

Constants

```
> const EULER = 2.7182818284
> EULER = 13
> EULER
> 2.7182 818284 // Value stays
the same
Warning! If array or object, the
reference is kept
constant. If the constant is a
reference to an object,
you can still modify the
content, but never change the
variable.
> const CONSTANTS = []
> CONSTA NTS.pu sh( EULER)
> CONSTANTS
> [ 2.7182 818284 ]
> CONSTANTS = { 'euler':
2.7182 818284 }
> CONSTANTS
> [ 2.7182 818284 ]
```

Object Notation

```
// Computed properties
> let key = new Date().ge -
tTime()
> let obj = { [key]: "value" }
> obj
> { '14599 588 82881': 'value' }
// Object literals
balloon = { color, size };
// Same as
balloon = {
  color: color,
  size: size
}
// Better method notations
obj = {
```

Object Notation (cont)

```
> foo (a, b) { ... },
bar
```

Destructuring Arrays

```
> let [a, b, c, d] = [1, 2, 3,
4];
> consol e.l og(a);
> 1
> b
> 2
```

Destructuring Objects

```
> let luke = { occupation:
'jedi',
  father: 'anakin' }
> let {occup ation, father} =
luke
> consol e.l og( occ upa tion,
father)
> jedi anakin
```

Spread Operator

```
// Turn arrays into comma
separated
// values and more
> function logger (...args) {
  con sol e.l og('%s argume -
nts',
  arg s.l ength)
  arg s.f orE ach (co nso -
le.log)
  // arg[0], arg[1], arg[2]
}
```

let vs var

```
> var average = 5
> var average = (average + 1) /
2
> average
> 3
> let value = 'hello world'
> let value = 'what is new'
```

let vs var (cont)

```
> // -> throws TypeError: Identifier 'value'
has
already been declared
Be aware of Temporal Dead Zones:
> console.log(val) // -> 'undefined'
> var val = 3
> console.log(val) // -> 3
Because it's equivalent to:
> var val
> console.log(val)
> val = 3
> console.log(val)
Variables declared with "let/const" do not
get hoisted:
> console.log(val)
// -> Throws ReferenceError
> let val = 3
> console.log(val) // -> 3
```

Promises

```
new Promise((resolve, reject) =>
{
  req ues t.g -
et(url, (error, response,
body) =>
{
  if (body)
{
  -
res olv e(J SON.pa rse (bo dy));
} else {
-
res olv e({});
}
})
}).t hen(() => { ...
})
.ca tch ((err) =>
throw err)
// Parall elize tasks
```

Promises (cont)

```
> Promise.all([
  promise1, promise2, promise3
]).then(() => {
  // all tasks are finished
})
```

Classes, Inheritance, Setters, Getters

```
class Rectangle extends Shape {
  constructor(id, x, y,
    w, h) {
    super(id, x, y)
    this.width = w
    this.height = h
  }
  // Getter and setter
  set width(w) {
    this._width = w
  }
  get width() {
    return this._width
  }
}
class Circle extends Shape {
  constructor(id, x, y,
    radius) {
    super(id, x, y)
    this.radius = radius
  }
  do_a(x) {
    let a = 12;
    super.do_a(x + a);
  }
  static do_b() { ... }
}
```

Classes, Inheritance, Setters, Getters (cont)

```
> Circle.do_b()
```

Binary, Octal and Hex Notation

```
> 0b1001011101 // 605
> 0o6745 // 3557
> 0x2f50a // 193802
```

Arrow Function

```
> setTimeout(() => {
  ... console.log('delayed')
  ... }, 1000)
```

Equivalent with Anonymous Function

```
> setTimeout(function () {
  ... console.log('delayed')
  ... }.bind(this), 1000)
```

Equivalent with IIFE

```
> (function () {
  ... var cue = 'Luke, I am your father'
  ... console.log(cue) // 'Luke, I am -
  ... }())
> console.log(cue) //
Reference Error
```

New Scoped Functions

```
> {
  ... let cue = 'Luke, I am your father'
  ... console.log(cue)
  ... }
> 'Luke, I am your father'
```

String Interpolation, Thanks to Template Literals

```
> const name = 'Tiger'
> const age = 13
> console.log(`My cat is named ${name} and is ${age} years old.`)
> My cat is named Tiger and is 13 years old.
// We can preserve newlines...
let text = (`cat
dog
nickel odeon`
)
```

Default Params

```
> function howAreYou (answer = 'ok') {
  console.log(answer)
```