Hablemos de tipos Javascript, es un lenguaje débilmente equipado a diferencia de otros, donde hay que declarar específicamente y de manera estricta qué tipo de datos va a vivir, va a existir en una variable. En Javascript eso es dinámico, vamos a ver algunos conceptos que nos ayuden a entender esta cuestión, que es una cuestión importante en el mundo de Javascript.
Hablemos de tipos, lo cual es un tema un tanto polémico en Javascript, porque Javascript es un lenguaje débilmente tipado, es decir, las variables o los datos pueden tener un tipo, pero el tipo no es fijo, no es estar estático, es dinámico a diferencia de otros lenguajes.
Esto no es malo, ni bueno en sí mismo, puede ser bueno para algunas cosas pero para otras nos somete a posibles errores, tenemos que ser muy cuidadosos con el manejo del tipo de dato en Javascript.
Okey, veamos lo siguiente, supongamos que tenemos una variable que se llama “nombre” y el nombre es, tengo que poner comillas aquí obligatoriamente, porque voy a escribir un "string", una, un texto vamos a decir que el nombre es “Marcelo Eduardo”, por ejemplo, ¿okey?. Bien y aquí vamos a poner “let”, edad, igual120, es una buena aspiración, podríamos poner más o menos, no importa, es un detalle. A quí tenemos un tipo de datos "string", o sea un texto, aquí tenemos un “number”, podríamos también tener un tipo “booleano”, un tipo “booleano”, un tipo “boolean”, ¿si? “boolean” o “booleano”, es un valor, es un tipo que sólo admite dos posibilidades "true", “false”, verdadero o falso, ¿si?, podríamos poner por ejemplo todo el “let” “vive” es igual a "true" o podría ser “false”, lo que sea, ¿okey?, aquí cualquier otro dato está bien, simplemente para mostrar posibilidades.
Bueno, estos son algunos de los datos, de los tipos de datos que podemos manejar en Javascript, de hecho, permítanme decirles, antes de continuar, que en Javascript tenemos los datos primitivos, que son siete, si no me equivoco, son siete "string", “number”, “bigint”, que es para números muy grandes, perdón que lo estoy simplificando un poco, pero básicamente es así, “boolean” para valores verdadero o falso, luego tenemos “undefined”, es el valor que Javascript le pone por defecto a aquellas variables que no hemos inicializado, como vimos hace instantes, y luego, también tenemos “null”, el valor nulo, y “symbol”, okey, todo lo demás en Javascript, todo lo demás es un objeto, ¿okey?, todo lo demás es un objeto, ya veremos las diferencias, básicamente.
Ahora, ¿cómo podemos saber de qué tipo es?, en caso de que quisiéramos averiguarlo, de qué tipo es una, una variable, bueno, a ver “let” indefinida, aquí no vamos a poner ningún valor, aquí podríamos indicar “let” nula, igual “null”, bueno y así sucesivamente, tenemos una función incluida también en Javascript que se llama “type of” y es muy útil cuando queremos averiguar de qué tipo es una variable. Por ejemplo, podríamos hacer “console log”, “type of” y luego el nombre de la variable que queremos someter a exámen. Comencemos por nombre y deberíamos ver "string", aquí "string", preguntemos por edad y va a decir “number”, el tipo de esa variable es “number”. ¿Qué ocurre con “vive”? Nos dirá “boolean”, “booleano”, esto indica que solamente puede ser verdadero o falso, ¿podríamos preguntar el valor de “indefinida”?, y sí, adivinaron “undefined”, porque no le hemos puesto ningún valor. Javascript por nosotros la inicializa a esta variable con un valor de “undefined”. Por el contrario, si preguntamos por el valor de la variable “nula”, nos dirá que es un objeto. Este es un caso muy especial de Javascript, hay algo, hay algunos como éste, ¿okey? “null”, esta variable al asignarle el valor “null”, se convierte al tipo objeto, por más que les hayamos dicho, lo cual es

técnicamente cierto, que uno de los tipos primitivos es “null”, cuando preguntamos por el tipo de “null” nos dice que es un objeto. Podríamos ver esto como un error en el lenguaje, sí, podríamos verlo como un error en el lenguaje, pero bueno, es un clásico de Javascript preguntar por el tipo de un objeto nulo y averiguar que en realidad es, perdón, por una variable de tipo, con un valor nulo, y averiguar que no es un primitivo como, no sé, como nos dice la documentación, sino, que es un objeto.
Bueno, iríamos aquí a la cuestión del huevo y la gallina y no es nuestro caso, por lo menos no en este momento. Perfecto.
Ahora, veamos lo siguiente, podemos hacer operaciones, sí, claro, podemos hacer operaciones, por ejemplo, de tipo matemático. Suponemos que tenemos lo siguiente: una variable “let”, “number one”, igual 200 y “number two”, igual, uno. Okey, muy bien. ¿Cómo hacemos para sumar esto? Básicamente, es muy sencillo con los mismos operadores, los mismos signos que utilizamos en la notación matemática, por ejemplo, podríamos crear una nueva variable, digamos, “let resultado” que podría ser básicamente, perdón la redundancia, el resultado de una operación sobre estas dos variables, por ejemplo, podríamos decir que el resultado es igual a la resta de “number one” y “number two”.
Bien y ahora aquí podríamos hacer un “console log” de resultado, “console log” de resultado y aquí vemos 199, bien y, ¿qué pasa si damos vuelta estos términos? Bueno, pues, nos daría un número negativo, ¿verdad? ahora, cuando escribiesemos el resultado menos 199, obviamente, si a uno le restamos 200, okey, es un número negativo el resultado. Recuerden que cuando decimos que Javascript no necesita ser descargado, porque es un lenguaje interpretado y que ya vive en nuestros navegadores, esto es tan cierto como preguntar aquí mismo, aquí adentro, por el valor de una variable, ¿cuál es el valor de number uno?, 200. Okay, ¿cuál es el valor de number dos?, uno, y aquí mismo podría ser una operación, por ejemplo, number uno, igual nueve, ¿si? Y ahora number uno, más, number dos, nos daría otro resultado, porque number uno ya no es 200. Aquí acabo de asignarlo, simplemente quería mostrarles que ustedes pueden trabajar directamente aquí, en la consola, no lo haremos habitualmente, pero quería mostrarles que es posible.
Muy bien, por supuesto podemos hacer otro tipo de operaciones, multiplicaciones, restas, sumar, restar, al mismo tiempo hacer operaciones complejas, bien, lo que ustedes quieran, ¿si? Solamente es cuestión de probar con más variables y ver qué ocurre cuando empiezan a anidar y a concatenar otras expresiones matemáticas.
Muy bienm vamos a ver entonces qué ocurre más allá de estas operaciones básicas y esto es lo que nos permite hacer Javascript y sí es un problema cuando queremos, por ejemplo, sumar variables que son de distinto tipo. En otros lenguajes no podríamos hacerlo pero aquí, en Javascript sí, porque es un lenguaje muy permisivo en ese sentido, entonces, supongan que tenemos el dato uno y dato dos, dato uno es, bueno, 20 y dato dos es nuestro perro Firulais, vaya nombrecito, ¿no?
Okey, bien, y ahora podríamos hacer, por ejemplo, aquí tenemos un error en consola, porque estoy aquí, hice una suma de dos variables que ya no existen, okey, por eso nos dice “number two is not defined”, ¿okey? Obvio, no está definida y número uno, “number one” tampoco, pero el error rompe cuando llega aquí y nos sigue evaluando el resto, ¿entienden?, voy a borrar esta línea y ya no tendremos problemas.
Y ahora vamos a hacer una operación, digamos que “resultado”, ésta podría definirla con “const” también, porque no va a cambiar de manera directa, por lo tanto sería igual, “resultado” será igual a dato uno, más, dato dos. Y aquí tengo un problema, estoy sumando un número a una, a un “string”, a un “number” y un "string". Bien, Javascript va a tomarse la

libertad de interpretar esto. Cuando quiera sumarle un número a un "string", Javascript va a ser lo siguiente: va a interpretar este primer término como si fuera un "string" y el resultado, tal vez les sorprenda, porque no será un error, el resultado es 20 Firulais y si preguntamos por el “type of” “resultado”, nos dirá que es de tipo "string", porque este 20, que es un número, al ser sumado a un "string", se concatena como si fuera texto, por lo tanto, esta variable es de tipo "string".
Muy bien, pero ¿qué ocurre si hacemos lo siguiente?, ¿qué ocurre si hacemos dato uno? Vamos a hacer dato dos, 10 y aquí a Firulaís le pondremos que es el dato tres. ¿Qué ocurre si hacemos dato uno, más dato dos, más dato tres? Y pedimos por pantalla el resultado, oops, ¿que pasó aquí?, bien como las operaciones matemáticas van en este sentido, tengo dos números, ¿puede sumarlos Javascript?, los suma y ya no tenemos 20, tenemos 30, pero cuando tenemos 30, se encuentra con que debe sumar otro aquí, otro factor aquí, y esto es un "string", por lo tanto ahí, sí comienza a concatenar, ¿okey, se entiende? Entonces si sumamos un solo número con un "string", el número se convierte en "string", pero si tenemos más de una operación con números, tratará de resolver, Javascript, tratará de resolver tantas como pueda cuando ya no pueda resolverlas, porque se encuentre con que queremos sumar una número con un texto, ahí sí ya convertirá todo a texto, es decir a "string".
Bien, perfecto, continuamos entonces que ya tenemos estas cuestiones resueltas, por lo menos tratamos de entenderla,s hay que tener mucho cuidado entonces, en ese sentido, porque Javascript no me lee "string" los tipos, los tipos de datos. Además puedo cambiar el tipo de datos sobre la marcha y eso puede ser un problema si no estoy muy atento, muy atenta. Por ejemplo, puedo tener una variable que se llame “dato” y que tenga el valor de “manzana” y luego a “manzana” puedo asignarle “100” y esto no me va a dar ningún tipo de error, por ejemplo, aquí haremos un”console log” de datos y si lo repetimos aquí abajo, okey, veremos, “dato”, perdón aquí puse “manzana” y es “dato”, ahí está primero es manzana y es de tipo "string" y luego es 100 y este tipo “number”, es más, luego puedo decir que “dato” es igual a "two" y esto no me va a dar un error y lo voy a mostrar y ahora se convirtió en un tipo “booleano”.
Bueno, esto no puedo hacerlo en otros lenguajes y en Javascript sí puedo. Algunos creen que esto es, bueno, quienes vienen de otros lenguajes creen que esto es sumamente malo, porque cuesta acostumbrarse a este al nivel de control que hay que tener para que Javascript no comience a hacer cosas inesperadas.
Bien sabido esto, vamos a pasar a nuestra próxima clase.