Modern JS

@bengourley

1995

```
document.write(“Hello World”)
```

Microsoft® Internet Explorer® 6

2001

Windows® Internet Explorer® 7

2006

Windows® Internet Explorer® 8

2009

5 years

3 years

“Evergreen”!

Everything…

```
<script> … </script>
```

Just Works™

ES5 Array iterators

+ Internet Explorer 11

```
Array.prototype.forEach(…)

Array.prototype.map(…)

Array.prototype.filter(…)

Array.prototype.reduce(…)
```

ES5 Array iterators

+ Internet Explorer 11

e.g.

```
const players = [

  { firstName: ‘Andy’, lastName: ‘Murray’ },
  { firstName: ‘Roger’, lastName: ‘Federer’ },
  { firstName: ‘Novak’, lastName: ‘Djokovic’ }

]

players.map(player => `${player.firstName} ${player.lastName}`)
```

```
[ ‘Andy Murray’, ‘Roger Federer’, ‘Novak Djokovic’ ]
```

ES5 Object functions

+ Internet Explorer 11

`Object.keys(…)`

`Object.create(…)`

ES5 Object functions

+ Internet Explorer 11

e.g.

```
const player = {

  firstName: ‘Andy’,
  lastName: ‘Murray’

}

Object.keys(player)
```

```
[ ‘firstName’, ‘lastName’ ]
```

DOM Selectors, Events

+ Internet Explorer 11

```
element.querySelector(…)

element.querySelectorAll(…)


element.addEventListener()

event.stopPropagation()

event.preventDefault()
```

DOM Selectors, Events

+ Internet Explorer 11

e.g.

```
const btn = document.querySelector('form button[type=submit]')

btn.addEventListener('click', e => {

  // stop the form submitting to server
  e.preventDefault()

  // do our own thing
  if (validate()) sendMessage()

})
```

ES6 Array, Object

```
Array.prototype.find(…)
```

```
Object.assign(…)
```

ES6 Array, Object

e.g.

```
// before
jQuery.extend({}, defaults, options)

// after
Object.assign({}, defaults, options)
```

```
players.find(player => lastName === ‘Murray’)
{ firstName: ‘Andy’, lastName: ‘Murray’ }

players.find(player => lastName === ‘Gourley’)
  null
```

ES6 Template strings

```
`hello, world!`
```

ES6 Template strings

e.g.

```
const event = `Bathcamp`
console.log(`Hi everyone at ${event}!`)
  “Hi everyone at Bathcamp!”
```

```
`

multiline strings
are possible now
too :)

`
```

`ES6` variable declaration

+ Internet Explorer 11

`const`

`let`

ES6 Variable declaration

+ Internet Explorer 11

e.g.

```
const greeting = 'hi'

greeting = 'ciao' Error!
```

```
for (let i = 0; i < 10; i++ ) {
  // i is in this scope
}
// but not leaked in this scope
```

ES6 Variable declaration

+ Internet Explorer 11

e.g.

```
const player = {
  firstName: ‘Andy’,
  lastName: ‘Murray’
}

player.firstName = ‘Jamie’ No Error!
```

`ES6` Classes

+ Internet Explorer 11

`class`

`extends`

`constructor/super`

ES6 Classes

+ Internet Explorer 11

e.g.

```
class TennisPlayer extends Person {

  constructor (firstName, lastName) {
    super(firstName, lastName, ‘tennis player’)
  }

  playShot() {
    // method body
  }
}
```

ES6 Map, Set

+ Internet Explorer 11

```
new Map()

new Set()
```

ES6 Map, Set

+ Internet Explorer 11

e.g.

```
const m = new Map()
m.set('a', 10)
m.get('a')  -> 10
```

```
const s = new Set()
s.add('a')
s.add('b')
s.forEach(fn), s.has(item), s.clear()
```

ES6 Functions

arrow functions

default parameters

ES6 Functions

e.g.

```
const square = n => Math.pow(n, 2)

const volume = (x, y, z) => x * y * z

const getPlayerMatchData = (matchId, playerId) => {
  const match = matches.get(matchId)
  return match.players.find(p => p.id === playerId)
}
```

```
const drawLine = (vector, strokeWidth = 1) => {
  // draw it!
}
```

ES6 Rest, Spread

```
...
```

ES6 Rest, Spread

e.g.

```
// rest
const fn = (a, b, ...args) => {}
fn(1, 2, 3, 4, 5, 6) -> args = [ 3, 4, 5, 6 ]

// spread
const values = [ 14, 23, 4, 8, 19 ]
Math.max(...values) -> 23
```

`ES6` Promises

`new Promise()`

`Promise.all(…)`

`Promise.race(…)`

ES6 Promises

e.g.

```
const myAsyncFn = () => new Promise((resolve, reject) => {
  // do something then…
  resolve(data)
  // or…
  reject(new Error(‘something broke’))
})

myAsyncFunction()
  .then(() => console.log(‘done’))
  .catch(err => console.error(err))
```

ES2016+ PAD LEFT!!!!

`String.prototype.padStart()`

`String.prototype.padEnd()`

ES2016+ async, await

async function

await keyword

ES2016+ async, await

e.g.

```
const run = async () => {
  try {
    await myAsyncFunction()
    console.log('done')
  } catch (err) {
    console.error(err)
  }
}

run()
```

References

http://caniuse.com/

http://kangax.github.io/compat-table/

https://developer.mozilla.org/bm/docs/Web/JavaScript