs

Console.log works in browsers(in dev tools)and Node.js

Functions can produce return value and or side effects

Parse functions

Number()

String()

Boolean()

loops

For (<before first loop>; <check condition before every loop>; <run after every loop>)

Usually best loops count from 0

Counters in general best to start from 0

break;

continue;

++

--

+=

-=

\*=

/=

switch (number) {

case 1 :

console.log(“you picked one”)

break;

case 2 :

console.log(“you picked two”)

default :

console.log(“you picked a number more than 2”)

some functions, such as Number(), has a capitalized first letter, this marks them as constructors, which we will talk about in chapt 6