Erronka1. Javascript-ES6

Web garapena bezeroen ingurunean

1. Sarrera
2. Aldagaiak: let, const
3. Arrow functions
4. REST Parametroak
5. OOP
6. Desegituratzen (destructuring)
7. Template literals

Erronka1. Javascript-ES6

1. Sarrera

ECMAScript ECMA International-ek argitaratutako programazio-lengoaia estandarra da. Garapena 1996an hasi zen Netscap-ek proposatutako JavaScript hizkuntzatik abiatuta.

Orain arte ECMAScript 5 erabili dugu gehien bat baina orain ECMAScript 6 (ES6) JavaScripten bertsio eguneratua nolakoa den ikusiko dugu. Hobekuntza esanguratsuak sartu zituen web garapenerako hizkuntza modernizatuz eta sinplifikatuz. Adibidez, let, const, gezi-funtzioak (arrow functions), klaseak, promesak eta moduluak.

•Aurkezpen honetan ES6 bertsioaren ezaugarriak ezagutuko ditugu aurreko bertsioarekin alderatuz.
•Gogoan izan behar da ES5 eta ES6 bertsioak guztiz **bateragarriak** direla, eta, beraz, bata zein bestearen aginduak programa berean erabil daitezkeela.
•Aldagaiak eta konstanteak modu berri batean definitzen dira eta eremu edo **scope**-en garrantzia ulertu behar da.
•Funtzioen adierazpena laburtu egiten da gezi edo **arrow functions** erabiliz.
•**Desestructuring** kontzeptua ulertu eta kodean duen erabilgarritasuna ikusiko dugu.
•Objektuen metodo interesgarri batzuk ezagutuko ditugu eta **objektuei orientatutako programazioa** nola egin daiteken ikusi.

Manual osoa:

https://desarrolloweb.com/manuales/manual-de-ecmascript-6.html

Erronka1. Javascript-ES6

2. Aldagaiak: let, const

Aldagaien irismena (**scope**) adierazteko aukera ematen du, erabiltzen ari den blokera mugatuz.
Hau da, **var** erabilita definitzen ditugun aldagaiak, aldagai global edo lokal bat definitzen du, deklaratzen den blokearen eremua kontuan hartu gabe; **let-ek**, aldiz, kontuan hartzen du eremu hori.

Let erabiltzeko sintaxia kortxeteak aukerakoak direla kontutan izanda:

let var1 [= valor1] [, var2 [= valor2]] [, ..., varN [= valorN]];

index.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">
    <title>Document</title>
    <script type="text/javascript"
src="es6.js"></script>
</head>
<body>
    <h1>ES6 - ezaugarrien adibideak</h1>
</body>
</html>
```

es6.js

```
var persona = "Ada";
var persona = "Lovecode";
console.log (persona);

let persona2 = "Charles";
//let persona2 = "Babbage"; //errorea
console.log (persona2);

const persona3 = "Grace";
//persona3 = "Hopper"; //errorea
console.log (persona3);
```

Erronka1. Javascript-ES6

2. Aldagaiak: let, const

const array-ekin erabilita

Modu zorrotza, "use strict" idatzita, ES6 bertsiotik erabil daiteke, eta, besteak beste, ez du uzten deklaratu gabeko aldagaiak erabiltzen

es6.js

```
function cambiarArray(){
    "use strict";

    const MIARRAY = [1, 2, 3];
    console.log(MIARRAY);

    //MIARRAY = [4, 5, 6];  //Hau ezin da egin
    MIARRAY[0] = 4;
    MIARRAY[1] = 5;
    MIARRAY[2] = 6;
    console.log(MIARRAY);
}
cambiarArray();
```

Erronka1. Javascript-ES6

3. Arrow functions

Gezi funtzioek, **“arrow functions”,** betiko funtzioak modu trinkoan definitzea ahalbidetzen dute. Funtzioak balio bat itzultzen duen sententzia bat besterik ez badu, gezi funtzioak erabiliz, giltzak **“{ }”** eta **return** hitza ken ditzakegu. Parametroak erabilita ere, askoz ere laburragoa da kodea.

Sintaxia:
() => { sententziak }
(parametroa) => {sententziak } parametroa => {sententziak }

es6.js

```
var miFuncion = function () {
    return new Date();
}
var miFuncion = () => new Date();

var arraysConcatenados = function (array1, array2) {
    return array1.concat(array2);
}
console.log(arraysConcatenados([1,2],[3,4,5]));

var arraysConcatenados2 = (array1, array2) =>  array1.concat(array2);
console.log(arraysConcatenados2([1,2],[3,4,5]));
```

Erronka1. Javascript-ES6

4. REST Parametroak

REST parametroek, sarrerako parámetro kopuru ezezaguna array moduan maneiatzea ahalbidetzen digute, eta array hori definitzeko, hiru puntu jarri behar ditugu aurretik.

Sintaxia:

```
function(a, b, ...arrayBat) {
    // ...
}
```

es6.js

```
function miLista2(a, b, ...otrosElementos){
   console.log("a="+a);
   console.log("b="+b);
   console.log("Resto de elementos de la lista: ", otrosElementos);
 }
 miLista2("pera", "manzana", "zanahoria", "melón", "limón", "lima");
```

Erronka1. Javascript-ES6

5. OOP

Javascripten aurreko bertsioan **function** hitz erreserbatua erabiltzen genuen "klase" bat definitzeko (antzerako zerbait), ES6 bertsio honetan **class** erabiltzen da; gainera, propietateak metodo eraikitzaile baten barruan adierazi behar dira.

Sintaxia honako hau izango litzateke:

```
class NombreClase {
      constructor(parametro1 [,parametro 2...]) {
            this.propiedad1 = parametro1;
            [this.propiedad2 = parametro2;]
      }
}

let|const nombreObjeto = new NombreClase (argumentos);
```

Erronka1. Javascript-ES6

5. OOP

es6.js

```
class Telefono {
    constructor (marca, modelo){
      this.marca = marca;
      this.modelo = modelo;
    }
}

let miTelefono = new Telefono ("Google", "Pixel");
console.log(miTelefono.marca+" "+miTelefono.modelo);
```

Erronka1. Javascript-ES6

6. Desegituratzen (Desestructuring)

Array edo objektu bat elementu independenteetan banatu eta aldagaiei edo konstanteei balioa eman adierazpen bakar batean.

Hau da, aukera ematen digu array edo objektu bat zatitan banatzeko eta aldagai edo konstanteetara pasatzeko, bereizitako sententzietan egin beharrik gabe.

es6.js

```
//array-ekin
let cc, ba;

[cc, ba="Merida"] = ["Cáceres", "Badajoz"];
console.log(cc);
console.log(ba);

//objetuekin
const varios = {p: 11, q: true, r: "Hola" };
const {p, r} = varios;
console.log(p);
//console.log(q);
console.log(r);
```

Erronka1. Javascript-ES6

6. Desegituratzen (Desestructuring)

es6.js

```
//objetuekin (aldagai berriekin)
const objeto = {nombre: "Ada", apellido: "Lovecode"};

//objetuetik nombre hartzen dugu eta n aldagaira pasatu
const {nombre: n, apellido: a} = objeto;
console.log(n + " "+ a);

//objetuetik hartzen dugu nombre eta apellido
const {nombre, apellido} = objeto;
console.log(nombre + " " + apellido);

//objetuak deklaratzeko beste modu bat
let x, y;
({x, y} = {x: 1, y: 2});
console.log(x);
console.log(y);
```

Erronka1. Javascript-ES6

6. Template literals

“Template literals” bat string motako Javascript kate bati datuak egokitzeko modu berri bat da, garbiagoa eta zuzenagoa, saiakeran burua galdu gabe komatxoak zenbatzen.

es6.js

```
const resultado = sumar(1, 2, 3, 4, 5);

console.log(`La suma es: ${resultado}`);
```

Erronka1. Javascript-ES6

Ariketa: Ikasleen kudeaketa

ES5en garatutako javascript kodea duen webgune bat dugu eskola bateko ikasleak kudeatzeko. Kode hori ES6 bertsiora aldatu behar da.

Egin beharrekoak:

1. Let eta const erabili var ordez.
2. Aldatu eraikitzaileak klaseen arabera.
3. Ahal denean, erabili arrow functions.
4. Behar den lekuan rest parametroak aplikatu.
5. template literals erabili ahal den leku guztietan.
6. destructuring erabili objektuen balioak ateratzeko.

Erronka1. Javascript-ES6

```
// Constructor de Estudiante en ES5
function Estudiante(nombre, edad, ...asignaturas) {
    this.nombre = nombre;
    this.edad = edad;
    this.asignaturas = asignaturas;
  }

  // Añadir un método para saludar
  Estudiante.prototype.saludar = function() {
    console.log('Hola, me llamo ' + this.nombre + ' y tengo ' + this.edad + ' años.');
  };

  // Añadir un método para listar las asignaturas
  Estudiante.prototype.listarAsignaturas = function() {
    console.log('Mis asignaturas son: ' + this.asignaturas.join(', '));
  };

  // Crear un nuevo estudiante
  var estudiante1 = new Estudiante('Ana', 20, 'Matemáticas', 'Historia', 'Literatura');
```

Erronka1. Javascript-ES6

```
// Destructuring manual para extraer nombre y edad
  var nombre = estudiante1.nombre;
  var edad = estudiante1.edad;

  // Mostrar datos del estudiante
  console.log('Nombre: ' + nombre);
  console.log('Edad: ' + edad);

  // Mostrar saludo y asignaturas
  estudiante1.saludar();
  estudiante1.listarAsignaturas();

  // Función en ES5 para sumar notas
  function calcularPromedio() {
    var suma = 0;
    for (var i = 0; i < arguments.length; i++) {
      suma += arguments[i];
    }
    return suma / arguments.length;
  }

  // Calcular promedio de notas
  var promedio = calcularPromedio(85, 90, 78, 92);
  console.log('El promedio es: ' + promedio);
```