Blank page ..

Pretty , isn’t it? 

Why Booleans ?
Because it’s .. logical

- ➔ One of the core concepts of programming - decisions if-else.
- ➔ Javascript gives us some really cool opportunities to get stuff done with them, like: conditional rendering, truthy/falsy, casting, ternary operators, skipping var initializing
- ➔ I was struggling with them as well, in the beginning.
- ➔ Thanks for the suggestion Michael 👍

How is this gonna go ?

- ➔ The truth tables of boolean operators
- ➔ Types of booleans in JavaScript
- ➔ Boolean expressions
- ➔ Cool stuff
- ➔ Some links
- ➔ Some code

Logic/boolean Truth tables

True *vs* **False** is like **1** *vs* **0** or like **On** *vs* **Off** -> it's a binary state (**something-true** vs **something-false**)

Truth Tables for `boolean` Operators

&& (and)

Value of A	Value of B	A && B
true	true	**true**
true	false	**false**
false	true	**false**
false	false	**false**

|| (or)

Value of A	Value of B	A || B
true	true	**true**
true	false	**true**
false	true	**true**
false	false	**false**

! (not)

Value of A	!A
true	**false**
false	**true**

AND Truth Table

Inputs		Output
A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Table

Inputs		Output
A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

XOR Truth Table

Inputs		Output
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

NAND Truth Table

Inputs		Output
A	B	$Y = \overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

Boolean/logic.. things

- The actual values:
  true is true
  false is false

- An expression between boolean values:
  (true || false) is true
  (false && true) is false
  !true is false

- An expression that evaluates to boolean
  (1 == 2) is false
  (2 == 3-1) is true
  (4==1 ? true : false) is false

- Casted values (!!) - anything can be boolean
  !! "text" is true
  !! 0 is false

- Truthy/Falsy things - everything is boolean (within context)
  zero, '', null, undefined, NaN are falsy
  anything else that has relevant value are truthy
  Empty objects **{}** or arrays **[]** are considered truthy
- Truthy/Falsy is interpreted where a logical value is expected
  - also called *implicit conversion* to boolean
  - in if-else structures:
    if(**expr**) doSomething;
  - in logical expressions:
    falseResult = **("Ana" && false)** –> **true && false** = false
    trueResult = **("Joe"?true:false)** –> **true?true:false** = true
    falseResult = **!"Alabama"** –> **!true** = false

Truthy/Falsy is powerful stuff, but *don't forget about* it, or the debugging is gonna be ***painful*** !!

Truthy/Falsy saves time and lines of code

```
if( userName.length() > 0 )      -> if( userName )
if( myList.length > 0 )          -> if( myList.length )

let person = { name: 'Jack', age: 34 };
console.log(person.job || 'unemployed'); // 'unemployed'
```

How do boolean expressions work?

- ➔ They are evaluated from left to right until is not necessary to do it anymore, that is, if the result of the left side is enough, then the interpreter doesn't bother evaluating the rest of the expression

- ➔ OR expressions get evaluated until one is true
  exprA || exprB || exprC –> if **exprA** is true, there is no point in evaluating the rest, as **true || anything** is true

- ➔ AND expressions get evaluated until one is false
  exprA && exprB && exprC –> if **exprA** is false, the others don't get evaluated as **false && anything** is false

- ➔ This expression:
  result = **exprA && exprB || exprC || exprD && exprE && exprF**

  -> if **exprA** if false then the others don't get evaluated because **false && *anything*** is false
  -> if **exprA && exprB** are both true, then the others don't get evaluated, **true || *anything*** is true
  -> in order for it all to be evaluated it should have these values:
  exprA = true, **exprB** = false, **exprC** = false, **exprD** = true, **exprE** = true, **exprF** = true/false

Cool stuff you can do with it

- Test if an object exists before running it's method - this trick it's used literally everywhere
  mox.doSomething() this throws an error if mox doesn't exist
  mox && mox.doSomething() this does not, because if mox doesn't exist, it's falsy, then the expression is false and there's no point in evaluating the right side, therefore **mox.doSomething()** won't throw an error

- Test if a variable (any variable) is has value before using it
  result = 5 + val throws an error if val is undefined
  result = 5 + val || 0 it will work, regardless if val is defined or not, because **val** in **val || 0** is interpreted as truthy and if it has a truthy value it's considered true and that value is returned, if it's considered false, then the **0** from the right side of **||** is returned.

- Conditional rendering in react

```
return (
        (arrayItems.length == 0) && <p>the list is empty</p>
)
```

Some useful links

https://developer.mozilla.org/en-US/docs/Glossary/Truthy
https://developer.mozilla.org/en-US/docs/Glossary/Falsy
https://www.freecodecamp.org/news/how-to-convert-value-to-boolean-javascript/
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Nullish_coalescing_operator

https://codeburst.io/javascript-showdown-vs-7be792be15b5
https://codeburst.io/javascript-what-is-short-circuit-evaluation-ff22b2f5608c → good article as well
https://www.youtube.com/watch?v=-u260xZ9e4M → a good video discussing booleans

Some code in this gist:
https://gist.github.com/bar-alex/c4665de4be12f2a05650cca6bb5cb8f2