EJERCICIOS DE CONVERSIÓN/COERCIÓN

null

1. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a) `null.toString()` b) `null?.toString()` c) `typeof null`	d) `if (!null) console.log("hola");`

Undefined

2. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a) `undefined.toString()` b) `undefined?.toString()` c) `typeof undefined`	d) `if (!undefined) console.log("hola");;`

Boolean

3. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a) `let a1 = !!(35);` b) `let a2 = !!(0);` c) `let a3 = !!(-0);` d) `let a4 = !!(NaN);` e) `Boolean(undefined)` f) `Boolean(!undefined)`	g) `Boolean(35)` h) `Boolean(0)` i) `Boolean(+0)` j) `Boolean(-0)` k) `Boolean(NaN) == false` l) `!!""`	m) `!""` n) `"" == true` o) `[] == false` p) `+true + 20` q) `true + 20` r) `-true + 20`

Number

4. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a) `let n1 = NaN;` `n1 !== n1` b) `+"34"` c) `12_234 + 1` d) `+"12_234" + 1`	e) `~~(-0)` f) `-0 << 2 >> 2` g) `Number({})`

BigInt

5. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a)
```
Number(9007199254740992n) + 1;
```
(el número es `Number.MAX_SAFE_INTEGER`)

b)
```
Number(9007199254740992n + 4000n)
```

c)
```
let a = BigInt(true);
let b = BigInt(false);
let c = a + b;
```

d)
```
let a = BigIntr
```

e)
```
BigInt(undefined);
```

f)
```
BigInt("");
```

Strings

6. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a)
```
"23" + 2
+"23" + 2;
```

b)
```
"23" - 2
2 - "23"
```

c)
```
+"23.4"
```

d)
```
-"23.4";
```

e)
```
+"23443443" + 2344n;
```

f)
```
BigInt("23443443") + 2344n;
```

Object

7. Dadas las siguiente expresiónes en JavaScript, para cada una aplicando las reglas de orden de agrupación (precedencia y asociatividad) y orden de evaluación:
   - pon paréntesis de forma que, sin modificar su funcionalidad, haga claro en qué orden se evalúa.
   - indica el valor resultante de la expresión y de las variables, si las hay

a)
```
if ({}) console.log("hola");
```

b)
```
if (!{})
    console.log("hola");
  else
    console.log("adios");
```

c)
```
{} + 23;
```

d)
```
23 + {};
```

e)
```
23 + +{};
```

f)
```
+{} + 23;
```

g)
```
23 - {};
```

h)
```
{} - 23;
```

i)
```
{} + "hola"
```

j)
```
"hola" + {}
```

k)
```
{} + 23 - "23"
```

l)
```
{} + 23 + "23"
```

m)
```
-"23" + 23 + {}
```

n)
```
{} + null
```

o)
```
null + {}
```

p)
```
null - {}
```

q)
```
{} - null
```