---

title: Average

id: average

description: Average Component of mathematical script in Fill My Cycle Website

sidebar\_label: average

keywords: [average, mathematical average, function]

---

# Average Fuction:

#### Imagine you have a special table with rows of numbers, and you want to do something with those numbers. This function will help us with that. Let's go through each step:

1. We start by getting a table, which is like a special chart with rows and columns. We call this table "tbl". It's the table we want to work with and perform some actions on.

2. We create two special variables, "s" and "n". "s" will help us keep track of the sum of numbers, and "n" will keep track of how many rows there are in the table. We start with both variables set to 0.

3. Now, we go through each row of the table one by one. We start counting from 0 and keep going until we reach the last row. For each row, we do something special.

4. We check if the first input field in the row has a value in it. It's like looking at the box in that row and seeing if there's a number inside. If there is, we take that number and add it to our sum variable "s". But if there's nothing inside the box, we stop everything and return "false". It means there's a problem, like a missing number.

5. After going through all the rows, we divide the sum "s" by the total number of rows "n" to find the average. It's like adding up all the numbers and then dividing the total by how many numbers there are.

6. Next, we check if the average is less than 20. If it is, we set it to 20. It means that even if the average is really small, it should be at least 20.

7. Now, we check if the average is an odd number. If it is, we do some special adjustments to make it an even number. If the average is slightly less than an even number (like 21.3), we subtract the remainder (in this case, 1) to make it an even number (20). But if the average is slightly more than an even number (like 22.7), we add the difference (in this case, 0.3) to make it the next even number (24).

8. After all these calculations, we check which table we were working with. If it's a table called "tbl01", we find a special input field called "ip1" and put the average value there. But if it's a different table, we find a different input field called "ip2" and put the average value there.

So, the "avg(tbl)" function helps us find the average of numbers in a table, make sure it's not too small, adjust it to be an even number if needed, and then put the result in a specific input field. It's like a little helper that does all these things for us!<br />

As you see in the code below:

```script title="script.js" showLineNumbers

function avg(tbl){

var s=0,n=tbl.children.length;

for(var i=0;i<n;i++){

if(tbl.children[i].lastElementChild.firstElementChild.value){

s+= parseInt(tbl.children[i].lastElementChild.firstElementChild.value);

}

else{

return false;

}

}

s/=n;

if(s<20) s=20;

if(s%2!=0) {

if(s%2<1){

s=s-s%2;

}

else{

s+=(2-s%2);

}

}

if(tbl==$("#tbl01")[0]){

$("#ip1").val(s);

}

else{

$("#ip2").val(s);

}

}

```

---

title: Calculation

id: calculation

description: Calculation Component of mathematical script in Fill My Cycle Website

sidebar\_label: Calculation

keywords: [calculation, mathematical calculation, function]

---

# Calculation Function:

#### Here we will explain you the calculation function of the script:

1. The line (1) declares a variable called `man\_mode` and assigns it the value false. This variable is not used in the provided code snippet.

2. The line (3) defines a function named `calculate` that takes two parameters: `tbl` and `ip`. `tbl` represents a table element, and `ip` represents an input element.

3. The line (4) retrieves the value from the `ip` input element and converts it to a floating-point number using the `parseFloat` function. The converted value is then stored in the `marks` variable.

4. In the block of code (5-7) `if` statement checks if the length of the value in the `ip` input element is 2 characters, it's not equal to "10", and the marks value is less than or equal to 20. If all these conditions are true, it sets the `ip` value and marks value to 20.

5. In the block of code (8-10) `if` statement checks if the marks value is greater than 100. If it is, it sets the `ip` value and marks value to 100.

6. In the line (11) `if` statement checks if the marks value is between 20 and 100 (inclusive). If it is, the code inside this if statement is executed.

7. The block of code (12-20) checks if the `marks` value is odd (not divisible by 2). If it is, it adjusts the marks value to the nearest even number. If the decimal part of `marks % 2` is less than 1, it subtracts that decimal part from `marks` to make it even. Otherwise, it adds the difference between 2 and the decimal part to `marks` to make it even. Finally, it updates the `ip` value to the adjusted `marks` value.

8. In the block of code (21-31), several variables are declared and initialized.

\* `diff` is set to 0.

\* `l` is an array with initial values [5, 5, 5, 5, 5].

\* `w` is an array with initial values [6, 6, 2, 2, 4].

\* `lvl` is calculated by dividing the `marks` value by 20 and removing the decimal part using `marks - marks % 20`. Then it adds 1 if there is a remainder (`Math.ceil(marks % 20 / 20)`). The resulting value is used to set all elements of the `l` array to `lvl`.

\* If the `marks` value is not a multiple of 20 (`marks % 20 != 0`), the difference between 20 and the remainder is stored in the `diff` variable. Otherwise, `diff` is set to 0.

9. In the block of code (32-53) `while` loop is used to distribute the remaining `diff` value among the elements of the `l` array. The loop continues as long as `diff` is not zero. Inside the loop, different conditions are checked to reduce `diff` and decrement specific elements of the `l` array accordingly.

10. In the block of code (54-56) nested `for` loop is used to clear the inner HTML content of specific table cells (`<td>` elements) in the `tbl` table. It iterates over the rows (`i`) and columns (`j`) of the table and sets the `innerHTML` property of each cell to an empty string.

11. In the block of code (57-60) `for` loop iterates over the elements of the `l` array. For each element, it sets the inner HTML of a specific table cell to an HTML string that includes an image tag (`<img>`) with a source (`src`) attribute pointing to a checkmark image. The width and height of the image are also specified in the `style` attribute. Additionally, it multiplies the corresponding element of the `w` array with the corresponding element of the `l` array.

12. The block of code (61-63) adds the `marks` value to the end of the `w` array using the `push` method. Then, it returns the `w` array as the result of the `calculate` function.

13. As per line (64) if the marks value is not between 20 and 100 (inclusive), the function returns `false`.

```script title="script.js" showLineNumbers

var man\_mode = false;

function calculate(tbl,ip){

var marks=parseFloat(ip.value);

if(ip.value.length==2 && ip.value!="10" && marks<=20){

ip.value=marks=20;

}

if(marks>100){

ip.value=marks=100;

}

if(marks>=20 && marks<=100){

if(marks%2!=0) {

if(marks%2<1){

marks=marks-marks%2;

}

else{

marks+=(2-marks%2);

}

ip.value=marks;

}

var diff=0;

var l=[5,5,5,5,5];

var w=[6,6,2,2,4];

var lvl=(marks-marks%20)/20+Math.ceil(marks%20/20);

l[0]=l[1]=l[2]=l[3]=l[4]=lvl;

if(marks%20!=0){

diff=20-marks%20;

}

else{

diff=0;

}

while(diff){

if(diff>=6){

l[1]--;

diff-=6;

}

if(diff>=4){

l[4]--;

diff-=4;

}

if(diff>=2 && diff<=4){

l[2]--;

diff-=2;

}

if(diff>=6){

l[0]--;

diff-=6;

}

if(diff>=2){

l[3]--;

diff-=2;

}

}

for(var i=0;i<5;i++)

for(var j=1;j<6;j++)

tbl.children[i].children[j].innerHTML="";

for(var i=0;i<5;i++){

tbl.children[i].children[6-l[i] ].innerHTML='<img src="./assets/images/check.png" style= "height:50px; width:60px;">';

w[i]\*=l[i];

}

w.push(marks);

return w;

}

return false;

}

```

---

title: Checker

id: checker

description: Checker Component of mathematical script in Fill My Cycle Website

sidebar\_label: Checker

keywords: [Checker, mathematical Checker, function]

---

# Checker Function:

#### The `check()` function is like a special code that helps us do some cool things on a web page. Let's break it down step by step:

1. The function starts by looking for some important things on the page. It wants to find two tables, one called "tbl1" and another called "tbl2". It also wants to find two input fields, one called "ip1" and another called "ip2". It uses special commands like "document.getElementById()" to find these things.

2. Then, it checks if a certain key on the keyboard is pressed. The keys are like the buttons you press to type letters or numbers. The function wants to know if the key with the number 9 or the key with the number 16 is pressed. If it is, it will stop doing anything else and just go back to what it was doing before.

3. Next, it tries to figure out which input field is being used. It does this by looking at the place where you're typing, like the input field you clicked on. It looks at the "event" happening at that moment. An event is like a little message that tells the function what is happening. It tries to find the input field you're typing in by using "event.target" or "event.srcElement".

4. If you're typing something in the input field, the function checks how long the thing you typed is and if it's not equal to "10". If it has at least 2 characters and it's not exactly "10", something interesting happens!

5. It checks which input field you're typing in. If it's the first one, it does something called "focusing" on the second input field. It's like telling the computer to move the cursor to the second input field, so you can start typing there. If it's the second input field, it does something else. It stops you from typing more and takes you to a different part of the web page called "#result".

6. After that, the function checks if there's anything typed in the first input field. If there is, it uses another special function called "calculate()" to do some calculations using the first table and the first input field.

7. Then, it checks if there's anything typed in the second input field. If there is, it uses the same "calculate()" function to do some calculations using the second table and the second input field.

8. Finally, it does something called "cycle()" and gives it the results of the calculations it did before. This "cycle()" function probably does something really important with those results, but we don't know exactly what without looking at its code.

So, in simple terms, the `check()` function helps us find things on a web page, move around when we type, do some calculations, and maybe show us some results later. It's like a little helper that does all these tasks for us!

```script title="script.js" showLineNumbers

function check(){

var tbl1=document.getElementById("tbl1");

var tbl2=document.getElementById("tbl2");

var ip1=document.getElementById("ip1");

var ip2=document.getElementById("ip2");

if(event.keyCode==9 || event.keyCode==16)

return false;

var ip = event.target || event.srcElement;

if(ip.value){

if(ip.value.length>=2 && ip.value!="10"){

if(ip==ip1){

$(ip2).focus();

}

else if(ip==ip2){

$(ip).blur();

location.replace("#result");

}

}

}

var w=[];

if(ip1.value){

w[0]=calculate(tbl1,ip1);

}

if(ip2.value){

w[1]=calculate(tbl2,ip2);

}

cycle(w[0],w[1]);

}

```

---

title: Input Handler

id: input-handler

description: Input Handler Component of mathematical script in Fill My Cycle Website

sidebar\_label: Input Handler

keywords: [Input Handler, mathematical Input Handler, function]

---

# Input Handler

#### This function is triggered when a key is pressed on an input field. Let's go through each step:

1. We start by checking if the key code of the pressed key is either 9 (Tab key) or 16 (Shift key). If it is, we stop further execution of the function and return false. It's like ignoring certain keys that we don't want to respond to.

2. We identify the input field where the key was pressed. It's like finding the specific input field where we are typing.

3. We check if the input field has a value. If it does, we perform some actions inside this condition.

4. We check if the value in the input field has a length of at least 2 characters and is not equal to "10". If it meets these conditions, we proceed with further actions.

5. We find the next input field by navigating through the HTML elements. It's like finding the next input field in a sequence.

6. We also find the last input field in another table with the ID "tbl02". These fields are located somewhere else in the HTML structure.

7. We convert the value in the current input field to an integer and store it in a variable called "val". It's like reading the number entered in the input field.

8. We check if the value is less than 0. If it is, we set the value of the input field to 0. It's like ensuring that the value is not negative.

9. We check if the value is greater than 100. If it is, we set the value of the input field to 100. It's like ensuring that the value is not greater than 100.

10. We check if the current input field is the last input field in "tbl02". If it is, we blur the input field (remove focus) and replace the location hash with "#result". It's like moving the focus away from the input field and updating the URL.

11. If the current input field is not the last input field in "tbl02", we check if it is the last input field in its parent row. If it is, we find the first input field in the first row of "tbl02" and give it focus. It's like moving the focus to the first input field in the next row.

12. If none of the above conditions are met, we move the focus to the next input field using jQuery's "$()" function. It's like shifting the focus to the next input field in the sequence.

13. Finally, we call the "perform()" function, which is not defined in this code snippet but likely performs some other actions based on the changes made.

So, this function helps us respond to keyup events in input fields. It performs validation on the input value, navigates to the next input field, updates the URL, and triggers the "perform()" function. It's like a handler that listens to keyboard input and manages the flow of focus and actions accordingly!

```script title="script.js" showLineNumbers

function expkeyup(){

if(event.keyCode==9 || event.keyCode==16)

return false;

var ip = event.target || event.srcElement;

if(ip.value){

if(ip.value.length>=2 && ip.value!="10"){

var next=ip.parentElement.parentElement.nextElementSibling

if(next){

next=next.lastElementChild.firstElementChild;

}

var last1=ip.parentElement.parentElement.parentElement.lastElementChild.lastElementChild.firstElementChild;

var last2=document.getElementById("tbl02").lastElementChild.lastElementChild.firstElementChild;

var val = parseInt(ip.value);

if(val<0){

ip.value=0;

}

else if(val>100){

ip.value=100;

}

if(ip==last2){

$(ip).blur();

location.replace("#result");

}

else if(ip==last1){

$("#tbl02 tr input:first").focus();

}

else{

$(next).focus();

}

}

}

perform();

}

```

---

title: Page Interaction

id: page-interaction

description: Page Interaction Component of mathematical script in Fill My Cycle Website

sidebar\_label: Page Interaction

keywords: [Page Interaction, mathematical Page Interaction, function]

---

# Page Interaction:

####Here's what the function does step by step:

1. It focuses on a specific field with an ID called "c11". It's like pointing your finger at that field so you can start typing in it.

2. It simulates a click on the "c11" field. It's like tapping on it with your finger.

3. It triggers a "click" event on the "c11" field. It's like pretending you clicked on it again.

4. It listens for when you press a key on the keyboard while typing in the "c11" field. If you press certain keys like the Tab key or Shift key, it won't do anything. It's like ignoring those keys.

5. It gets the value you typed in the "c11" field and checks if it's greater than 15. If it is, it changes the value to 15. It's like making sure the number is not too big.

6. If the value you typed is greater than another field called "c10", it triggers a function called "c11change". It's like telling another part of the page to do something if the number is bigger.

7. If you typed at least one character in the "c11" field that is not "1", it focuses on another field called "c21". It's like moving your attention to that field.

8. The function continues and does similar things for the "c21" field. It listens for key presses, checks the value, triggers a function called "c21change" if necessary, and focuses on another field if certain conditions are met.

9. It sets up listeners for when you type or change values in fields with a class called "exp". It's like keeping an ear out for any changes you make in those fields.

10. It sets up listeners for when you type or change values in fields with a class called "ip". It's like keeping an ear out for any changes you make in those fields as well.

11. It sets up a listener for when you click a button with a class called "reset". When you click that button, it clears all the values in input fields and performs some actions to reset the page.

12. It sets up a listener for when you click a button with an ID called "man\_mode". When you click that button, it switches the page to a different mode.

13. It sets up listeners for when you click on certain cells in a table with an ID called "tbl1" or "tbl2". When you click on those cells, it triggers a function called "cell\_check".

14. It sets up listeners for when you click on certain elements with classes "instr-ref" or "man-mode-ref". When you click on those elements, it changes the location of the page to show different instructions or information.

Overall, this function sets up all the interactions and behaviors for the web page so that you can use it easily and perform different actions by clicking, typing, or changing values.

```script title="script.js" showLineNumbers

function c11change(){

var c10=parseInt($("#c10").text());

var c11=parseInt($("#c11").val());

var tbl01=document.getElementById("tbl01");

var tbl02=document.getElementById("tbl02");

var tbl3=document.getElementById("tbl3");

if(c11 || c11==0){

if(c11<=c10){

c11=c10;

$("#c11").val(c11);

}

var ct = tbl01.children.length;

var pc20 = parseInt($("#c20").text());

var pc21 = parseInt($("#c21").val());

if($("#c21").val()==""){

pc21=pc20+$("#tbl02").children().length-1;

}

var c20 = c11+1;

$("#c20").text(c20);

if(c11<ct){

var diff=ct-c11;

while(diff--){

$("#tbl02 tr:first").before(tbl01.lastElementChild);

}

}

else if(c11>=ct){

if(c11>=pc21){

var t=pc21-ct;

while(t--){

$("#tbl01").append(tbl02.firstElementChild);

}

$("#tbl02").append('<tr>\

<td>'+c20+'</td>\

<td><input class="form-control input-sm exp" type="number" min="0" max="100" step="1"/></td>\

</tr>');

var diff=c11-pc21;

var c=pc21+1;

while(diff--){

$("#tbl01").append('<tr>\

<td>'+c+'</td>\

<td><input class="form-control input-sm exp" type="number" min="0" max="100" step="1"/></td>\

</tr>');

c++;

}

}

else{

var diff=c11-ct;

while(diff--){

$("#tbl01").append(tbl02.firstElementChild);

}

}

$("input.exp").on("keyup", expkeyup);

$("input.exp").on("change", perform);

}

perform();

}

var ct1 = tbl01.children.length;

var ct2 = tbl02.children.length;

tbl3.children[0].children[0].innerText="1 to "+ct1;

tbl3.children[1].children[0].innerText=(ct1+1)+" to "+(ct1+ct2);

}

```

:::tip

Same Logic has been used for `c21change()` function for second table.

:::

---

title: Perform

id: perform

description: Perform Component of mathematical script in Fill My Cycle Website

sidebar\_label: Perform

keywords: [Perform, mathematical Perform, function]

---

# Perform Function

#### Imagine you have a special show where you need to do different tasks in a certain order. This function "perform()" helps us with that. Let's go through each step:

1. First, we find a table called "tbl01". It's like a special chart with rows and columns. We use a special code called "document.getElementById()" to find this table.

2. Then, we do something special with this table. We call another function called "avg(tbl01)". It's like telling someone to find the average of the numbers in that table. So, we're asking for the average of all the numbers in the table "tbl01".

3. We repeat the same process with another table called "tbl02". We find it using the same special code "document.getElementById()" and then call the "avg(tbl02)" function to find the average of the numbers in that table too.

4. After finding the averages of both tables, we move on to the next task. We call another function called "check()". It's like asking someone to perform a checklist or a set of instructions.

5. Inside the "check()" function, there might be more instructions and actions happening, but we don't know the details without looking at the code of that function.

So, the "perform()" function helps us find and work with specific tables, calculate the averages of the numbers in those tables, and then move on to perform additional actions using the "check()" function. It's like a script for a show, where we do things in a specific order to make everything work smoothly!

```script title="script.js" showLineNumbers

function perform(){

var tbl01=document.getElementById("tbl01");

avg(tbl01);

var tbl02=document.getElementById("tbl02");

avg(tbl02);

check();

}

```

---

title: Table Calculator

id: table-calculator

description: Table Calculator Component of mathematical script in Fill My Cycle Website

sidebar\_label: Table Calculator

keywords: [Table Calculator, mathematical Table Calculator, function]

---

# Table Calculator:

#### Let's break down the JavaScript function "cal\_man(tbl)" in a simple way:

This function calculates and manages some values based on a table. Let's go through each step:

1. We start with an array called "w" which holds some initial values: [6, 6, 2, 2, 4, 0]. It's like having a list of numbers to work with.

2. We go through a loop that repeats 5 times. In each iteration, we perform some actions.

3. Inside the loop, we find a specific row in the table using the variable "tr". It's like picking a row from the table.

4. Then, we find a specific cell in that row which contains an image. We store this cell in the variable "td". It's like selecting a cell with an image.

5. We calculate the column index of the selected cell relative to its row. It's like finding the position of the cell within the row.

6. If the calculated column index is less than 0 (which means it's an invalid index), we stop further execution of the function and return false. It's like checking if the column index is valid.

7. If the column index is valid, we perform some calculations using the values in the "w" array. It's like doing some mathematical operations.

8. We multiply the value in the "w" array at index "i" by (5 - col), where "col" is the calculated column index. It's like multiplying a number by another number and storing the result.

9. We add the calculated value to the last element of the "w" array (at index 5). It's like accumulating the calculated values.

10. After the loop is completed, we return the updated "w" array. It's like giving back the final result.

So, the "cal\_man(tbl)" function calculates and manages values based on a table's rows and columns. It multiplies and accumulates numbers based on certain conditions and returns the updated array of values. It's like performing some calculations and keeping track of the results!

```script title="script.js" showLineNumbers

function cal\_man(tbl){

var w=[6,6,2,2,4,0];

for(var i=0;i<5;i++){

var tr=$(tbl.children[i]);

var td=tr.children().children('img').parent();

var col=tr.children().index(td)-1;

if(col<0){

return false;

}

w[i]\*=(5-col);

w[5]+=w[i];

}

return w;

}

```

---

title: Table Manager

id: table-manager

description: Table manager Component of mathematical script in Fill My Cycle Website

sidebar\_label: Table Manager

keywords: [Table Manager, mathematical Table Manager, function]

---

# Table Manager

#### This function checks and manages values in two tables. Let's go through each step:

1. We start by finding a table with the ID "tbl1" and storing it in the variable "tbl1". It's like picking the first table.<br />

2. Then, we find another table with the ID "tbl2" and store it in the variable "tbl2". It's like picking the second table.<br />

3. We create an empty array called "w". It's like having a list to store values.<br />

4. We calculate some values for the first table using the "cal\_man()" function and store the result in the first element of the "w" array (at index 0).<br />

5. Similarly, we calculate some values for the second table using the "cal\_man()" function and store the result in the second element of the "w" array (at index 1).<br />

6. Finally, we call the "cycle()" function, passing the calculated values from the first table (w[0]) and the second table (w[1]) as arguments. It's like performing some action with those values.<br />

So, the "check\_man()" function checks and manages values in two tables. It calculates some values for each table and performs an action with the calculated results. It's like checking and handling the values in a organized way!

```script title="script.js" showLineNumbers

function check\_man(){

var tbl1=document.getElementById("tbl1");

var tbl2=document.getElementById("tbl2");

var w=[];

w[0]=cal\_man(tbl1);

w[1]=cal\_man(tbl2);

cycle(w[0],w[1]);

}

```

---

title: Table Transformation

id: table-transformation

description: Table Transformation Component of mathematical script in Fill My Cycle Website

sidebar\_label: Table Transformation

keywords: [Table Transformation, mathematical Table Transformation, fuction]

---

# Table Transformation:

#### This function performs a cycle of actions related to tables. Let's go through each step:

1. We start by finding three tables: "tbl1", "tbl2", and "tbl3". It's like picking specific tables.

2. We create an array called "w" to store values: [w0, w1, [], []]. It's like having different containers for different things.

3. If the first element of "w" (w[0]) exists, we update the first row of "tbl3" with the values from w[0]. It's like displaying some values in the first row of a table.

4. If w[0] doesn't exist, we clear the cells in "tbl1" (except the first column) or leave them empty. It's like erasing or leaving empty cells in a table.

5. Regardless of w[0]'s existence, we clear the first row of "tbl3" or leave it empty. It's like erasing or leaving empty cells in the first row of a table.

6. If the second element of "w" (w[1]) exists, we update the second row of "tbl3" with the values from w[1]. It's like displaying some values in the second row of a table.

7. If w[1] doesn't exist, we clear the cells in "tbl2" (except the first column) or leave them empty. It's like erasing or leaving empty cells in a table.

8. Regardless of w[1]'s existence, we clear the second row of "tbl3" or leave it empty. It's like erasing or leaving empty cells in the second row of a table.

9. If both w[0] and w[1] exist, we perform calculations by adding the corresponding values from w[0] and w[1] and store them in w[2]. We also calculate the average by dividing the values in w[2] by 2 and store them in w[3].

10. We update the third row of "tbl3" with the values from w[2] and the fourth row of "tbl3" with the values from w[3]. It's like displaying calculated values in specific rows of a table.

11. If either w[0] or w[1] doesn't exist, we clear the cells in the third and fourth rows of "tbl3" or leave them empty. It's like erasing or leaving empty cells in specific rows of a table.

So, the "cycle(w0, w1)" function performs a cycle of actions on tables. It displays values, performs calculations, and updates specific rows of a table based on the provided values. It's like going through a sequence of steps to manage and update the tables accordingly!

```script title="script.js" showLineNumbers

function cycle(w0,w1){

var tbl1=document.getElementById("tbl1");

var tbl2=document.getElementById("tbl2");

var tbl3=document.getElementById("tbl3");

var w=[];

w[0]=w0,w[1]=w1;

w[2]=[],w[3]=[];

if(w[0]){

for(var i=0;i<6;i++){

tbl3.children[0].children[1+i].innerText=w[0][i];

}

}

else{

if(!man\_mode){

for(var i=0;i<5;i++) for(var j=1;j<6;j++)

tbl1.children[i].children[j].innerHTML="";

}

for(var i=0;i<6;i++){

tbl3.children[0].children[1+i].innerText="";

}

}

if(w[1]){

for(var i=0;i<6;i++){

tbl3.children[1].children[1+i].innerText=w[1][i];

}

}

else{

if(!man\_mode){

for(var i=0;i<5;i++) for(var j=1;j<6;j++)

tbl2.children[i].children[j].innerHTML="";

}

for(var i=0;i<6;i++){

tbl3.children[1].children[1+i].innerText="";

}

}

if(w[0] && w[1]){

for(var i=0;i<6;i++){

w[2][i]=w[0][i]+w[1][i];

w[3][i]=w[2][i]/2;

tbl3.children[2].children[1+i].innerText=w[2][i];

tbl3.children[3].children[1+i].innerText=w[3][i];

}

}

else{

for(var i=0;i<6;i++){

tbl3.children[2].children[1+i].innerText="";

tbl3.children[3].children[1+i].innerText="";

}

}

}

```