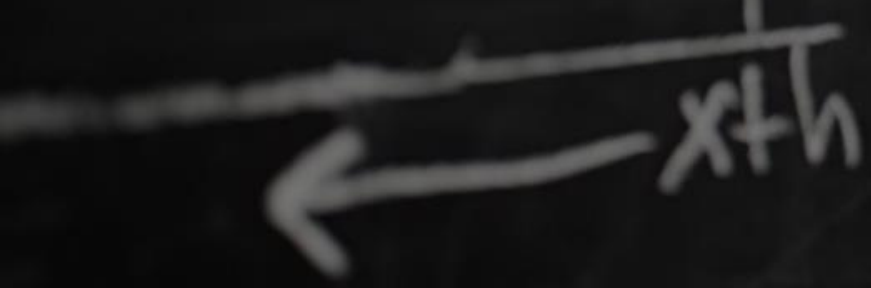
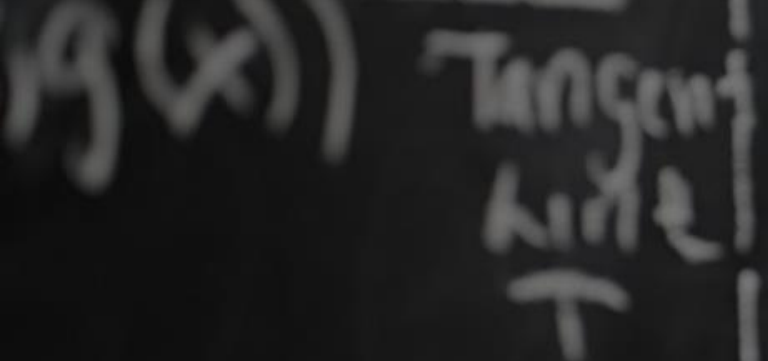


# Functions



$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

$$f(x) = \lim_{h \rightarrow 0} \frac{(x+h)^2 - x^2}{h}$$

$$= \lim_{h \rightarrow 0} \frac{x^2 + 2xh + h^2 - x^2}{h}$$

$$= \lim_{h \rightarrow 0} \frac{2xh + h^2}{h}$$

$$= \lim_{h \rightarrow 0} \frac{h}{h(x+h-x)} = \lim_{h \rightarrow 0} \frac{1}{x+h-x} = \frac{1}{2\sqrt{x}}$$

$$f(x) = \lim_{\Delta x \rightarrow 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$
$$f(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h}$$



Representando  
funciones

# Funciones / 1

```
function a(par1, par2) {  
    return par1 + par2;  
}  
  
console.log(a(1, 1))  
console.log(a(1, "a"))  
console.log(a("1", 4))
```



# Funciones / 2 → Lambda

```
const potencia = (base, exponente) => {  
  let resultado = 1;  
  for (let cuenta = 0; cuenta < exponente; cuenta++) {  
    resultado *= base;  
  }  
  return resultado;  
};  
console.log(potencia(2, 3))
```

```
const cuadrado1 = (x) => { return x * x; };  
const cuadrado2 = x => x * x;  
const bocina = () => {  
  console.log("Meec");  
};
```

¿Qué hace  
este  
programa?

```
function gallina() {  
    return huevo();  
}  
function huevo() {  
    return gallina();  
}  
console.log(gallina() + " vino  
primero.");  
// → ??
```



# Parámetros opcionales

```
function cuadrado(x) { return x * x; }  
console.log(cuadrado(4, true, "camello"));
```

Inconveniente: Pueda llamar a una function con el numero incorrecto de parámetros

Ventaja: Puedo gestionarlo bien... tanto por defecto como por exceso

```
function menos(a, b) {  
    if (b === undefined) return -a;  
    else return a - b;  
}
```

```
console.log(menos(10));  
// → -10  
console.log(menos(10, 5));  
// → 5
```

# Funciones /5: Resto de Argumentos

- `function` sumar(...args) {  
 `let` suma = 0;  
 `for` (`let` arg of args)  
 suma += arg;  
 `return` suma;  
}

`let` x = sumar(4, 9, 16, 25, 29, 100,  
66, 77);

- x = sumTodo(1, 123, 500, 115, 44, 88);

```
function sumTodo() {  
  let suma = 0;  
  for (let i = 0; i < arguments.length; i++) {  
    suma += arguments[i];  
  }  
  return suma;  
}
```

# Funciones /3: Valores por defecto

```
function potencia(base, exponente = 2) {  
    let resultado = 1;  
    for (let cuenta = 0; cuenta < exponente;  
cuenta++) {  
        resultado *= base;  
    }  
    return resultado;  
}
```

```
console.log(potencia(4));  
// → 16  
console.log(potencia(2, 6));  
// → 64
```





# Funciones /4: La clausura (Closure): funciones como valores

```
function valorEncapsulado(n) {  
    let local = n;  
    return () => local;  
}  
let valorUno = valorEncapsulado(1);  
let valorDos = valorEncapsulado(2);  
console.log(valorUno());  
console.log(valorDos());
```


```
function multiplicador(factor) {  
    return numero => numero * factor;  
}  
let doble = multiplicador(2);  
console.log(doble(5));
```

# Ejercicios 1 y 2


1. Escribe una función que calcule si un numero es par o impar
2. Contando elementos:

El comando `"String"[n]` te da el 'n'-esimo carácter de un string. (Ojo, el primero es el 0, el ultimo en `string.length - 1`. Un string de dos caracteres tiene longitud 2 y sus elementos están en las posiciones 0 y 1).

1. Escribe la función `contarBs` que recibe un string como parámetro y devuelve el número de "B"s que contiene.
2. Escribe ahora la función `contarCaracteres` que funciona igual que `contarBs`, pero tiene un Segundo parámetro que indica el carácter que buscamos. Reescribe `contarBs` para que la use



## Funciones /6: Funciones recursivas



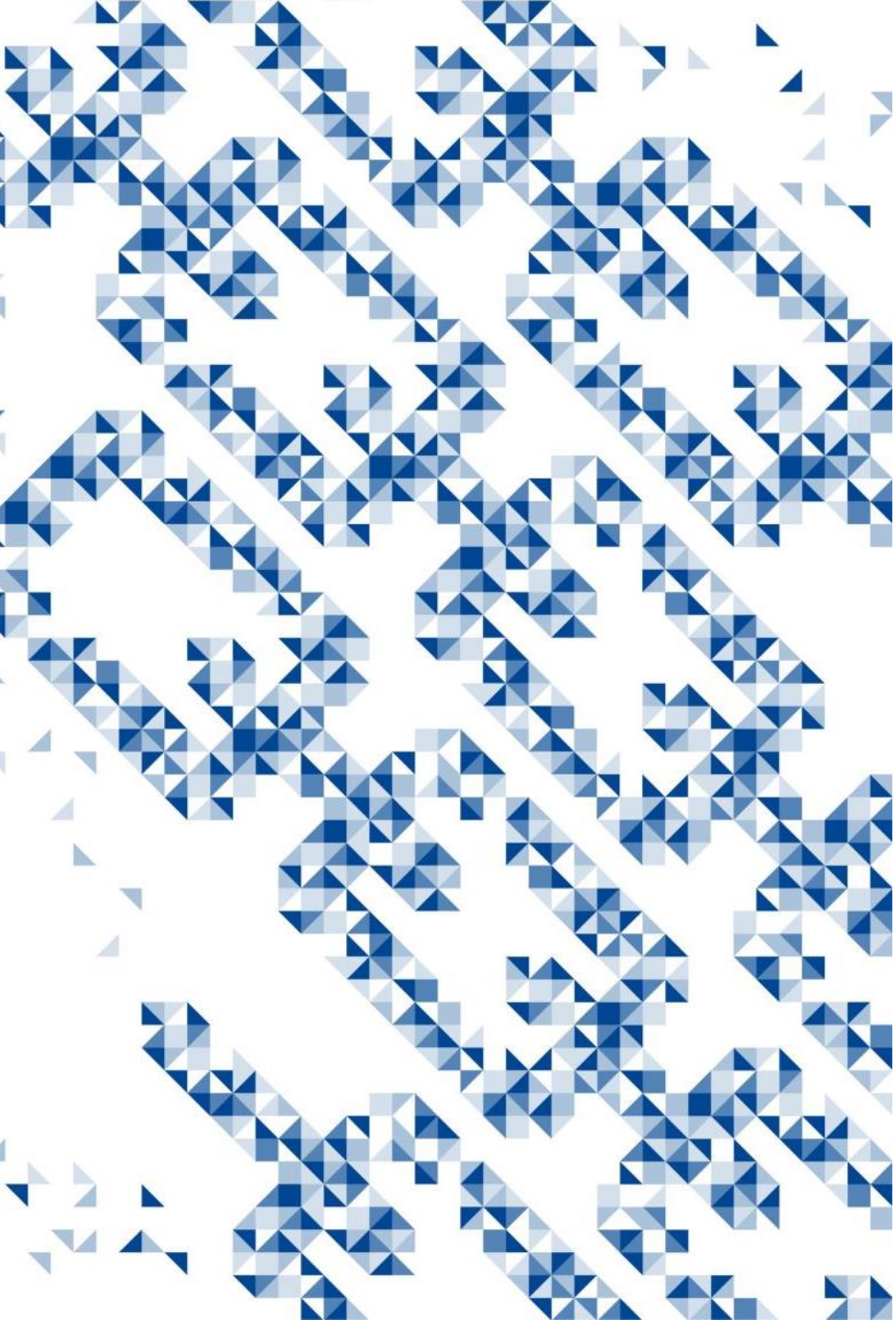
```
function potencia(base, exponente) {  
  if (exponente == 0) {  
    return 1;  
  } else {  
    return base * potencia(base,  
exponente - 1);  
  }  
}  
console.log(potencia(2, 3));
```

## Ejercicio 3 y 4:

3. Imprime por la consola un tablero de ajedrez, las blancas son espacios, las negras asteriscos

4. Evalúa de forma recursiva si un numero es par

- Cero es par.
- Uno es impar
- N es
- n es par si  $(n-2)$  es par



## Ejercicio 5: Resolvamos juntos un puzzle apto para funciones recursivas

---

- Empezando desde el numero 1 y sumando 5 o multiplicando por 3 tantas veces como queramos se genera una serie de números infinita.
- ¿Cómo podríamos escribir una función que si le damos un número nos diga si se puede producir de esta manera?



# Plantilla de literales

---

- Ya lo tenemos: podríamos haber utilizado plantillas de literales para los strings:
  - Usamos `"` o `'` para delimitar un string
  - Pero también se puede usar un: ```
  - Un string definido con este símbolo sustituye los valores de las plantillas `${}` que tiene embebidas ANTES de efectuar la asignación:
  - P.ej: ``la mitad de 100 es ${100 / 2}``

# ¿Cómo se pasan los parámetros?

- Los argumentos se pasan por valor
- La función solo conoce el valor, no su ubicación en memoria
- Si la función cambia el valor del argumento, no se cambia el valor original
- Los objetos se pasan por referencia
- Si una función cambia alguna propiedad de un objeto el valor original quedará cambiado

"11 Paso Parámetros Valor Referencia.html"

# Funciones y los efectos laterales

Una función bien escrita del exterior solo usa sus parámetros y al acabar devuelve un valor

No modifica valores externos visibles a ella en variables globales

No imprime cosas porque si





# Entregable 1: Haz en **60'** la función potencia() que si....

---

- Recibe un parámetro (x) devuelve el número al cuadrado,  $x*x$
- Recibe dos parámetros (x,y), devuelve  $x^y$
- Recibe tres parámetros (x,y,z) devuelve  $x^{y^z}$
- Recibe cuatro parámetros te devuelve la función (w,x,y,z) ->  
 $\{ x + \text{imprime literal } y + w^x + \text{imprime literal } z \}$  (pero en JS 😊)
- Define una constante cuadrado(x) que utilizando la ultima definición imprima algo así como:
  - $x + \text{"al cuadrado es " + } x*x + \text{" y es par "}$  si el resultado es par,
  - y lo equivalente si el resultado es impar
  - Ayuda: El cuadrado de un número par es par, y el de un número impar es impar

ARRIBA!  
ARRIBA!  
ANDALE!  
ANDALE!


¿Quieres mas? ¡Ándale!

Function

INPUT x

FUNCT

OR

 Tutorials ▾ References ▾ Exercises ▾ Sign Up

HTML CSS **JAVASCRIPT** SQL PYTHON JAVA PHP BOOTSTRAP HOW TO W3.C

JS Tutorial  
JS HOME  
JS Introduction  
JS Where To  
JS Output  
JS Statements  
JS Syntax  
JS Comments  
JS Variables  
JS Let  
JS Const  
JS Operators  
JS Arithmetic  
JS Assignment  
JS Data Types  
**JS Functions**

Ad served by Google

Ad options Send feedback Why this ad?

## JavaScript Functions

← Previous

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

Example