Arrow Function

IIC3585 - ES6

Características generales:

1. Las Funciones Flecha siempre son anónimas.

2. Son menos verbosas.

3. “This” se define según el contexto que rodea la función

Sintaxis

```
//Varios argumentos
(param1, param2, paramN) => { statements }
(param1, param2, paramN) => expression

//Un argumento
singleParam => { statements }
singleParam => expression

//Sin argumentos
() => { statements }

//Cuerpo entre paréntesis para retornar un objeto literal
params => ({foo: bar})
```

Situaciones y ejemplos

Diferencias en el orden

Declaration VS Expression

La función está definida antes de la ejecución:

```
6   //Use of function declaration
7   //The order does not affect the result
8   alert(fn());
9
10  function fn() {
11    return 'Hello world!';
12  }
```

Situaciones y ejemplos

Diferencias en el orden

Declaration VS Expression

```
18  //When using expressions, the order matters.
19  //Wrong use of expressions
20  alert(fho()); //'function fho doesn't exist'
21
22  var fho = () => {
23    return 'Heya';
24  }
25
26  //Correct use of expressions
27  var fho = () => {
28    return 'Heya';
29  }
30
31  alert(fho());
```

Situaciones y ejemplos

1. **Sintaxis**

Retornar objetos planos

```
// Al llamar func() retorna undefined
var func = () => {  foo: 1  };

//SyntaxError: function statement requires a name
var func = () => {  foo: function() {}  };

Se debe encerrar el objeto entre paréntesis
para que funcione como se espera:

//Funciona bien
var func = () => ({ foo: 1 });
```

Situaciones y ejemplos

1. **Diferencias**
2. Ventaja
3. Desventaja

Valor de this

```
function myFunction(p1, p2) {
    return p1 * p2;
}
```

Tiene this propio, se crea un contexto nuevo

```
(p1, p2) => { return p1 * p2 };
```

Valor de this se hereda de scope que lo rodea

Situaciones y ejemplos

1. Diferencias
2. **Ventaja**
3. Desventaja

Ventaja: No se necesita .bind() o similares

```
function Prefixer(prefix) {
    this.prefix = prefix;
}
Prefixer.prototype.prefixArray = function (arr) {
    var that = this;
    return arr.map(function (x) {
        return that.prefix + x;
    });
};

var pre = new Prefixer('Hi ');
var prefixed = pre.prefixArray(['Joe', 'Alex']);
prefixed.forEach(alert);
//prefixed = ['Hi Joe', 'Hi Alex']
```

Situaciones y ejemplos

1. Diferencias
2. **Ventaja**
3. Desventaja

Ventaja: No se necesita .bind() o similares

```
function Prefixer(prefix) {
    this.prefix = prefix;
}
Prefixer.prototype.prefixArray = function (arr) {
    return arr.map(x => {return this.prefix + x;});
};

var pre = new Prefixer('Hi ');
var prefixed = pre.prefixArray(['Joe', 'Alex']);
prefixed.forEach(alert);
//prefixed = ['Hi Joe', 'Hi Alex']
```

Situaciones y ejemplos

1. Diferencias
2. Ventaja
3. **Desventaja**

Desventaja: Contexto de los métodos de objetos

```
test = "attached to the module";

var foo = {
  test: "attached to an object"
};

foo.bar = () => {
 alert(this.test);
};

foo.bar();  //Imprime "atached to the module"
```