# DT lab #02

# Javascript – “this”

- In JS “this” behaves differently
- In most cases, the value of this is determined by how a function is called (runtime binding).
- The bind() method it is introduced to set the value of a function's this regardless of how it's called
- It has different output in case of strict mode and non-strict mode.
- *Non-strict mode* – is always a reference to an object - *a property of an execution context (global, function or eval)*,
- *Strict mode* - can be any value.
- https://www.javascripture.com/Function#bind

# “this” - example

```
const test = {
prop: ‘message’,
func: function() {
    return this.prop;
  },
};
console.log(test.func());

// expected output: “message”
```

Global context -In the global execution context (outside of any function), this refers to the global object whether in strict mode or not.

```
// In web browsers, the window object is also the global object:
console.log(this === window); // true
a = 37;
console.log(window.a); // 37
this.b = "MDN";
console.log(window.b) // "MDN"
console.log(b) // "MDN"
```

Function context - Inside a function, the value of this depends on how the function is called.

```
// non strict mode
function f1() {
  return this;
}
// In a browser:
f1() === window; // true
// In Node:
f1() === global; // true

// strict mode
function f2() {
  'use strict'; // see strict mode
  return this;
}
f2() === undefined; // true
```

# The bind method

- Calling *f.bind(someObject)* creates a new function with the same body and scope as f, but where this occurs in the original function, in the new function it is permanently bound to the first argument of bind, regardless of how the function is being used.

```
function f() {
  return this.a;
}

var g = f.bind({a: 'azerty'});
console.log(g()); // azerty

var h = g.bind({a: 'yoo'}); // bind only works once!
console.log(h()); // azerty

var o = {a: 37, f: f, g: g, h: h};
console.log(o.a, o.f(), o.g(), o.h()); // 37,37, azerty, azerty
```

# Arrow functions

- In arrow functions, this retains the value of the enclosing lexical contexts this. In global code, it will be set to the global object:

```
var globalObject = this;
var foo = (() => this);
console.log(foo() === globalObject); // true
```

- If “this” arg is passed to call, bind or apply on invocation of an arrow function it will be ignored.

```
// Call as a method of an object
var obj = {func: foo};
console.log(obj.func() === globalObject); // true

// Attempt to set this using call

console.log(foo.call(obj) === globalObject); // true

// Attempt to set this using bind
foo = foo.bind(obj);
console.log(foo() === globalObject); // true
```

# Function - bind

- // The following result in equivalent behavior, but the first
- // should be used unless the body changes at run time.
- var f1 = function(x, y) { return x + y; };
- var f2 = Function('x', 'y','return x + y;');
- var f3 = Function('x, y', 'return x + y;');
- var f4 = new Function('x', 'y', 'return x + y;');

- // The following demonstrate creating generator and async functions
- var GeneratorFunction = Object.getPrototypeOf(function*(){}).constructor;

- // The following are equivalent:
- var g1 = function*(x, y) { yield x; yield y; }
- var g2 = GeneratorFunction('x', 'y', 'yield x; yield y;');
- var g3 = new GeneratorFunction('x, y', 'yield x; yield y;');

- var AsyncFunction = Object.getPrototypeOf(async function(){}).constructor;

- // The following are equivalent:
- var a1 = async function(url) { return (await fetch(url)).status; }
- var a2 = AsyncFunction('url', 'return (await fetch(url)).status;');
- var a3 = new AsyncFunction('url', 'return (await fetch(url)).status;');

# “this” with Vue.js

```
var globalObject = this;
var foo = (() => this);
console.log(foo() === globalObject); // true

var app = new Vue({
  el: '#app',
  data: {
    message: ''
  },

  methods: {
    process: function(){
     console .log ( this.message);
    }
  }
})
```

```
// Pure JS
this.classList.toggle('active');    //  is just the reference

// jQuery
$(this).toggleClass('active');       // adds the jQuery library to the reference
```