

#### ❑ **ACTIVITY 1 -- RUN CODE IN CONSOLE FROM EXTERNAL FILE**

- Create an empty web page that loads a JavaScript file called **greeting.js**
- Write JavaScript code in the **greeting.js** file that allows you to type **hello** on the console.

#### ❑ **ACTIVITY 2 – Run code in node**

- Create a file called `greeting.js` that writes Hello on the console.
- Execute the code using node.

#### ❑ **ACTIVITY 3 – Reading numbers**

- Create a web page that asks the user for a number through a reading box.
- A message box indicates if what the user typed is really a number or not.

#### ❑ **ACTIVITY 4 – Random background**

- Create a web page that displays a random background color every time we enter it.

#### ❑ **ACTIVITY 5 – Salary calculation**

- Create a web application that asks, in two data reading boxes: the name, surname, salary (number with decimals) and age of a person (a number).
- We will assume that the user enters the data correctly (we will not validate it).
- The page will indicate the written name and surname, age and salary (once recalculated with what the following points indicate)
  - ✓ If the salary is greater than 2000 euros, it will not change
  - ✓ If the salary is between 1000 and 2000:
    - If in addition, the age is greater than 45 years, it is raised by 3%
    - If the age is less than 45 or equal, it is raised by 10%
  - ✓ If the indicated salary is less than 1000
    - The salary for people under 30 years will be exactly 1100 euros
    - If the age is between 30 and 45 years, the salary goes up by 3%
    - For those over 45, it rises by 15%

#### ❑ **ACTIVITY 6 – Number guessing game**

- Make a web page that implements a game of finding a random number under the premises explained below:
  - The page will calculate a number from 1 to 100
  - It will then ask the user for the number
  - If the user types something that is not a number, the error is indicated and the number is requested again.
  - If the number entered by the user is correct, it is indicated that it was correct and we will finish the game

- If not, it tells you if the number is smaller or higher and asks again what it is
- If any box is canceled, the game ends up indicating that the game was canceled.
- In the end if the game has been successfully completed the number of attempts is indicated
- The user is allowed to play again using a confirmation box

#### ☐ **ACTIVITY 7 – Triangle of asterisks**

- Create a Web application that asks the user for a positive integer
  - If what the user types is not a number or it is not positive, the page will not display anything. It will go blank.
  - The application will write a triangle with as many asterisks as the user number indicates.
  - To make the effect more effective we will use, to write the asterisks, a monospaced letter.
- EX:

```
*
* *
* * *
* * * *
* * * * *
```

#### **(CHOOSE AT LEAST 2 FROM THE FOLLOWING ACTIVITIES)**

#### ☐ **ACTIVITY 8 – Other asterisk triangles**

- Do a practice similar to the previous one but get other triangles
- Triangle 1

```
* * * * *
* * * *
* * *
* *
*
```

- Triangle 2

```
      *
     * *
    * * *
   * * * *
  * * * * *
```

- Triangle 3

```
* * * * *
 * * * *
  * * *
   * *
    *
```

### ❑ ACTIVITY 9 – Factorial

- Create a web page that asks the user for a positive integer and calculates its factorial
- The factorial of 5 (usually denoted as 5!) is the result of multiplying  $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5$ , that is: 120

### ❑ ACTIVITY10 – Unicode Code Table

- Create a web page that displays the first 10000 symbols in the Unicode table
- It will be displayed in a table in which each row indicates the code number, followed by the character of that code.
- Each row will display 10 symbols
- Example of some rows:

Tabla Unicode

1	□	2	□	3	□	4	□	5	□	6	□	7	□	8	□	9	□	10	□
11	□	12	□	13	□	14	□	15	□	16	□	17	□	18	□	19	□	20	□
21	□	22	□	23	□	24	□	25	□	26	□	27	□	28	□	29	□	30	□
31	□	32	□	33	□	34	□	35	□	36	□	37	□	38	□	39	□	40	□
41	)	42	*	43	+	44	-	45	=	46	.	47	/	48	0	49	1	50	2
51	3	52	4	53	5	54	6	55	7	56	8	57	9	58	:	59	:	60	<
61	=	62	>	63	?	64	@	65	A	66	B	67	C	68	D	69	E	70	F
71	G	72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O	80	P
81	Q	82	R	83	S	84	T	85	U	86	V	87	W	88	X	89	Y	90	Z
91	[	92	\	93	]	94	^	95	_	96	`	97	a	98	b	99	c	100	d
101	e	102	f	103	g	104	h	105	i	106	j	107	k	108	l	109	m	110	n
111	o	112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w	120	x
121	y	122	z	123	{	124		125	}	126	~	127	DEL	128	€	129	□	130	□
131	f	132	~	133	...	134	†	135	‡	136	•	137	‰	138	§	139	◊	140	GE
141	□	142	□	143	□	144	□	145	□	146	□	147	□	148	□	149	□	150	□
151	—	152	—	153	™	154	®	155	®	156	®	157	□	158	®	159	™	160	—
161	□	162	g	163	£	164	□	165	¥	166	□	167	g	168	—	169	©	170	*
171	α	172	—	173	—	174	®	175	—	176	α	177	±	178	—	179	α	180	—
181	μ	182	¶	183	—	184	—	185	—	186	—	187	—	188	¼	189	½	190	¾
191	ü	192	À	193	Á	194	Â	195	Ã	196	Ä	197	Å	198	Æ	199	Ç	200	È
201	É	202	Ê	203	Ë	204	Ì	205	Í	206	Î	207	Ï	208	Ð	209	Ñ	210	Ò

### ❑ ACTIVITY11 – Random Boxes

- Create a web application that displays 2000 squares of random color of 50 pixels.
- The position on the screen will also be random
- Example of result:

