ESX - The basics

Shehab Elhariry

Agenda

1. What is ESX?
2. Let vs. Const
3. Template Literals
4. Destructuring (Arrays, Objects)
5. Spread operator
6. Rest Parameter
7. New String functions
8. For…of
9. Default params
10. Arrow function
11. Babel (Getting Started)

What is ES9

> ES9 is the latest version of ECMA

> ECMA international

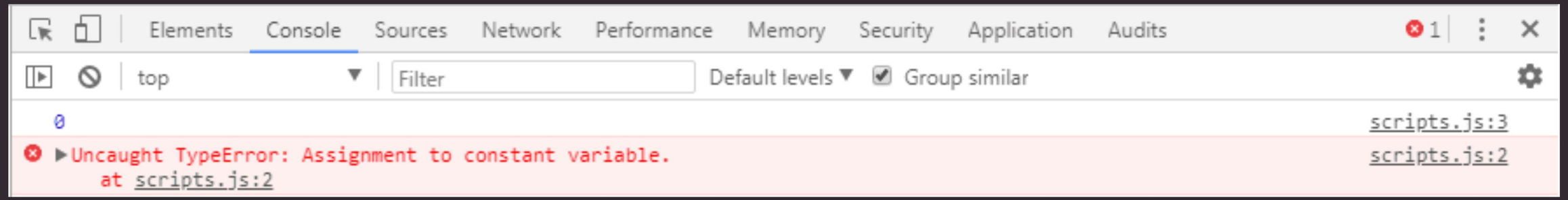
> ES6, ES2015 and harmony are used to mean almost the same thing.

> Know the code and dont concentrate on the version`

https://codeburst.io/javascript-wtf-is-es6-es8-es-2017-ecmascript-dca859e4821c

Appreciate the error

```
> A console error is a good thing.`
```



Let Vs. Const

> Declaring variables in ES5 was done using the var keyword

> The keyword const is used for values that you don’t intend to change

> In the old convention, we used to name constant values in all caps`

```
var PI = 3.14
```

Let Vs. Const

> Let is used for values that you intend to change a lot (In loops and re-asigns)

> The main difference between var and let is that let is scoped to the block {} while var is scoped to the function

```
let x = 4;

if (x == 4 ) {
  let x = 3;
  console.log(x);
}

console.log(x);
```

Template Literals

```
> If we had an array of names :`

  const Names    = [‘Ahmed’,‘Shehab’,‘Ramy’]

> and we want the output to be like that: `

Ahmed said: “I want to go on a trip with ‘Shehab’ and ‘Ramy’
```

Template Literals

> They are strings that include in them some variable you want to embed

> They are written between back quotes not ‘’ or “”

> The expression you want to write will be written inside ${some variable}`

> An awesome alternative to extensive concatenation with a lot of single and double quotes`

Destructuring

> Destructuring is the act of extracting values from arrays and objects in just a single step. (Very commonly used)`

```
const names = [‘ahmed’, ‘shehab’, ‘ali’];

let name_1 = names[0];
let name_2 = names[1];
let name_3 = names[2];
```

Destructuring (Arrays)

```
const names = [‘ahmed’, ‘shehab’, ‘ali’];

const [name_1, name_2, name_3] = names;

> This will do the same as the previous example`

> I can skip an index in case I didn’t want to destructure all the array`

const [name_1, , name_2] = names;
```

Destructuring (Objects)

>This can also be done to objects but with a very small twist.`

```
const person = {
      name: “Shehab”,
      age: 26
     };

const {name, age} = person;
```

> The variable names you choose must be the same as the name of the properties inside the object you are destructuring

Destructuring (code)

```
const arr = [1,2,3,4];

const item1 = arr[0];
const item2 = arr[1];
const item3 = arr[2];
const item4 = arr[3];

let [x,y,z] = arr;

console.log(x);
console.log(y);
console.log(z);
```

Destructuring (code)

```
const car = {
 fast: true,
 color: 'red',
 wheels: 4
}

var color = car.color;
var wheels = car.wheels;
var fast = car.fast;
```

```
const car = {
 fast: true,
 color: 'red',
 wheels: 4
}

const {fast, color, wheel} = car;
```

Spread Operator

> The spread operator, written with three consecutive dots (...), is new in ES6 and gives you the ability to expand, or spread, iterables into multiple elements.

```
const arr = [1,2,3,4];
console.log(...arr) // 1 2 3 4
```

Spread Operator continued

```
//spreading an array

const myArray1 = [1,2,3];
const myArray2 = ['Ahmed', 'Mohamed'];

const myNewArray = [...myArray1, ...myArray2];

// using spread operator as function params

function sum (a,b,c) {
  return a + b + c;
}

sum(...myArray1); // 6
```

Rest parameter

> The rest parameter, also written with three consecutive dots (...), allows you to represent an indefinite number of elements as an array.

```
const order = [89, 15, 'bigmacs', 'fries', 'pepsi'];

const [total, taxes, ...mealItems] = order;
```

Useful String functions

```
"string".startsWith(‘value’, start_index);

"string".endsWith(‘value’, start_index);

"string".includes(‘value’, start_index);
```

> A string function that searches in a string and check if it includes a certain value, returns a boolean`

For...Of

```
> In ES5 we used to loop over arrays with simple for loops
  or more complicated for…in loop`

  let names = [‘ahmed’,’mohamed’, ‘Ibrahim’]

  for (index in names) {
    console.log(names[index])
  }

This will get us each name.`
```

For...Of

> In ES6 the For…Of loop was introduced to handle interables and its syntax was much simpler by dropping the index`

```
let names = [‘ahmed’,’mohamed’, ‘Ibrahim’]

for (name of names) {
   console.log(name)
}
```

This will get us each name.

For...Of

> The For…Of can also deal with strings, in that case it goes through each letter at a time.`

```
let string = ‘OK’;

for (letter in string) {
   console.log(letter);
}
```

Default params

> ES6 make it possible to assign default parameters to functions to get a value instead of undefined`

```
function greet (greeting = ‘hello’) {
      console.log(greeting)
}

greet() //hello
```

Default params cont.

> Default parameters must come at the end of the params list

```
function multiply (a,b, c = 2) {
    return a * b * c;
}

console.log(multiply(5,6)) // 5 * 6 * 2 = 30

function multiply2 (x=2 , y ,z) {
    return x * y * z
}

console.log(multiply2(5,6)) // 5 * 6 * undefied = NAN
```

Default params cont.

> Default parameters are available to later default parameters

```
function complain (name='Ahmed', complain_body= `I hate ${name}`) {
  console.log(complain_body);
}

complain('Ramy');
```

Arrow functions

- ES6 introduces a new kind of function called the arrow function
- It can replace any other ES5 function by doing some steps and changing the syntax a bit.

```
const getName = () => ‘name’;
```

Expressions vs. declaration

```
// Declaration function

function sayHello () {
 console.log('hello');
}


// Expression function

const sayHello = function () {
 console.log('hello');
}
```

Arrow functions

> Arrow functions can only be used as an expression and cannot be used as a declaration

`- You can store them in variables
 - You can pass them as a callback function`

Arrow functions (steps)

```
Arrow function are expressions only
`
1- remove the function keyword
2- remove the return keyword
3- remove the semicolon
4- add an arrow ( => ) between the parameter
   list and the function body`
```

Arrow functions

```
> If the function takes one parameter we can
 remove the brackets ()`

 const myFunc = name => {
    console.log(name);
 }
```

Arrow functions

> If the body of the function is one statement only, we can remove the curly braces {} and in that case we don’t need a return keyword`

```
const myFunc = () => 'my name is Shehab Elhariry';

const es5Square = function (num) {
  return num * num;
}


// tada!
const es6Square = num => num * num;
```

Arrow functions examples

> No paramater, no returns

> No parameter, returns.

> 1 paramter, returns

> Callback function (Filter)

Babel

> Not all features of ES6 are supported by all browsers at the moment.

> https://kangax.github.io/compat-table/es6/

> We need transpilers that takes the new easier ES6 that we are learning and convert it to ES5 (old JS) That all browsers understand.

https://babeljs.io/repl/

Lab 1

1- In the js file, you will find a “”web designer” object.
change the “Your Name” placeholder to your name

2- Write a getAge() function that takes the years alive array and returns your age, save the value you return in a const of name age

3- Divide the web designer skills into 2 variables ‘designSkills’ and ‘developmentSkills’ (using ES6)

4- Uncomment the newSkills array and merge the developmentSkills array with the newSkills array in a new array ‘updatedDevSkills’

Lab 1

5- Destructure the diet array and using new ES6 write a function (getDrinks) that takes the newly created drinks variable and returns drinks that contain the letter ‘t’,
call that function and save the returned values in a variable ‘tDrinks’

6- Uncomment the function buildID().

In the return part f the function you will find a template literal. Replace the placeholder data with your actual data:

```
name => name
age => age (from getAge)
design Skills => designSkills
dev skills =>  updatedDevSkills
food => food
drinks => tDrinks
```

Lab 1

7- Add the HTML Fragment you created to the div with the class “card”

Hint* innerHTML