

Closures

El amigo incomprendido de Javascript.

FUNCIONES

Junto a los Objetos
son actores principales.

FUNCIONES

Como Freddy
proveen **CONTEXTO.**

FUNCIONES

```
var sabor = "Mermelada";

function crearPizza(rebanadas) {
  var sabor = "Margarita";

  var helper = function() {
    var sabor = "Pepperoni";

    console.log("La mia la quiero con " + sabor +
                " y " + rebanadas + " rebanadas");
  }

  console.log("Hacer una pizza se trata sobre todo del
sabor " + sabor);
  helper();
}
crearPizza(8);
```

FUNCIONES

CONTEXTO GLOBAL

```
var sabor = "Mermelada";

function crearPizza(rebanadas) {
  var sabor = "Margarita";

  var helper = function() {
    var sabor = "Pepperoni";

    console.log("La mia la quiero con " + sabor +
               " y " + rebanadas + " rebanadas");
  }

  console.log("Hacer una pizza se trata sobre todo del
sabor " + sabor);
  helper();
}
crearPizza(8);
```

FUNCIONES

CONTEXTO GLOBAL

```
var sabor = "Mermelada";
```

CONTEXTO crearPizza

```
function crearPizza(rebanadas) {  
  var sabor = "Margarita";  
  
  var helper = function() {  
    var sabor = "Pepperoni";  
  
    console.log("La mia la quiero con " + sabor +  
               " y " + rebanadas + " rebanadas");  
  }  
  
  console.log("Hacer una pizza se trata sobre todo del  
sabor " + sabor);  
  helper();  
}  
crearPizza(8);
```

FUNCIONES

CONTEXTO GLOBAL

```
var sabor = "Mermelada";
```

CONTEXTO crearPizza

```
function crearPizza(rebanadas) {  
  var sabor = "Margarita";
```

CONTEXTO helper

```
  var helper = function() {  
    var sabor = "Pepperoni";  
  
    console.log("La mia la quiero con " + sabor +  
               " y " + rebanadas + " rebanadas");  
  }
```

```
    console.log("Hacer una pizza se trata sobre todo del  
sabor " + sabor);  
    helper();  
}
```

```
crearPizza(8);
```

FUNCIONES

CONTEXTO GLOBAL

```
var sabor = "Mermelada";
```

CONTEXTO crearPizza

```
function crearPizza(rebanadas) {  
  var sabor = "Margarita";
```

CONTEXTO helper

```
  var helper = function() {  
    var sabor = "Pepperoni";  
  
    console.log("La mia la quiero con " + sabor +  
                " y " + rebanadas + " rebanadas");  
  }
```

```
  console.log("Hacer una pizza se trata sobre todo del  
sabor " + sabor);  
  helper();  
}
```

```
crearPizza(8);
```


FUNCIONES

CONTEXTO GLOBAL

```
var sabor = "Mermelada";
```

CONTEXTO crearPizza

```
function crearPizza(rebanadas) {  
  var sabor = "Margarita";
```

CONTEXTO helper

```
  var helper = function() {  
    var sabor = "Pepperoni";  
  
    console.log("La mia la quiero con " + sabor +  
               " y " + rebanadas + " rebanadas");  
  }
```

```
  console.log("Hacer una pizza se trata sobre todo del  
sabor " + sabor);  
  helper();  
}
```

```
crearPizza(8);
```

RESULTADO

```
crearPizza(8);
```

Hacer una pizza se trata sobre todo del sabor **Margarita**

La mia la quiero con **Pepperoni** y 8 rebanadas

Javascript utiliza un **Contexto Léxico**.

Utilizando el valor de las variables
al momento de **definir** la función/método.

Ahora, si. Closures

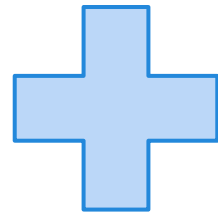
CLOSURES ES:

Una función que es evaluada a través del contexto léxico en la que fue definida, junto a las variables libres asociadas.

WAT?

CLOSURES ES:

Una declaración de función.



El contexto en la que
fue declarada.

CLOSURES ES:

```
function crearSaludo(nombre)
{
  var meVes = "Si";

  return function() {
    if(meVes) {
      console.log(meVes+' me ves, '+nombre);
    }
  }
}

var saludo = crearSaludo('pablo');
```

CLOSURES ES:

```
function crearSaludo(nombre)
{
  var meVes = "Si";

  return function() {
    if(meVes) {
      console.log(meVes+' me ves, '+nombre);
    }
  }
}

var saludo = crearSaludo('pablo');
```


CLOSURES ES:

```
function crearSaludo (nombre)
{
  var meVes = "Si";

  return function() {
    if(meVes) {
      console.log(meVes+' me ves, '+nombre);
    }
  }
}

var saludo = crearSaludo('pablo');
```

CLOSURES ES:

```
function crearSaludo(nombre)
{
  var meVes = "Si";

  return function() {
    if(meVes) {
      console.log(meVes+' me ves, '+nombre);
    }
  }
}

var saludo = crearSaludo('Pablo');
saludo();
"Si me ves, Pablo".
```

CLOSURES ES:

```
function crearSaludo(nombre)
{
  var meVes = "Si";

  return function() {
    if(meVes) {
      console.log(meVes+' me ves, '+nombre);
    }
  }
}

var saludo = crearSaludo('Christian');
saludo();
"Si me ves, Christian".
```

Y si devolvemos una función...
...que acepte parámetros?

CLOSURES ES:

```
function crearSaludo(nombre)
{
  return function(numero) {
    console.log(
      nombre+', me ves '+numero+' veces'
    );
  }
}
var saludoPablo = crearSaludo('Pablo');
var saludoChris = crearSaludo('Christian');
```

CLOSURES ES:

```
function crearSaludo(nombre)
{
  return function(numero) {
    console.log(
      nombre+', me ves '+numero+' veces'
    );
  }
}
var saludoPablo = crearSaludo('Pablo');
var saludoChris = crearSaludo('Christian');
```

CLOSURES ES:

```
var saludoPablo = crearSaludo('Pablo');  
var saludoChris = crearSaludo('Christian');
```

```
saludoPablo(2);
```

```
"Pablo, me ves 2 veces"
```

```
saludoPablo(7);
```

```
"Pablo, me ves 7 veces"
```

```
saludoChris(10)
```

```
"Christian, me ves 7 veces"
```

ZOMFG!

:O

(Y esto para qué sirve?)

USANDO CLOSURES

- **Fábrica de funciones**
- **Ocultar código**
- **Definir Módulos**
- **Extender el lenguaje**
(si da el tiempo, ejemplo)

Closures

FABRICANDO FUNCIONES

FABRICANDO FUNCIONES

```
function crearSaludo(nombre)
{
  return function(numero) {
    console.log(
      nombre+', me ves '+numero+' veces'
    );
  }
}
var saludoPablo = crearSaludo('Pablo');
var saludoChris = crearSaludo('Christian');
```

FABRICANDO FUNCIONES

```
function resizer(size)
{
    return function() {
        document.body.style.fontSize = size+'px';
    }
}
var size14 = resizer(14);
var size16 = resizer(16);
var size18 = resizer(18);
```

FABRICANDO FUNCIONES

```
$('#size14').on('click', size14);  
$('#size16').on('click', size16);  
$('#size18').on('click', size18);
```

```
<a href="#" id="size14">14</a>  
<a href="#" id="size16">16</a>  
<a href="#" id="size18">18</a>
```

Closures

OCULTAR CÓDIGO

PRIVACIDAD DEL CÓDIGO

```
var Contador = (function() {  
  var _contador = 0;  
  return {  
    incrementar : function() {  
      _contador++;  
    },  
    decrementar : function() {  
      _contador--;  
    },  
    ver : function() {  
      console.log(_contador);  
    },  
  }  
}) ();
```


PRIVACIDAD DEL CÓDIGO

```
var Contador = (function() {  
  var _contador = 0;  
  return {  
    incrementar : function() {  
      _contador++;  
    },  
    decrementar : function() {  
      _contador--;  
    },  
    ver : function() {  
      console.log(_contador);  
    },  
  }  
}) ();
```

INTERFAZ PUBLICA

PRIVACIDAD DEL CÓDIGO

```
var Contador = (function() {  
  var _contador = 0;  
  return {  
    incrementar : function() {  
      _contador++;  
    },  
    decrementar : function() {  
      _contador--;  
    },  
    ver : function() {  
      console.log(_contador);  
    },  
  }  
}) ();
```

INTERFAZ PRIVADA

INTERFAZ PUBLICA

PRIVACIDAD DEL CÓDIGO

```
Contador.ver();
```

0

```
Contador.incrementar();
```

```
Contador.incrementar();
```

```
Contador.incrementar();
```

```
Contador.ver();
```

3

```
Contador.decrementar();
```

```
Contador.decrementar();
```

```
Contador.ver();
```

1

Closures

DEFINIR MÓDULOS

Quizás lo más importante
que veamos sobre Closures

Lo bueno, es que ya lo vimos.

MÓDULOS

```
var Contador = (function() {  
  var _contador = 0;  
  return {  
    incrementar : function() {  
      _contador++;  
    },  
    decrementar : function() {  
      _contador--;  
    },  
    ver : function() {  
      console.log(_contador);  
    },  
  }  
}) ();
```

Closure Anónimo

```
(function() {  
    /*se ejecuta enseguida*/  
})();
```


MÓDULOS

```
var Contador = (function() {  
  var _contador = 0;  
  return {  
    incrementar : function() {  
      _contador++;  
    },  
    decrementar : function() {  
      _contador--;  
    },  
    ver : function() {  
      console.log(_contador);  
    },  
  }  
})();
```

MIXINS

```
var Emergencias = (function($, mod2) {  
  
    function privado_alertar() {  
        $('#alerta').html('Socorro!');  
        mod2.notificar();  
    }  
  
    return {  
        alertar : privado_alertar  
    }  
})(jQuery, otroModulo);
```

EXTENSIÓN DE MÓDULOS

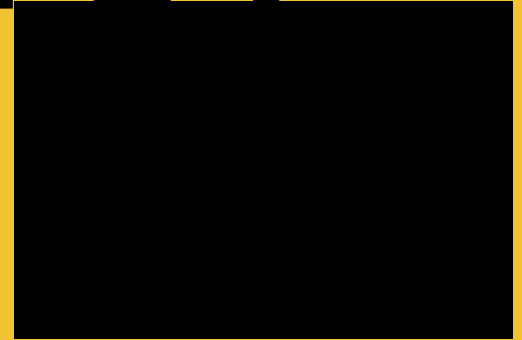
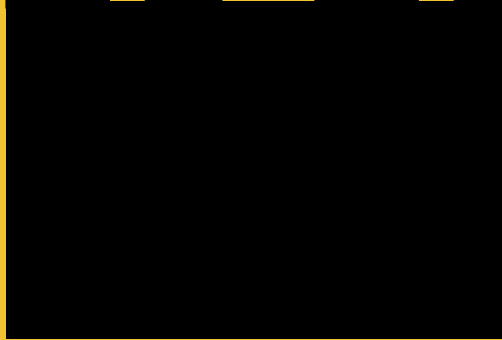
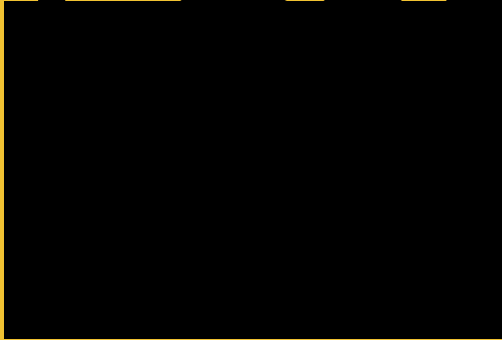
```
var Emergencias = (function(mod) {  
  
    function llamar911() {  
        window.location.href = "http://911.com";  
    }  
  
    mod.llamar911 = llamar911;  
    return mod;  
  
})(Emergencias || {});
```

Suficiente por ahora...

Diseño Modular

POTENCIA PARA
APLICACIONES EN JAVASCRIPT

¿ QUÉ ES ESTO?



Así debería verse tu aplicación web

MÓDULO

MÓDULO

MÓDULO

SANDBOX

SANDBOX

SANDBOX

NUCLEO DE LA APLICACIÓN

jQuery

Backbone

RequireJS

Mediator.js

Otras... más