

A person's hands are shown typing on a vintage orange typewriter. The typewriter is placed on a dark wooden surface. In the background, there is a warm, golden sunset or sunrise with sunlight filtering through trees, creating a bokeh effect. A pair of glasses is visible on the wooden surface to the right of the typewriter. The overall mood is nostalgic and creative.

io.js

ES6 Features

io.js 2.0 released today!

Switching between Node.js & io.js versions

nvm (node version manager)

nvm makes it easy to switch between versions

<https://github.com/creationix/nvm>

```
nvm install iojs
```

nvm bash completion

- Add to your `.bash_profile` or `.bashrc` file

```
[ -r $NVM_DIR/bash_completion ] && . $NVM_DIR/bash_completion
```

Example

```
$ nvm [tab][tab]
```

alias	deactivate	install	ls	run	unload
clear-cache	exec	list	ls-remote	unalias	use
current	help	list-remote	reinstall-packages	uninstall	version

ES6 features available by default

- No runtime flag required
- But make sure to add "use strict" to top of module

See test examples here:

<https://github.com/tonypujals/demo-iojs-es6/>

block scoping features

let

Get used to it. Use it instead of var.

- declares a block scope local variable
- optionally initialized to a value

```
let var1 = value1;
```

const

- creates a read-only named constant

```
const name1 = value1
```

Function in blocks

Could always create a function inside of a function scope
Now can create a function inside of block scope as well

```
it ('greet function declared in two different scopes (function in block)', function() {  
  
    function greet() {  
        return 'woof';  
    }  
  
    if (true) {  
        // function defined inside block  
        function greet() {  
            return 'meow';  
        }  
  
        assert.equal(greet(), 'meow');  
    }  
  
    assert.equal(greet(), 'woof');  
  
});
```

Collections

- Map
- WeakMap
- Set
- WeakSet

Map

- simple key/value map
- keys can be any object, not just string
- advantages of map over object
 - object has a prototype with default keys
 - object keys must be strings
 - can easily get the size of a map

WeakMap

Use when you don't want to prevent garbage collection when there are no other references to an object other than the key.

- collection of key/value pairs
- keys must be objects
- keys are not enumerable

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/WeakMap

Set

A collection of unique values of any type.

- can store undefined
- can store NaN (will only store one instance even though `NaN !== NaN`)
- iterates in insertion order

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set

WeakSet

A collection of weakly held objects

- elements must be objects
- elements are not enumerable

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/WeakSet

Generators

function* defines a generator function, which returns a Generator object.

- simplifies iteration using `yield` to return a value (or throw an error) back to caller

```
function* name([param[, param[, ... param]]) {  
    statements  
}
```

```
function* gen() {  
    yield 1;  
    yield 2;  
    yield 3;  
}
```

```
var g = gen(); // "Generator { }"
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Generator

Promises

Promise

- a promise represents a pending value that will be provided in the future
- a pending promise becomes *settled* when it is *fulfilled* (succeeds) with a value or *rejected* (fails) with a reason
- alternative to using callbacks for asynchronous programming

Promisify function that expects a callback

Wrap function in another function that returns a Promise


```
function promiseFunc(arg) {  
    return new Promise(function (resolve, reject) {  
        asyncFunc(arg, function (err, result) {  
            return err ? reject(err) : resolve(result);  
        });  
    });  
}
```

Calling promisified function

Provide then and catch handlers

Can chain then handlers

```
promiseFunc(arg)
  .then(function (result) {
    ...
  })
  .catch(function (err) {
    ...
  });
```

Same thing but with "fat" arrow functions

```
promiseFunc(arg)
  .then((result) => {
    ...
  })
  .catch((err) => {
    ...
  });
```

Classes

Classes

- syntactic sugar over prototype-based OOP model
 - inheritance
 - constructor
 - super
 - static
 - get / set methods

ES6 features that still need flags

"Fat" arrow functions

Arrow functions

Finally ... **this** works!

Need to pass the `--harmony_arrow_function` option

```
iojs --harmony_arrow_functions index.js
```

or

```
mocha -- --harmony_arrow_functions
```

Compare

```
function Person(){  
    this.age = 0;  
  
    setInterval(function() {  
        this.age++;  
    }.bind(this), 1000);  
}  
  
var p = new Person();
```

Or

```
function Person(){  
    var that = this;  
  
    this.age = 0;  
  
    setInterval(function() {  
        that.age++;  
    }.bind(this), 1000);  
}  
  
var p = new Person();
```

To

```
function Person(){  
  this.age = 0;  
  
  setInterval(() => {  
    this.age++;  
  }, 1000);  
}  
  
var p = new Person();
```

Not Covered

Next time...

- binary and octal literals
- new string methods
- symbols
- template strings
- object literal extensions

Resources

Resources

<https://github.com/tonypujals/demo-iojs-es6>

<https://iojs.org/en/es6.html>

<http://davidwalsh.name/es6-io>

<https://github.com/lukehoban/es6features>

<http://www.2ality.com>

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/
Reference/Iteration_protocols](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Iteration_protocols)