**JAVASCRIPT**

|  |  |  |
| --- | --- | --- |
| <script>  <script/> | Desde el html podemos realizar código javascript. Con estas etiquetas.  Pude in en cualquier parte y se ejecutara dependiendo de la posición en la qu ese encuentre. | |
| **Salida** | | |
| <script src=”doc\_javascript.js”><script> | Llamar a javascript desde un archivo externo. | |
| alert(“hola mundo”); | Mensajes con alert:   * Detienen la ejecución del programa * Se ejecuta en primer plano, no deja acceder al resto del programa | |
| document.write(“hola mundo”); | Escribe dentro documento html lo que hay dentro de la | |
| console.log(“hola mundo”); | Muestra información en consola. | |
| document.getElementById(“id\_caja”).value=nombre | Muestra el valor de la variable en la caja de texto html.  NOTA: es solo una de sus | |
| **Entrada** | | |
| nombre=prompt(“Ingrese nombre”, “mensaje guía”); | Mensaje emergente de entrada | |
| nombre=document.getElementById(“id\_caja”).value; | Obtenemos el valor de la caja de texto html y la asignamos en la variabe. | |
| **VARIABLES** | | |
| **let** nombre, edad, apellido  **var** nombre;  var nombre, edad, apellido;  var nombre=”Jonathan”, apellido=”Alanoca”;  var num=num2=num3=0;  edad=12;  apellido=”Alanoca”;  var variable01=apeliido + edad;  var variable01=[“jonathan”, 12, true]; | Las variables cumplen las siguientes reglas:   * Keysensitive: diferencian entre mayúsculas y minúsculas . * Solo pueden tener $ o \_ como símbolos * No deben usar las palabras reservadas del leguaje. * Se puede definir tanto con **var o con let.** | |
| const var nombre=”adf” | Una vez intanciado no se puede modificar | |
| var num+=1  num++;  num-=-1  num—;  num-=;  nun\*=; | Acumulador o concatenación en caso de se string. | |
| Number  String  Boolean  Arrays | Para ver el tipo de variable:  **typeof(nombre\_variable);**  **“string”,”number”, “boolean” o “array”** | |
| NUMBERS |  | |
|  |  | |
|  |  | |
| STRINGS | “jonathan Alanoca” | |
| nombre.length; | Devuelve número de caracteres de string 1-> | |
| number(var\_numerostr);  parseInt(numero, 10);  parseFloat(“1231.12”); | Convierte a un número (entero o decimal). En caso de no poder devuelve null.  Convierte el contenido de la variable en entero decimal.  Podemos especificar en qué sistema numérico se encuentra la varible 2-8-16 etc. Por defecto viene 10(decimal)  convierte a decimal | |
| nombre.indexOf(“Alanoca”); | Devuelve la posición que se encuentre la cadena 0-> | |
| nombre.lastIndexOf(“o”); | Devueve la ultima vez que se encuentra esa cadena o valor. | |
| Nombre.search(“Alanoca”); | Devuelve la posición que se encuentre la cadena 0-> | |
| nombre.charAt(1); | Devulve el carácter que se encuentra en esa posición. | |
| Nombre,repeat(2); | Repite la cadena n veces | |
| nombre.replace(“str\_busqueda”,”str\_reemplazar”) | Reemplazar una cadena. | |
| Nombre.slice(1,4); | Devulve los caracteres que se encuentran en ese rango index | |
| nombre.substr(2,2); | Devuelve una cadena (iniciando desde un index, numero de caracteres a tomar.) | |
| nombre.split(“-”); | Corta la cadena en cada carácter definido. El resutado lo convierte en array. | |
| Nombre.toLowerCase();  nombre.toUpperCase();  nombre.trim() | Cadena en minúscula.  Cadena en mayúscula.  Borra los espacios del inicio y final de nuestra cadena. | |
| Nombre.endsWith.(“noca”); | True o false si se cumple o no a condición | |
| nombre.includes(“jona”); | T o F si esa cadena se encuentra en nuestra variable. | |
| IsNaN(“adsf”);  ¡isNaN(“12.2”); | true si no es número.  false si es número. | |
| **NUMBERS** |  | |
|  |  | |
| Math.random()  Math.random()\*100 | Numero random de 0-1  Entre el 0-100 | |
| Math.floor(var) | Redondea a la baja . devuelve **Entero**un integer number | |
| toFixed(2) | Limitar la cantidad de decimales x.01 **Devuelve en String** | |
| If(isNaN(num)&&num!=null&&num!=””){}  Num=prompt(“ingrese”);  Num=num.trim();  While(isNaN(num)||num==null||num=””){} | Validar un número  Validar con while  Borra espacios al inicio y final | |
| **Arrays** | var nom1=[“José”,”Carlos”], nom2=[“Alan”,”Roberto”] | |
| nombrestotales.length; | Cantidad de variables que hay dentro de nuestro array.  1->++ | |
| nombrestotales=nom1.concat(nom2); | Junta dos arrays y la asignamos a una variable. | |
| Nombrestotales.includes(“jonathan”);  Nombrestotales.includes(“jonathan”,3); | true o false  busca a partir de ese index | |
| Nombrestotales.indexOf(“Carlos”); | Devuelve el index de la cadena. Primera en encontrar 0->++  En caso de no encontrar ninguno devuelve -1 | |
| nombres.lastIndexOf(“Carlos”) | Devuelve el index de la ultima de la lista. | |
| Variablestring=nombrestotales.toString() | Convierte el array en cadena separados por “,” estrictament | |
| variablestring=nombrestotales.join(“,”) | Convierte en una variable string y separa las varibles del array con el carácter especificado. | |
| Nombre.pop() | Borra el ultimo elemento del array. **Y devuelve el valor borrado** | |
| nombre.shift() | Borra el primer elemento**. Y devuelve el valor borrado**. | |
| Nombre.unshift(“Carla”); | Añade el principio del array. **Y devuelve su longitud.** | |
| Nombre.push(“Carla”); | Añade al final del array. **Y devuelve su longitud.** | |
| Nombre.reverse(); | Invierte los elementos del array | |
| nombrestotales.slice(2,4) | Toma los valores dentro de rango. | |
| Nombrestotales.sort() | Ordena los valores del array a-z o 0-9 | |
| nombrestotales.splice(1,0,”jony”,”Alberto”) | Bien: (campo1,campo2, campo3)  Campo 1: nos da la posición del array.  Campo2: cuantos valores vamos a borrar luego del posicionamiento.  Campo3: variables o valores que deseamos insertar.  **Devuelve los valores que se borraron.** | |
| CONDICIONALES | | |
| If()-si lo que hay dentro del paréntesis es true se realiza lo del cuerpo del if  If (condicion){  }  else if(*condicion2*){  }  else{  } | Comparadores:  ==  !=  >=  <=  ===  !==  &&  || | Igual (acepta “1”=1)  Deferente  Estricta =valor y tipo  Estrica ¡= valor y tipo.  Y lógico  O lógico |
| var condición=true;  If(condicion){}  If(!condicion){} | **If(…) detecta si la “condición” no es 0 (false).**  **Todo lo demás lo acepta como 1 (true).**  If(true) es una condición aceptable para el condicional.  If (varible) es una condición aceptable para el condicional.  If(“hola”) es una condición aceptable para el condicional.  If(1234123) es una condición aceptable para el condicional.  If(false) no es una condición aceptable y lo saltea.  If(0) no es una condición aceptable y lo saltea. | |
| var num=1;  num>0?alert(“aca se cumple”:alert(“aca no cumple”);  var respuesta=num>0?”si”;”no”;  var respesta =num>3?”primer intento”:  num>2?”primer intento”:  num>1?”primer intento”:  num>0?”primer intento”:  “se acabaron los intentos;  Alert(respuesta) | **TERNARIOS**  Condición?verdadero:falso;  Se puede asignar el resultado del condicional a una variable.  **Similar al if**  **If**  **If**  **If**  **else** | |
| var i=prompt("ingrese una fruta");      switch(i){          case "naranja":              alert("elijio naranja");              break;          case "pera":              alert("elijio pera");              break;          case "fresa":              alert("elijio pera");              break;          default:              alert("no esta en la lista");      }  switch(true){  case i>10:  Break;  } | Condicional: agarra el case que cumpla la igualdad,  Es importante el break en cada case para que no ejecute el siguiente y termine con el condicional.  Como poner un condicional en en swith, responde a la siguiente instrucción  True ==(i>10)  True==true | |
| **BUCLES** | | |
| while(true){} | Mientras sea verdadero el contenido… | |
| do alert(“realice nuevamente”);  while(true)  do{  alert(“una vez mas);  i++  }  while(i<4); | Do while siempre se va a realizar una vez minimamente  Mientras se cumpla la condición vuelve a realizar el bucle una vez mas.  Cuando va a haber mas de una línea de instrucción, | |
| while(edad<0||edad>100){}  while(nombre!=”jonathan”&& nombre!=”Gerardo”){}  while(nu1==1&&(num==2||num3=3)){} | Mientras edad este fuera del rango hará el ciclo.  Mientras no sea esos valores  Aclaración: reemplazar por true y false y por lógica tendremos while(true) o while(false)  Agrupar para dar prioridad a las condiciones | |
| for(var i=0, i<5,i++){  }  **for(var i in var\_lista){}**  **for(var i of var\_lista){}** | Declaración, condición, acumulador  Recorre los índices del array.  Recorre los valores del array | |
| break; | Corta el ciclo cuando se encuentre en esta liena | |
| cicloprincipal: while(true){  ciclosecundario: while(true){  **break cicloprincipal;**  }  } | Break “labeled”  Labeled etiqueta a un ciclo para identidicarlo.  Cortamos el ciclo especifico. | |
| continue;  while(true){  ……1  If(condición==true){continue;}  ……2  } | Continue “labeled”;  No deja pasar al código luego del continue  “Corta el cliclo” y reinicia con la siguiente vuelta. | |
| **HTML Y JAVASCRIPT** | | |
| <script type=”text/javascript” src=”ruta.js”></script> | Linkear java con html (por lo general en e HEAD).  *Html* <input type=”button” id=”un\_id” **onclick=fucion01()>** | |
| function funcion01(){  } | Estructura de una función en js. | |
| un\_id.style.height=”200px”;  var variable01=document.getElementById(“un\_id”);  variable01.style.height=”200px”; | Ejemplo de modificar propiedades de un input en html desde javascript.  Asignamos el id una variable y hereda todos sus métodos y propiedades, ejemplo. | |
| **EVENTOS** | | |
| Onload  Onclick  Ondbclick  Onfocus  onkeypress | Al cargar la paguina  Al clickear :  Al hacer doble click:  Cuando agarra el focus:  Al presionar una tecla : previamente el focus. | |

LABO3

|  |  |
| --- | --- |
| Funciones | Funciones aninimas |
| function suma(a,b){              return a+b;          } | Var nom=(a,b)=>{return a+b};  **uso** nom(12,2); |

DOM

ajax es la técnica para no recargar la paguina, solo se recarga la parte deseada.

|  |  |
| --- | --- |
|  |  |