

# Introduction to Programming

Class 24, 17 November 2016

Jack Phillips <[jack\\_phillips@asl.org](mailto:jack_phillips@asl.org)>

- Sit where you want today. Make sure you have a partner.

## Goals

**Goal 1:** You will understand how function arguments and variables work.

**Goal 2:** You will know how to get input from a webpage.

## Vocabulary

variable context  
element id

## Code

```
document.getElementById("name")  
element.value  
element.innerHTML
```

## Partners

- Sit where you want today.
- **If you were to change your name, what would you change your name to? Why?**

# BIG IDEAS!

What is modularization?

1. Modularization is a programming technique in which code broken down into smaller self-contained pieces.

Go over test

# Variable Context

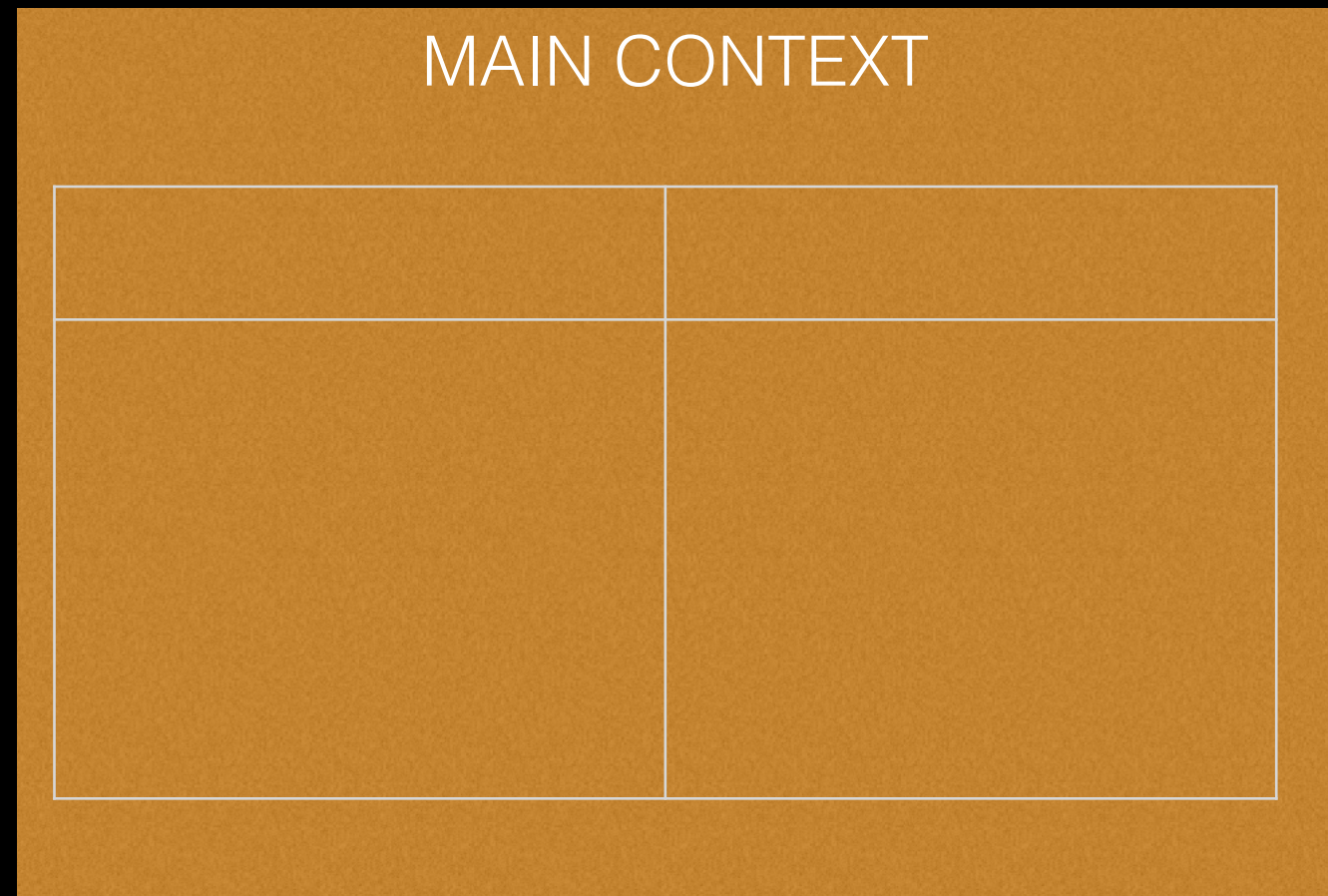
## double.js

```
var value = 5;  
show_double(value);  
alert(value);  
  
function double(num) {  
    var result = num * 2;  
    alert(result);  
    value = "Hi!";  
}
```

## double.js

### Pre-processing

```
var value = 5;  
show_double(value);  
alert(value);  
  
function show_double(num) {  
    var result = num * 2;  
    alert(result);  
    value = "Hi!";  
}
```



During pre-processing, the javascript interpreter creates a variable context and looks for var declarations and function definitions.



## double.js

### Pre-processing

➔

```
var value = 5;  
show_double(value);  
alert(value);  
  
function show_double(num) {  
    var result = num * 2;  
    alert(result);  
    value = "Hi!";  
}
```

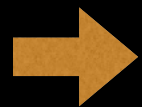
#### MAIN CONTEXT

value	undefined

## double.js

### Pre-processing

```
var value = 5;  
show_double(value);  
alert(value);
```



```
function show_double(num) {  
  var result = num * 2;  
  alert(result);  
  value = "Hi!";  
}
```

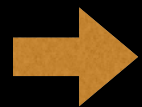
#### MAIN CONTEXT

value	undefined
show_double	undefined

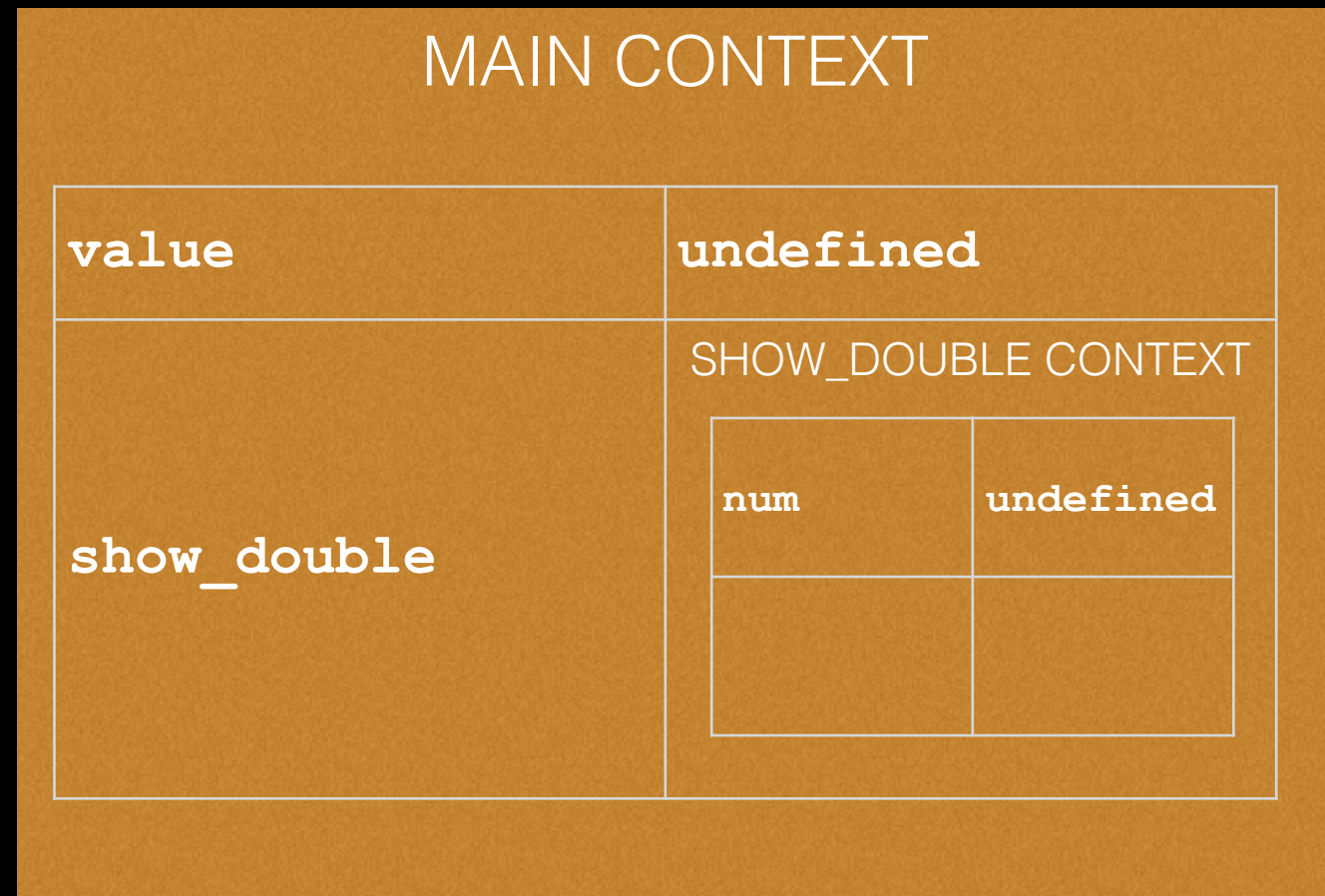
## double.js

### Pre-processing

```
var value = 5;  
show_double(value);  
alert(value);
```



```
function show_double(num) {  
  var result = num * 2;  
  alert(result);  
  value = "Hi!";  
}
```

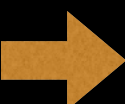


For each function definition, a variable context is created.

## double.js

### Pre-processing

```
var value = 5;  
show_double(value);  
alert(value);
```



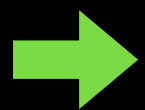
```
function show_double(num) {  
  var result = num * 2;  
  alert(result);  
  value = "Hi!";  
}
```

#### MAIN CONTEXT

value	undefined				
show_double	SHOW_DOUBLE CONTEXT				
	<table><tr><td>num</td><td>undefined</td></tr><tr><td>result</td><td>undefined</td></tr></table>	num	undefined	result	undefined
	num	undefined			
result	undefined				

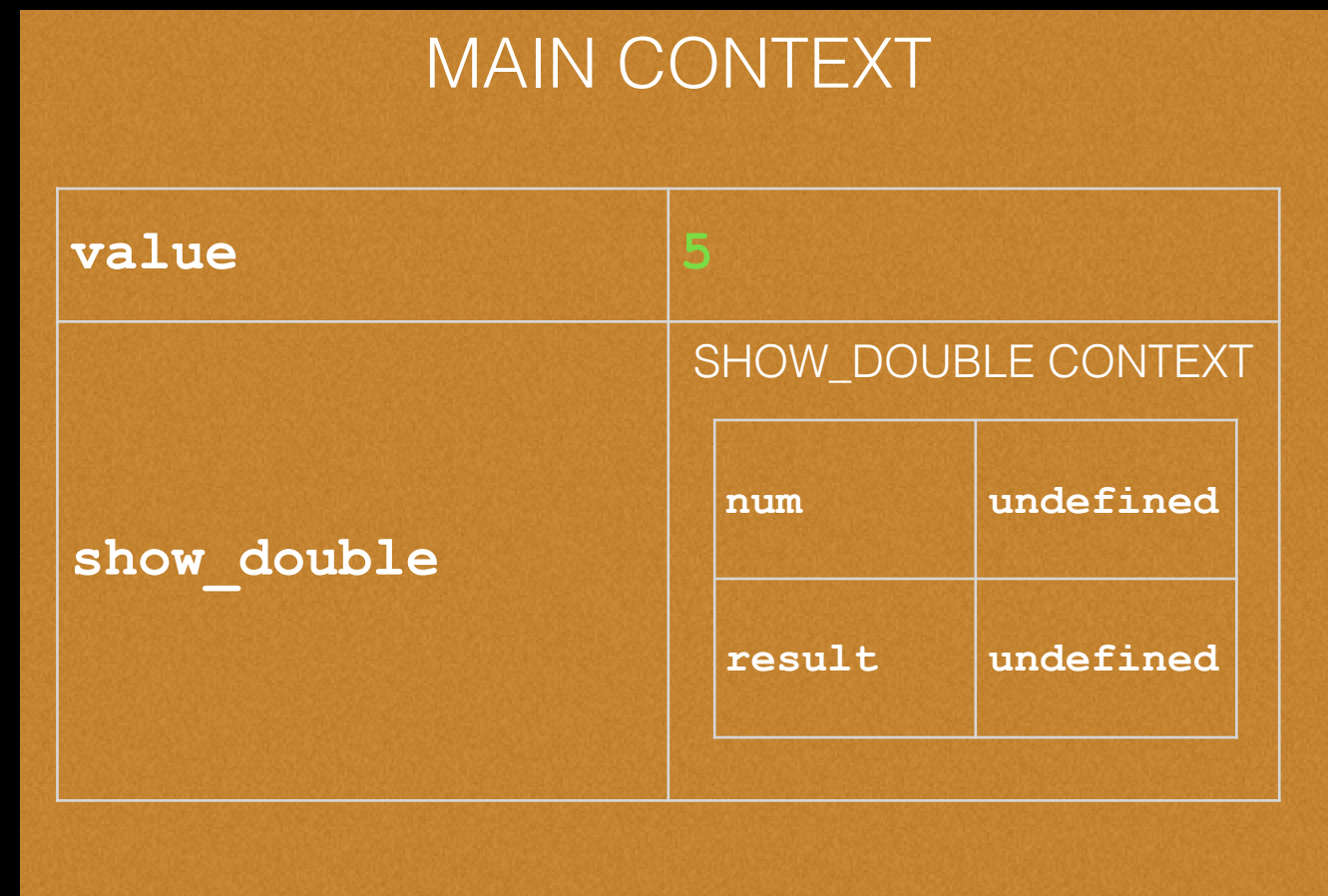
## double.js

### Executing



```
var value = 5;  
show_double(value);  
alert(value);
```

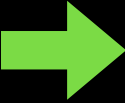
```
function show_double(num) {  
  var result = num * 2;  
  alert(result);  
  value = "Hi!";  
}
```



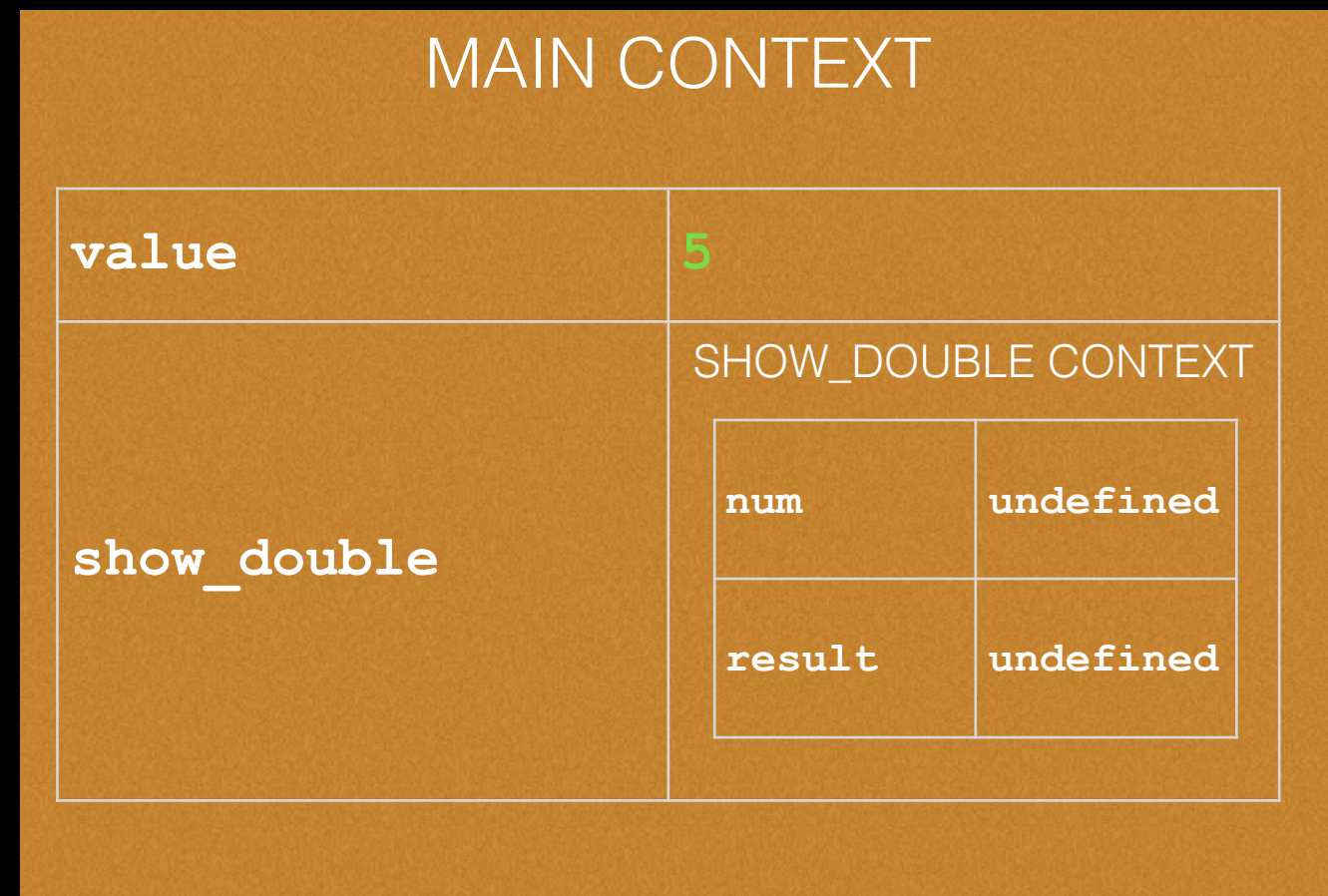
During execution, each line of code is run in turn.  
Variable assignments and function calls change the contexts.

## double.js

### Executing



```
var value = 5;  
show_double(value);  
alert(value);  
  
function show_double(num) {  
    var result = num * 2;  
    alert(result);  
    value = "Hi!";  
}
```

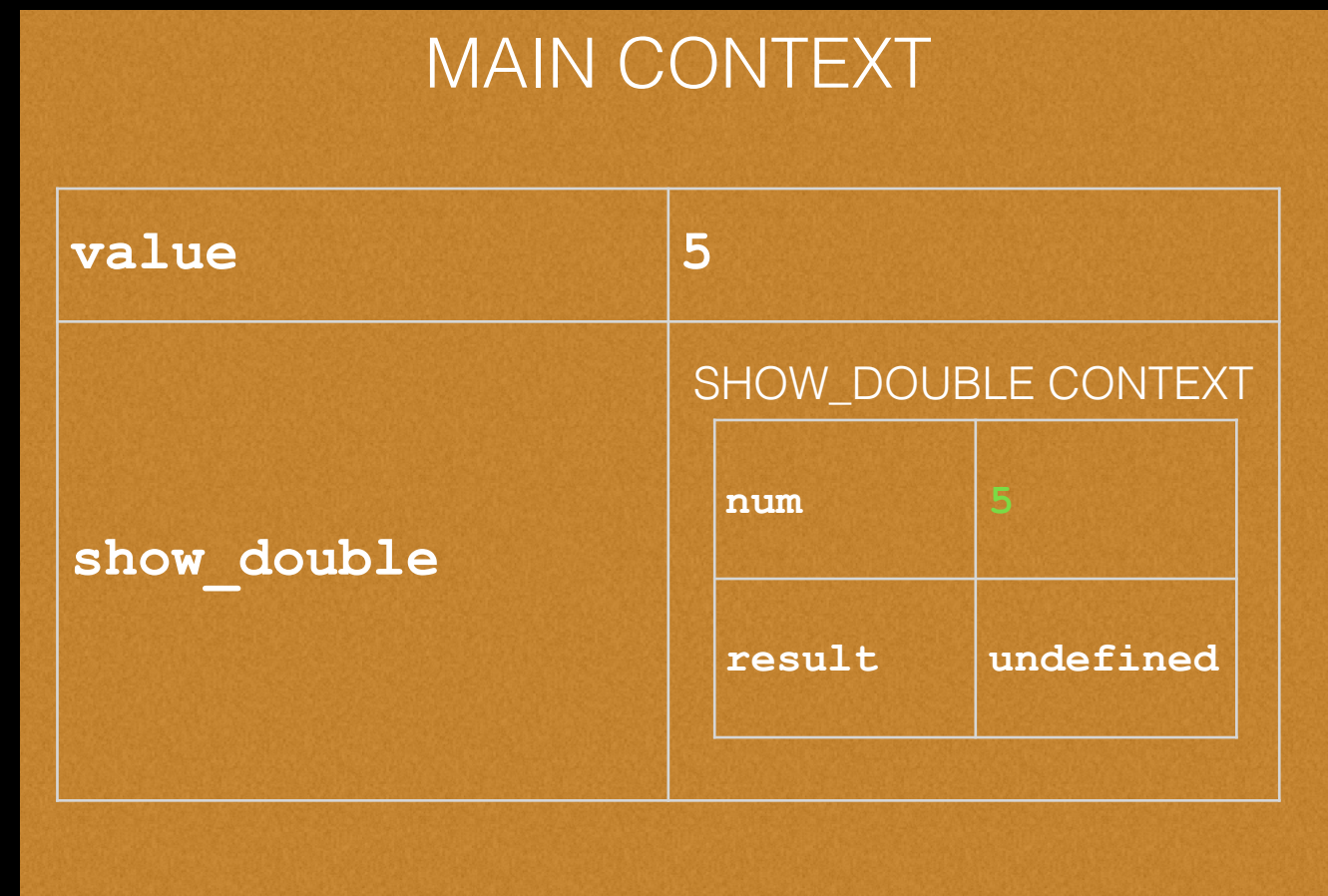




## double.js

### Executing

```
var value = 5;  
show_double(value);  
alert(value);  
  
→ function show_double(num) {  
    var result = num * 2;  
    alert(result);  
    value = "Hi!";  
}
```

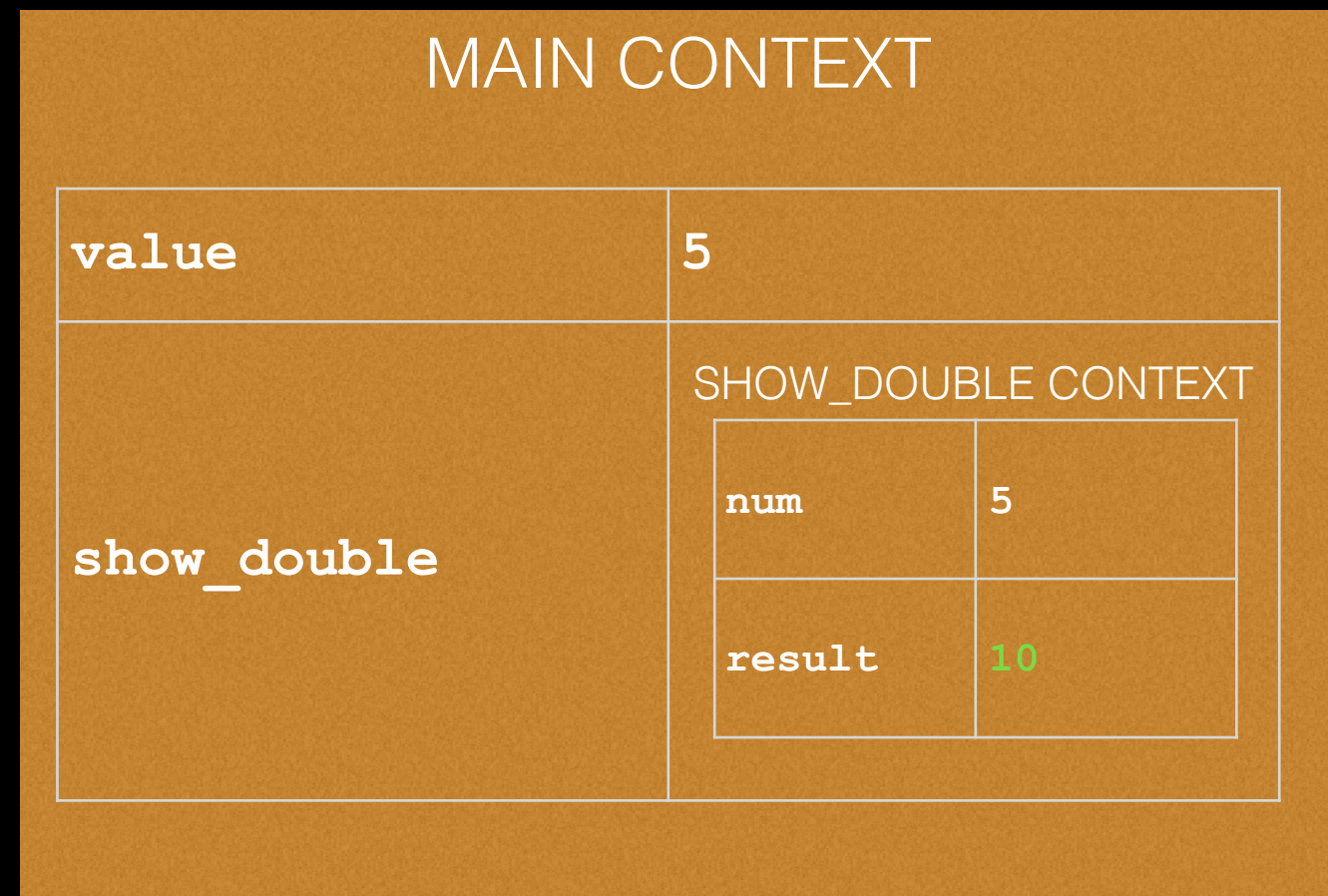


During a function call, values are passed as arguments into the function context.

## double.js

### Executing

```
var value = 5;  
show_double(value);  
alert(value);  
  
function show_double(num) {  
    var result = num * 2;  
    alert(result);  
    value = "Hi!";  
}
```

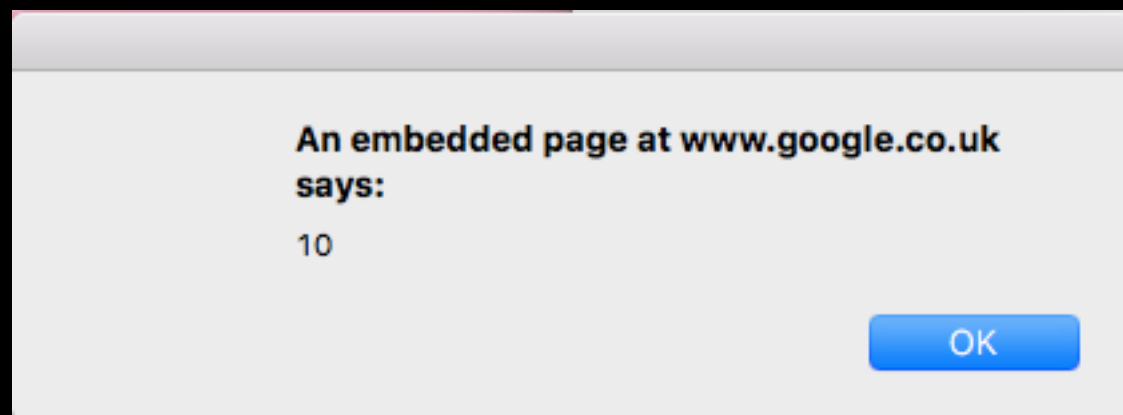
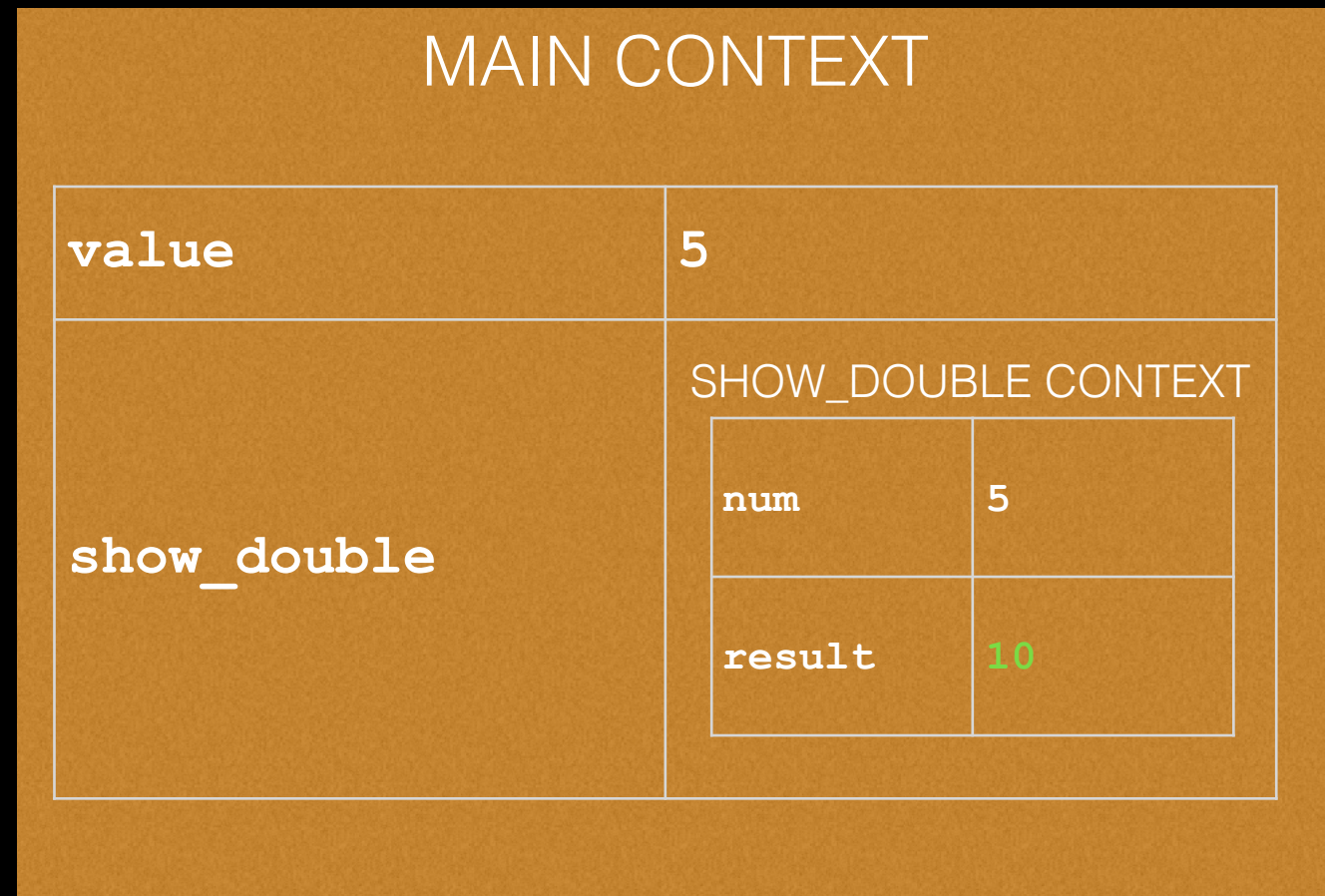
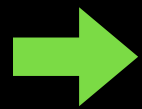




## double.js

### Executing

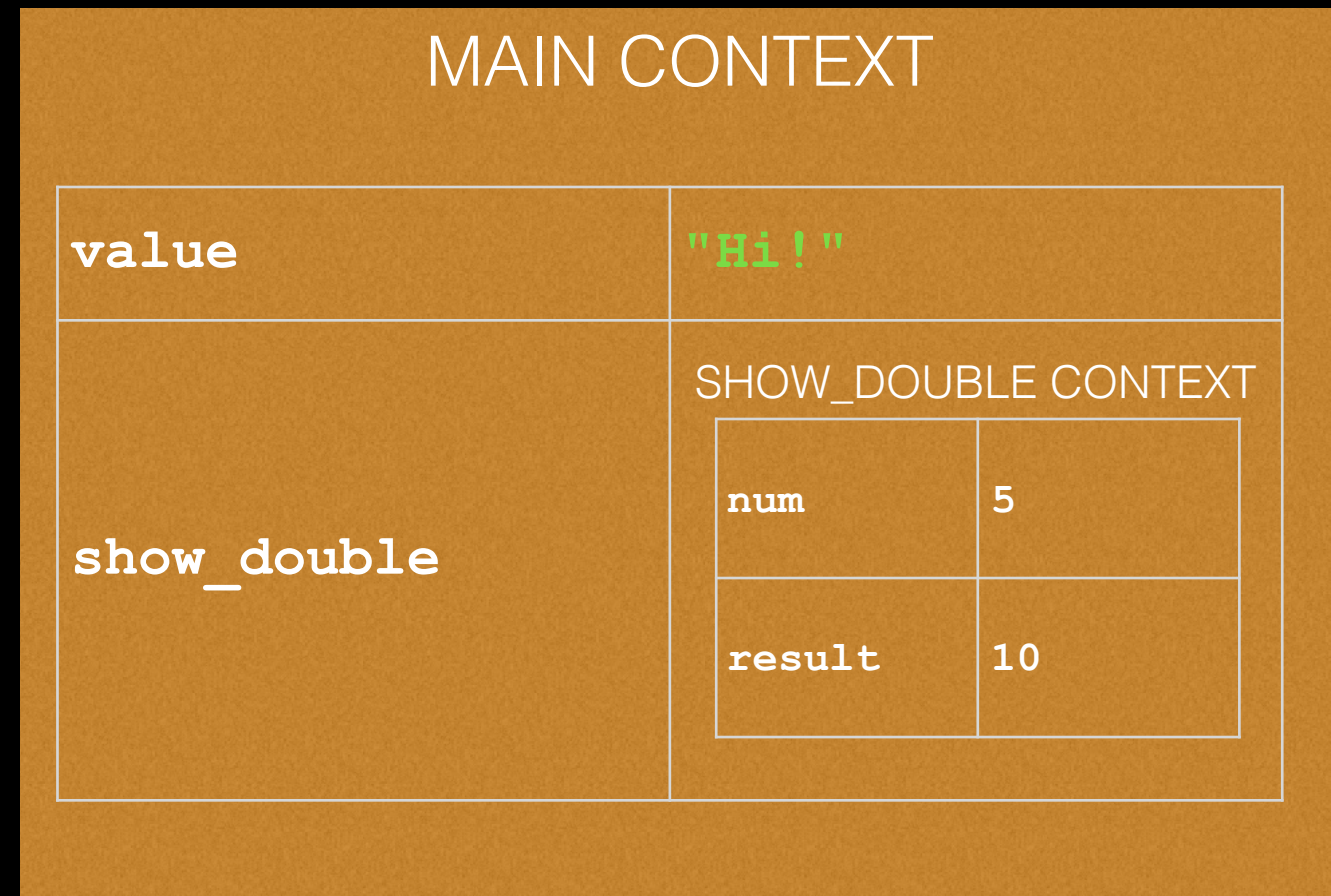
```
var value = 5;  
show_double(value);  
alert(value);  
  
function show_double(num) {  
    var result = num * 2;  
    alert(result);  
    value = "Hi!";  
}
```



## double.js

### Executing


```
var value = 5;  
show_double(value);  
alert(value);  
  
function show_double(num) {  
    var result = num * 2;  
    alert(result);  
    value = "Hi!";  
}
```



The value of a variable can change in any context **within which it is nested**. E.g., `show_double` can change the value of `value`. But the main context cannot change the value of `result`.

## double.js

### Executing



```
var value = 5;  
show_double(value);  
alert(value);
```

```
function show_double(num) {  
    var result = num * 2;  
    alert(result);  
    value = "Hi!";  
}
```

#### MAIN CONTEXT

value	"Hi!"				
show_double	<div>SHOW_DOUBLE CONTEXT</div> <table><tr><td>num</td><td>5</td></tr><tr><td>result</td><td>10</td></tr></table>	num	5	result	10
num	5				
result	10				

An embedded page at [www.google.co.uk](http://www.google.co.uk) says:

Hi!

☐ Prevent this page from creating additional dialogs.

OK

## double.js

```
var value = 5;  
show_double(value);  
alert(value);  
  
function show_double(num) {  
  var result = num * 2;  
  alert(result);  
  value = "Hi!";  
}
```



void  
function

## betterDouble.js

```
var value = 5;  
var new_value = show_double(value);  
alert(new_value);  
  
function show_double(num) {  
  var result = num * 2;  
  alert(result);  
  return "Hi!";  
}
```



return  
function

Generally, it's better to use a return statement rather than set variables outside a function context. Code is more modularized!



**DO**

**Create a variable context table for the following program.  
Make sure you think about the order in which the pre-processor and execution work.**

# sumAges.js

```
main();

function main() {
    var first_age = parseInt(prompt("Enter your age:"));
    var second_age = parseInt(prompt("Enter your friend's age:"));
    var total = sum(first_age, second_age);
    alert("Together, you are " + total + " years old");
}

function sum(num1, num2) {
    var result = num1 + num2;
    return result;
}
```

main	first_age	undefined
	second_age	undefined
	total	undefined
sum	num1	undefined
	num2	undefined
	result	undefined

# sumAges.js

```
main();

function main() {
    var first_age = parseInt(prompt("Enter your age:"));
    var second_age = parseInt(prompt("Enter your friend's age:"));
    var total = sum(first_age, second_age);
    alert("Together, you are " + total + " years old");
}

function sum(num1, num2) {
    var result = num1 + num2;
    return result;
}
```

main	first_age	15
	second_age	16
	total	undefined
sum	num1	undefined
	num2	undefined
	result	undefined

## sumAges.js

```
main();

function main() {
    var first_age = parseInt(prompt("Enter your age:"));
    var second_age = parseInt(prompt("Enter your friend's age:"));
    var total = sum(first_age, second_age);
    alert("Together, you are " + total + " years old");
}

function sum(num1, num2) {
    var result = num1 + num2;
    return result;
}
```

main	first_age	15
	second_age	16
	total	undefined
sum	num1	15
	num2	16
	result	undefined



# sumAges.js

```
main();

function main() {
    var first_age = parseInt(prompt("Enter your age:"));
    var second_age = parseInt(prompt("Enter your friend's age:"));
    var total = sum(first_age, second_age);
    alert("Together, you are " + total + " years old");
}

function sum(num1, num2) {
    var result = num1 + num2;
    return result;
}
```

main	first_age	15
	second_age	16
	total	undefined
sum	num1	15
	num2	16
	result	31

# sumAges.js

```
main();

function main() {
    var first_age = parseInt(prompt("Enter your age:"));
    var second_age = parseInt(prompt("Enter your friend's age:"));
    var total = sum(first_age, second_age);
    alert("Together, you are " + total + " years old");
}

function sum(num1, num2) {
    var result = num1 + num2;
    return result;
}
```

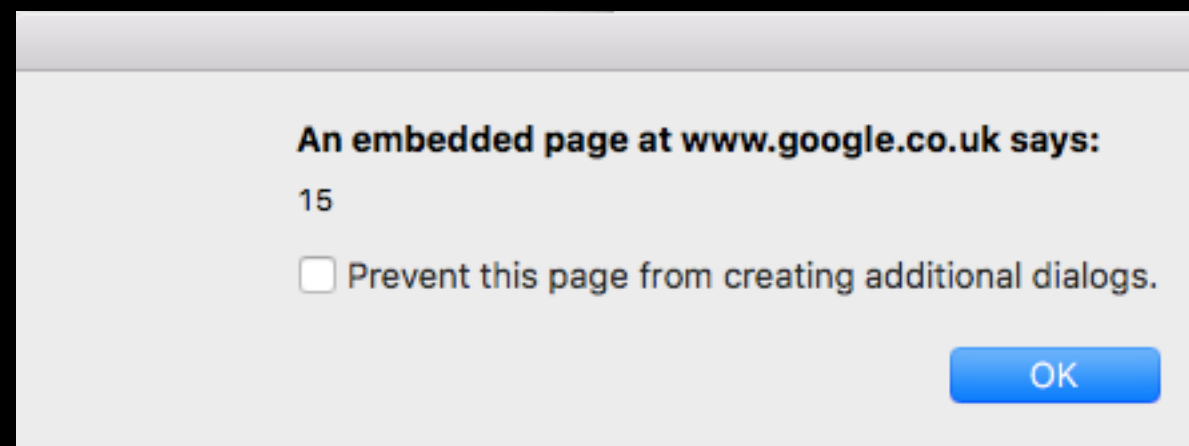
main	first_age	15
	second_age	16
	total	31
sum	num1	15
	num2	16
	result	31

## sumAges.js

```
main();

function main() {
    var first_age = parseInt(prompt("Enter your age:"));
    var second_age = parseInt(prompt("Enter your friend's age:"));
    var total = sum(first_age, second_age);
    alert(first_age);
}

function sum(num1, num2) {
    var result = num1 + num2;
    first_age = "WHAT?";
    return result;
}
```



# sumAges.js

```
main();

function main() {
  var first_age = parseInt(prompt("Enter your age:"));
  var second_age = parseInt(prompt("Enter your friend's age:"));
  var total = sum(first_age, second_age);
  alert(first_age);
}

function sum(num1, num2) {
  var result = num1 + num2;
  first_age = "WHAT?";
  return result;
}
```

Assigning a variable without the keyword `var` is like creating a global variable.

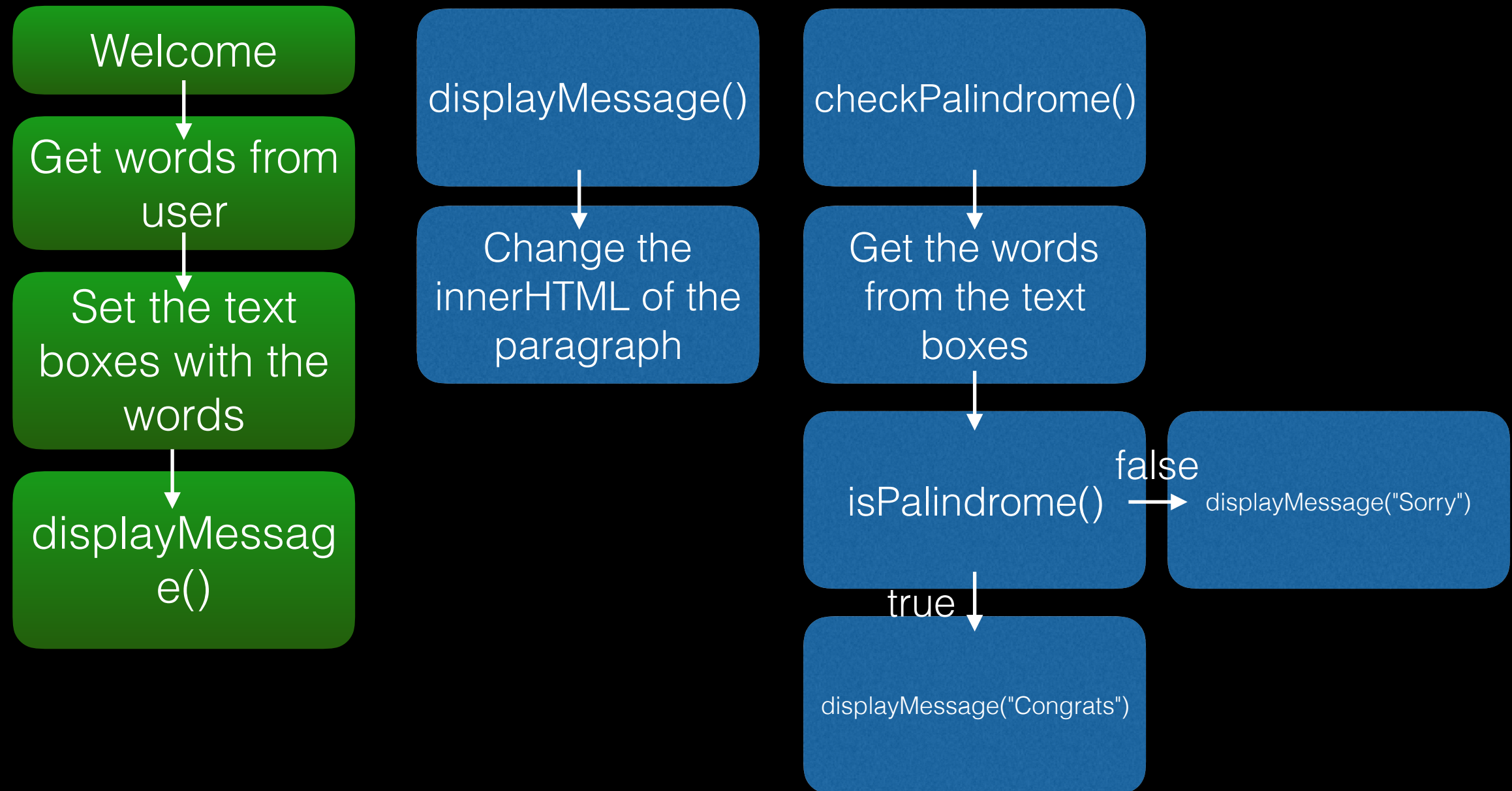
main	first_age	15
	second_age	16
	total	31
sum	num1	15
	num2	16
	result	31
first_age	"WHAT?"	

DO

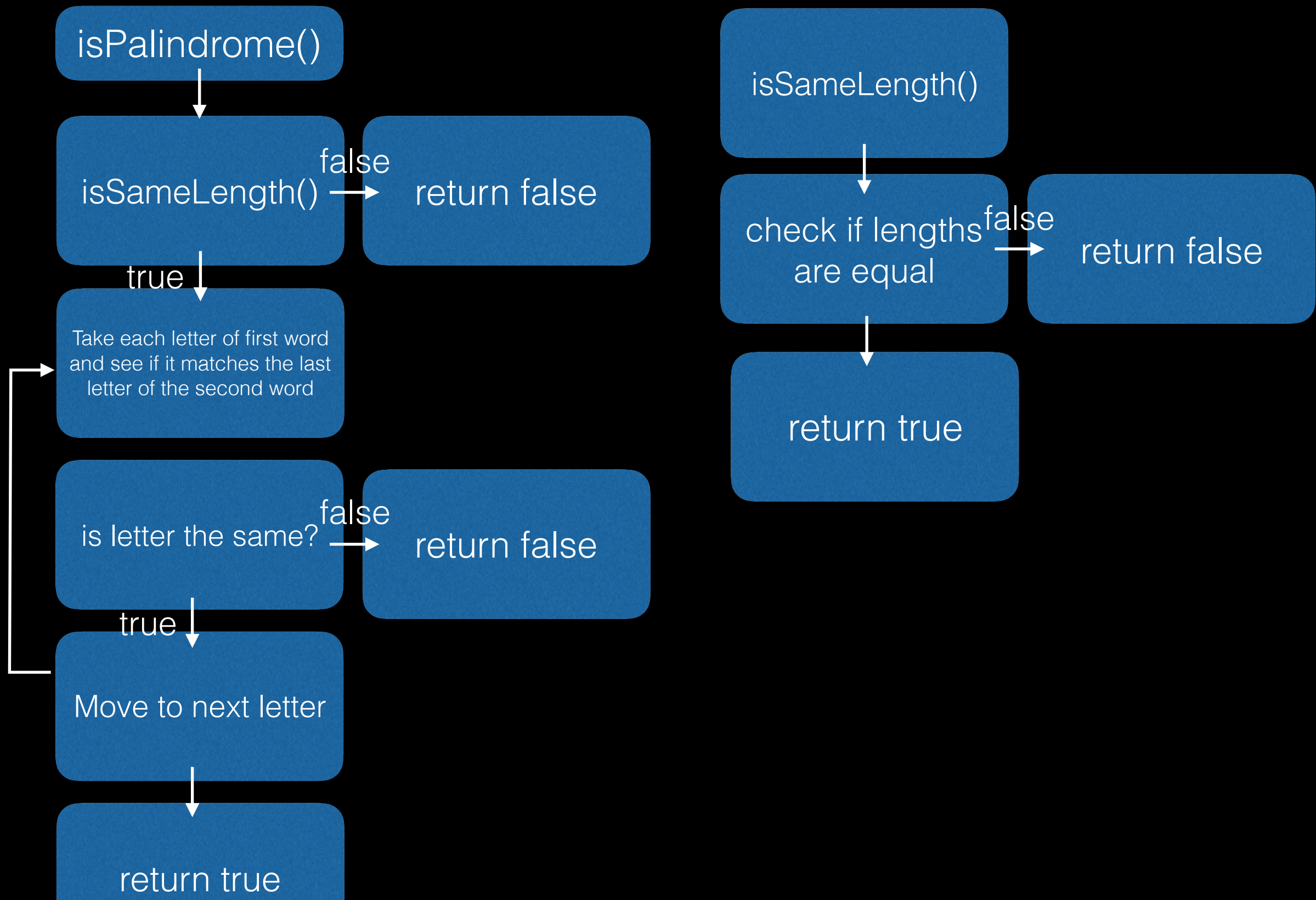
1. Run palindrome.html.
2. Create a flowchart of palindrome.js.
3. Create a variable context table of palindrome.js
4. Describe what the following code does:

```
<p>Word 1:<input type="text" id="word1"></p>  
<button onclick="checkPalindrome()">Check</button>  
  
var textBox1 = document.getElementById("word1");  
  
    var word1 = textBox1.value;  
  
    response.innerHTML = message;
```

## palindrome.js



## palindrome.js





<pre>&lt;p&gt;Word 1:&lt;input type="text" id="word1"&gt; &lt;/p&gt;</pre>	Creates a <b>text box element</b> inside a paragraph element and gives it the <b>id</b> of word1
<pre>&lt;button onclick="checkPalindrome()"&gt;Check &lt;/button&gt;</pre>	Creates a <b>button element</b> with the text "Check" and assigns the function checkPalindrom() to it when clicked
<pre>var textBox1 = document.getElementById("word1");</pre>	"Gets" the text box element with the id word1 and saves it in the variable textBox1
<pre>var word1 = textBox1.value;</pre>	"Gets" the <b>text in the text box</b> element and saves it in the variable word1
<pre>response.innerHTML = message;</pre>	Sets the <b>text</b> in the paragraph element called response to message.





**EXPLORE**

- 1. Look at `button.html`, `button.js`, `textbox.html`, and `textbox.js`.**
- 2. Modify the code to do something fun.**



**DO**

- 1. Modify your code from Task 4-1 to 4-3 to use a webpage for input and output rather than prompts and alerts.**