JAVASCRIPT CORECONCEPTS

FABIO**BIONDI**.io

let

```
let a = 1;
a = 2;
```

const

```
const b = { name: 'Fabio' }

b = { name: 'Mario' } // GENERATE EXCEPTION
b.name = 'Ciro';      // ALLOWED BUT DON'T DO IT ;)
```

Block Scope

```
let a = 2;
{
  let b = 20;
}
console.log(a) // 2
console.log(b) // undefined
```

Template Literals

```
const name = 'Fabio';
const age = 21;
console.log( `${name} is ${age * 2}` );
```

Destructuring

```
const user = {
  name: 'Fabio',
  surname: 'Biondi',
  coords: { lat: 43, lng: 12 }
};

const {
  name, surname, role: r = 'admin', coords: { lat, lng }
} = user ;

console.log (name, surname, r, lat, lng);
// Fabio Biondi Admin 43 12
```

Short Object Syntax

```
const name = 'Fabio'
const surname = 'Biondi'
const params = { name, surname }
```

Array Spread Operator

```
const list = [1, 2, 3];
foo(...list);
foo(a, b, c) { /* do something */ }
```

Object Spread Operator vs Object.assign:

```
const obj1 = { name: 'Fabio' }
const obj2 = { surname: 'Biondi' }

// clone objects
const cloned1 = { ...obj1 }
const cloned2 = Object.assign ({}, obj1)

// merge objects
const merge1 = Object.assign ({}, obj1, obj2, { id: 123 } )
const merge2 = {...obj1, ...obj2, id: 123}
```

Arrow Functions

```
const pow = a => a * a;
const add = (a, b) => a + b;
const divide = (a = 0, b = 1) => {
  return a / b
}
const getUser = () => ({name: 'Mario})
```

Array methods (map, filter, ...)

```
const users = [
  {"name": "Lisa", "age": 35},
  {"name": "Silvia", "age": 2},
  {"name": "Fabio", "age": 25},
]

const result = users.filter(user => user.age > 18)
                    .map(user => user.name)

// output ["Lisa", "Fabio"]
```

For...in vs For...of

```
const list = [
  { label: 'Fabio' },
  { label: 'Lisa' },
];

// ES5 for...in (avoid, you should use it just for objects)
for (const key in list) {
  console.log( list[key].label );          // 0 Fabio, 1 Lisa
}
// ES6 for...of
for (const user of list) {
  console.log( user.label );               // Fabio, Lisa
}
```

Map

```
const users = new Map()
const myInstance = {}                     // any instance
users.set( 100, { label: 'Silvia' }  )
users.set( myInstance, { someValue: 123 } )
console.log(users.get( 100 ))            // { "label": "Silvia" }
console.log(users.get( myInstance ))     // { "someValue": 123 }
console.log(users.size)                  // 2
```

Promise

```
const doStuff = new Promise((resolve, reject) => {
  // asynchronous stuff (i.e. XHR request, timers, ...)
  setTimeout(() => resolve('hello'), 2000);
});

doStuff.then(
  res => console.log(res), // hello
  err => console.log(err), // some error messages
);
```

Class

```
class MyClass {
  hello(name) {
    console.log(`Hi ${name}`)
  }
}

const a = new MyClass()
a.hello('Fabio')    // Hi Fabio
```

Module: import / export

```
import { doSomething } from './path/fileName'

export function doSomething() { }
```

Module: export default

```
import AnyName from './path/fileName'

// export class
export default class MyComponent { }
// or functions:
const add = (a, b) => a + b;
export default add;
// or more concise:
export default (a, b) => a + b;
```

Module: import all

```
import * as Utils from './path/fileName'
```

Immutability

```
const users = [
  { id: 1, name: 'Silvia' },
  { id: 2, name: 'Fabio' },
  { id: 3, name: 'Lorenzo' }
];

// ADD
const user = { id: 5, name: 'Lisa' }
const result = [...users, user]

// DELETE
const id = 2;
const result = users.filter(user => user.id != id)

// EDIT
const updatedUser = {id: 1, name: 'Mario' }
const result = users.map(user => {
  return user.id === updatedUser.id ? updatedUser : user
})
```