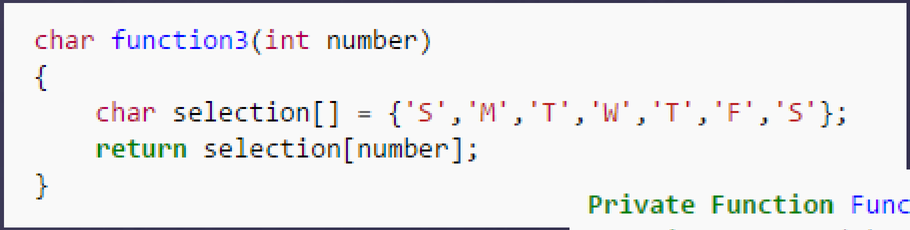
Functions & parameters

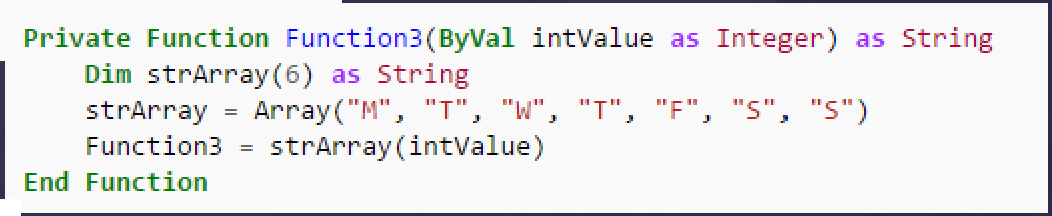
- What ?
- How ?
- Why ?

Functions & Parameters – What ?

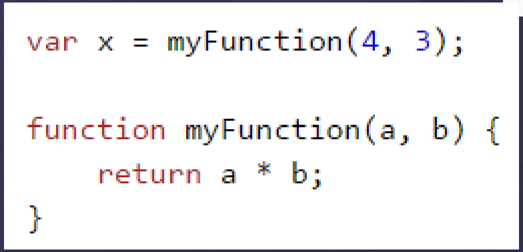
- Subroutine, procedure, routine, method or subprogram.



```
char function3(int number)
{
    char selection[] = {'S','M','T','W','T','F','S'};
    return selection[number];
}
```



```
Private Function Function3(ByVal intValue as Integer) as String
    Dim strArray(6) as String
    strArray = Array("M", "T", "W", "T", "F", "S", "S")
    Function3 = strArray(intValue)
End Function
```

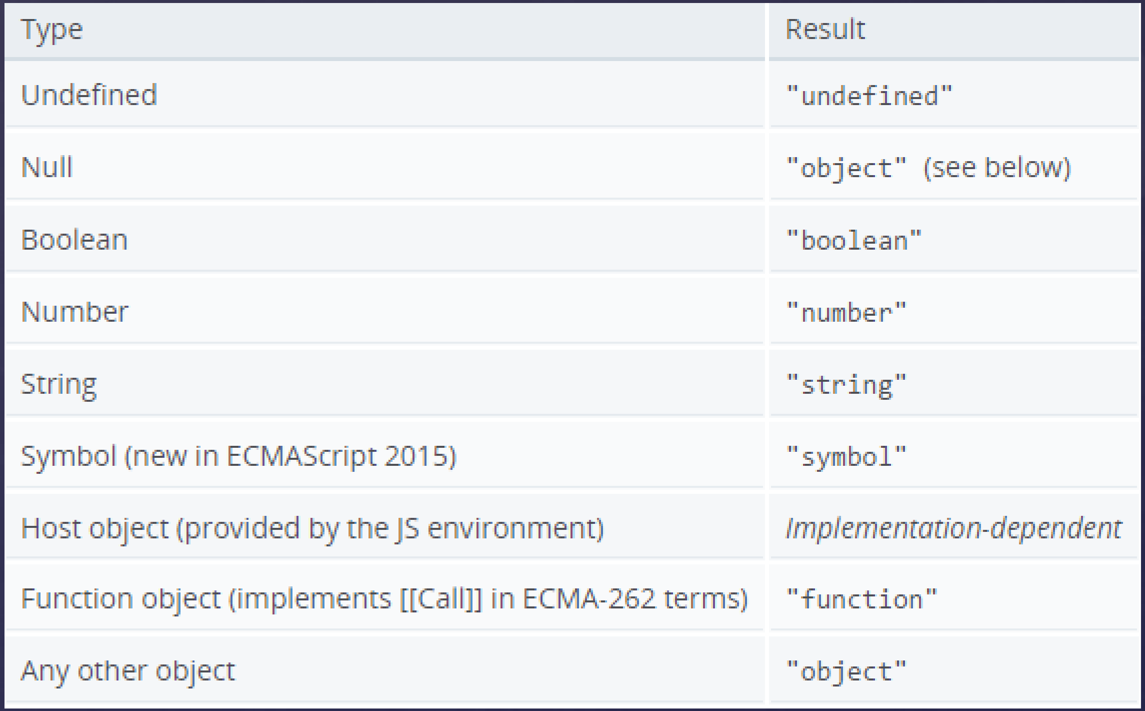


```
var x = myFunction(4, 3);

function myFunction(a, b) {
    return a * b;
}
```

Piece of code

Functions & Parameters – What ?

- Subroutine, procedure, routine, method or subprogram.
- Objects > properties and methods



Type	Result
Undefined	`"undefined"`
Null	`"object"` (see below)
Boolean	`"boolean"`
Number	`"number"`
String	`"string"`
Symbol (new in ECMAScript 2015)	`"symbol"`
Host object (provided by the JS environment)	*Implementation-dependent*
Function object (implements [[Call]] in ECMA-262 terms)	`"function"`
Any other object	`"object"`

Functions & Parameters – What ?

- Subroutine, procedure, routine, method or subprogram.
- Objects > properties and methods

```
function myFunction(a, b) {
    return arguments.length;
}
```

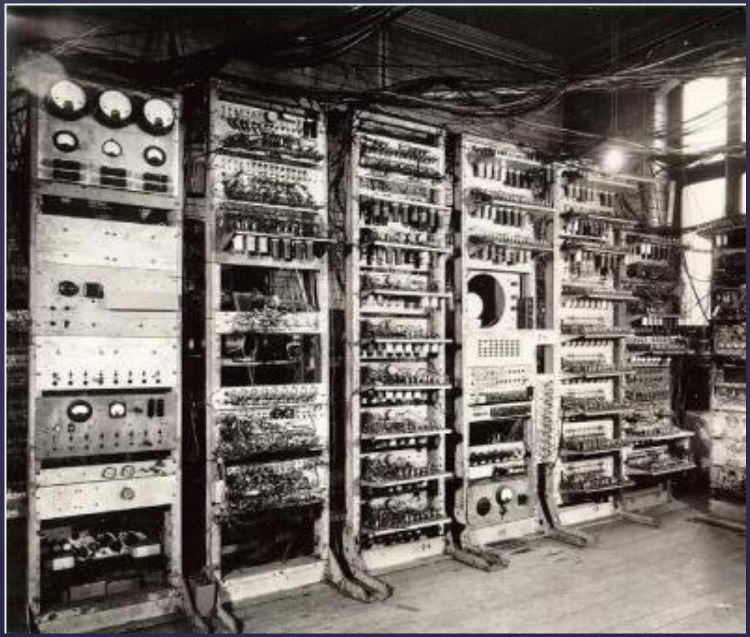
Length propertie

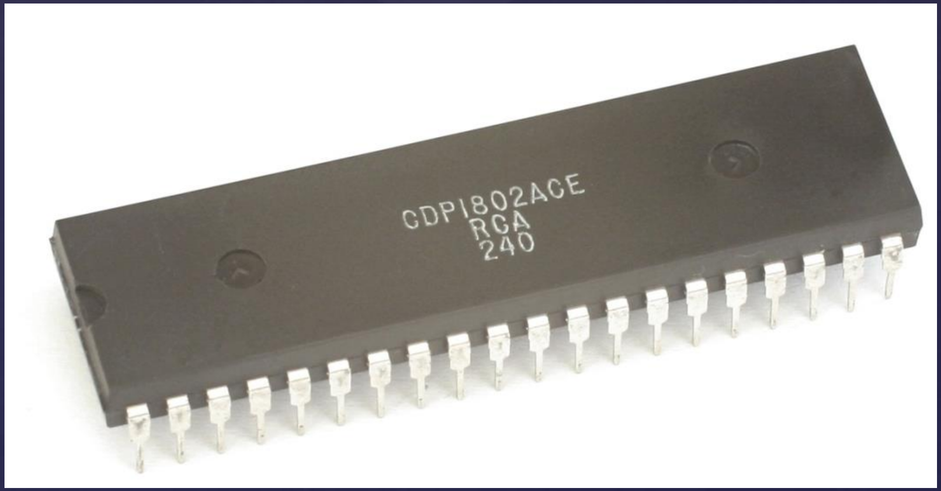
```
function myFunction(a, b) {
    return a * b;
}

var txt = myFunction.toString();
```

ToString method

Functions & Parameters – What ?

- Subroutine, procedure, routine, method or subprogram.
- Objects > properties and methods
- "The earliest computers and microprocessors, such as the *Small-Scale Experimental Machine* and the *RCA 1802*, did **not** have a single subroutine call instruction."



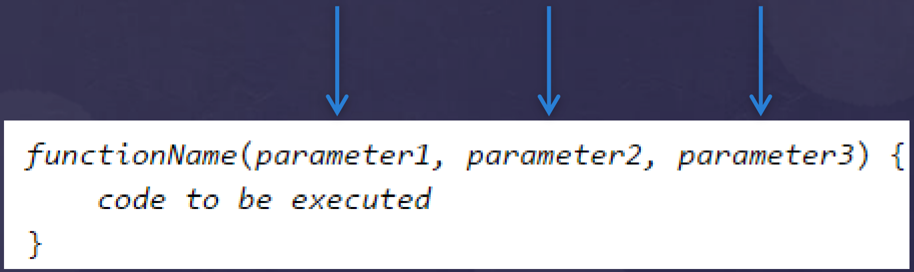


1960s

Functions & Parameters – What ?

- Names listed in the function definition

  The real values passed to the function



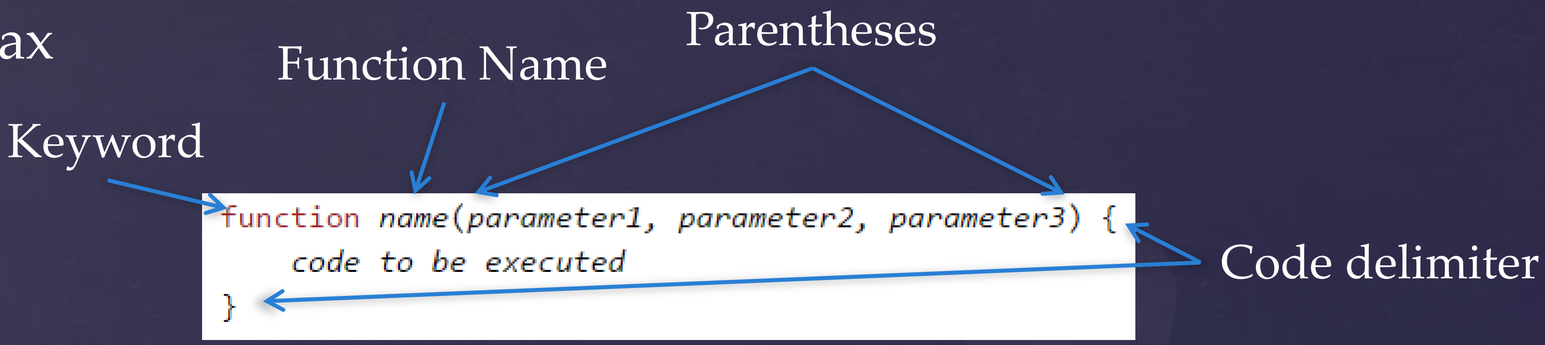
```
functionName(parameter1, parameter2, parameter3) {
    code to be executed
}
```

Functions & Parameters – What ?

- Names listed in the function definition
  The real values passed to the function
- Whatever you want and as much as you want
- Locals variables

Functions & Parameters – How ?

- Syntax



```
function name(parameter1, parameter2, parameter3) {
    code to be executed
}
```

Function names can contain letters, digits, underscores, and dollar signs

```
var x = function (a, b) {return a * b};
```

```
var myFunction = new Function("a", "b", "return a * b");
```

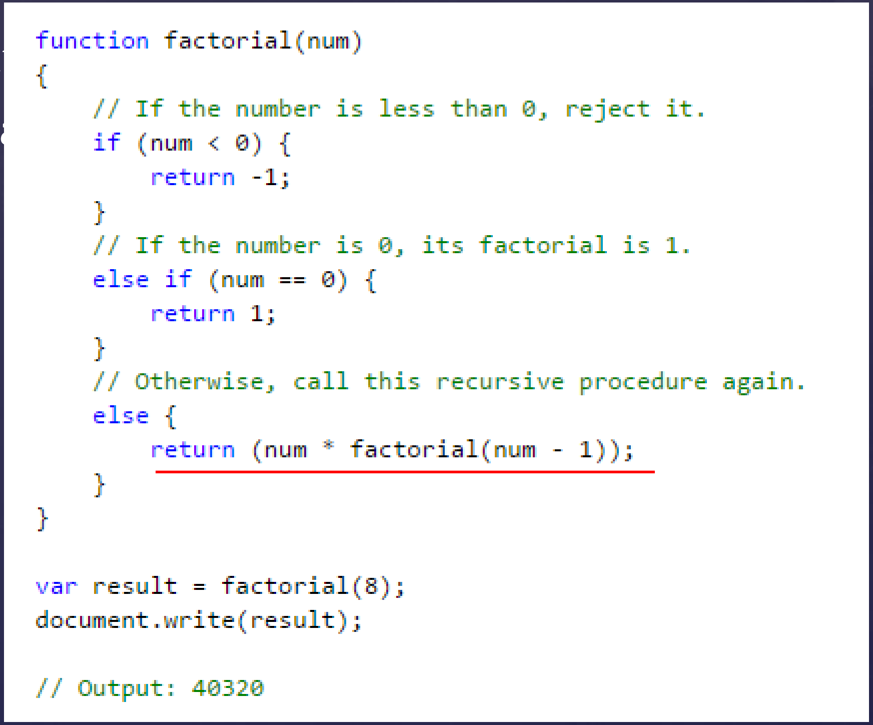
Functions & Parameters – How ?

- Syntax
- Invoke
  - When an event occurs
  - When it is called from code
  - Automatically (self invoked) - recursive

Functions & Parameters – How ?

- Syntax
- Invoke

  - When an event occurs
  - When i
  - Autom



```
function factorial(num)
{
    // If the number is less than 0, reject it.
    if (num < 0) {
        return -1;
    }
    // If the number is 0, its factorial is 1.
    else if (num == 0) {
        return 1;
    }
    // Otherwise, call this recursive procedure again.
    else {
        return (num * factorial(num - 1));
    }
}

var result = factorial(8);
document.write(result);

// Output: 40320
```

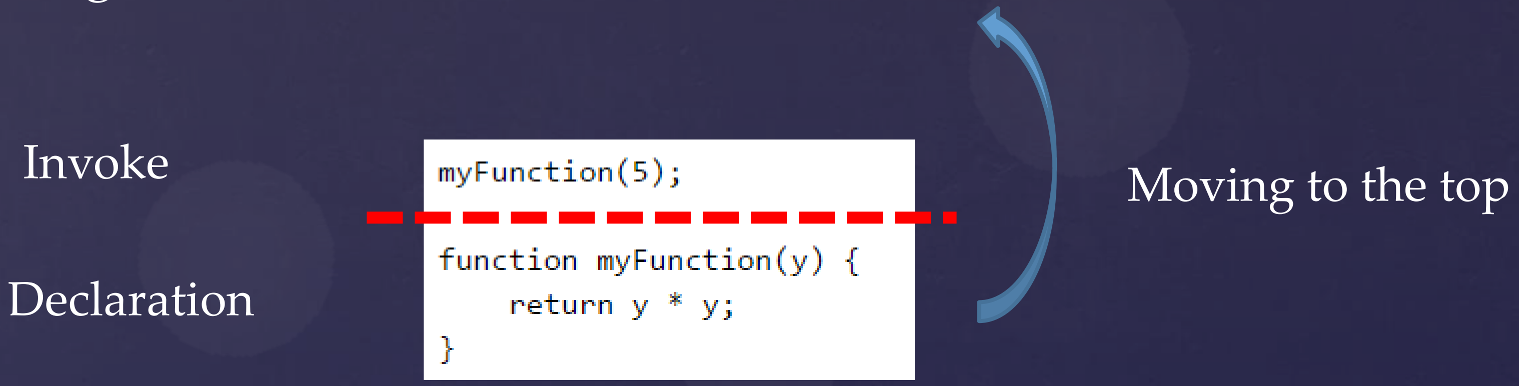
Functions & Parameters – How ?

- Syntax
- Invoke
  - When an event occurs
  - When it is called from code
  - Automatically (self invoked) - recursive

$$\begin{tikzcd}
\text{Invoked} \arrow[d] & & \text{Definition} \arrow[d] \\
\text{FunctionName()} & \text{=/=} & \text{FunctionName}
\end{tikzcd}$$

Functions & Parameters – How ?

- Syntax
- Invoke
- Hoisting



```
myFunction(5);

function myFunction(y) {
    return y * y;
}
```

Functions & Parameters – Why ?

- You're lazy !
- It's more readable
- For update

```
Function merci(){
return « Thank you for your
attention »;
}


console.log(merci());
```