

# Introduction to Programming

Class 23, 23 March 2017

- 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

## Code

## WARM UP

**What does the following code display?**

```
var s = "Hello";  
s = duplicate(s);  
var t = "Bye";  
t = duplicate(duplicate(duplicate(t)));  
alert(s + t);  
  
function duplicate(s) {  
    return s + s;  
}
```

## DISCUSS

- **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.

## Four Types of Functions

	Void	Return
No arguments		
Arguments		

**DISCUSS**

**How many times will the loop execute?**

```
var i = 0;
while (i<1000) {
    cube(i)
    i++
}

function cube(i) {
    i = i * i * i;
}
```

# Variable Context



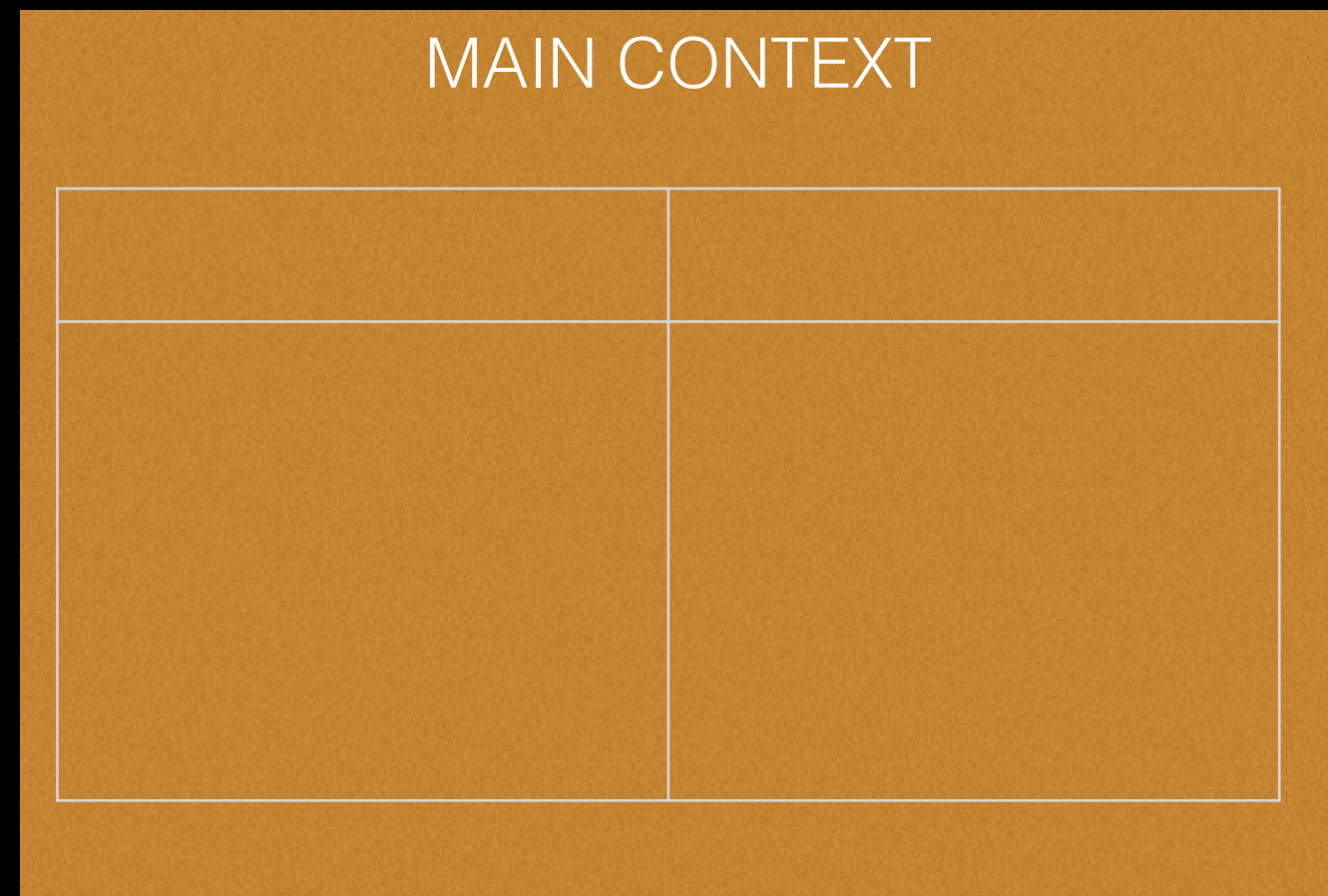
## double.js

```
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!";  
}
```

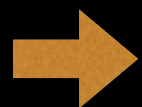
#### MAIN CONTEXT

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!";  
}
```

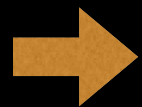
#### MAIN CONTEXT

value	undefined
show_double	undefined

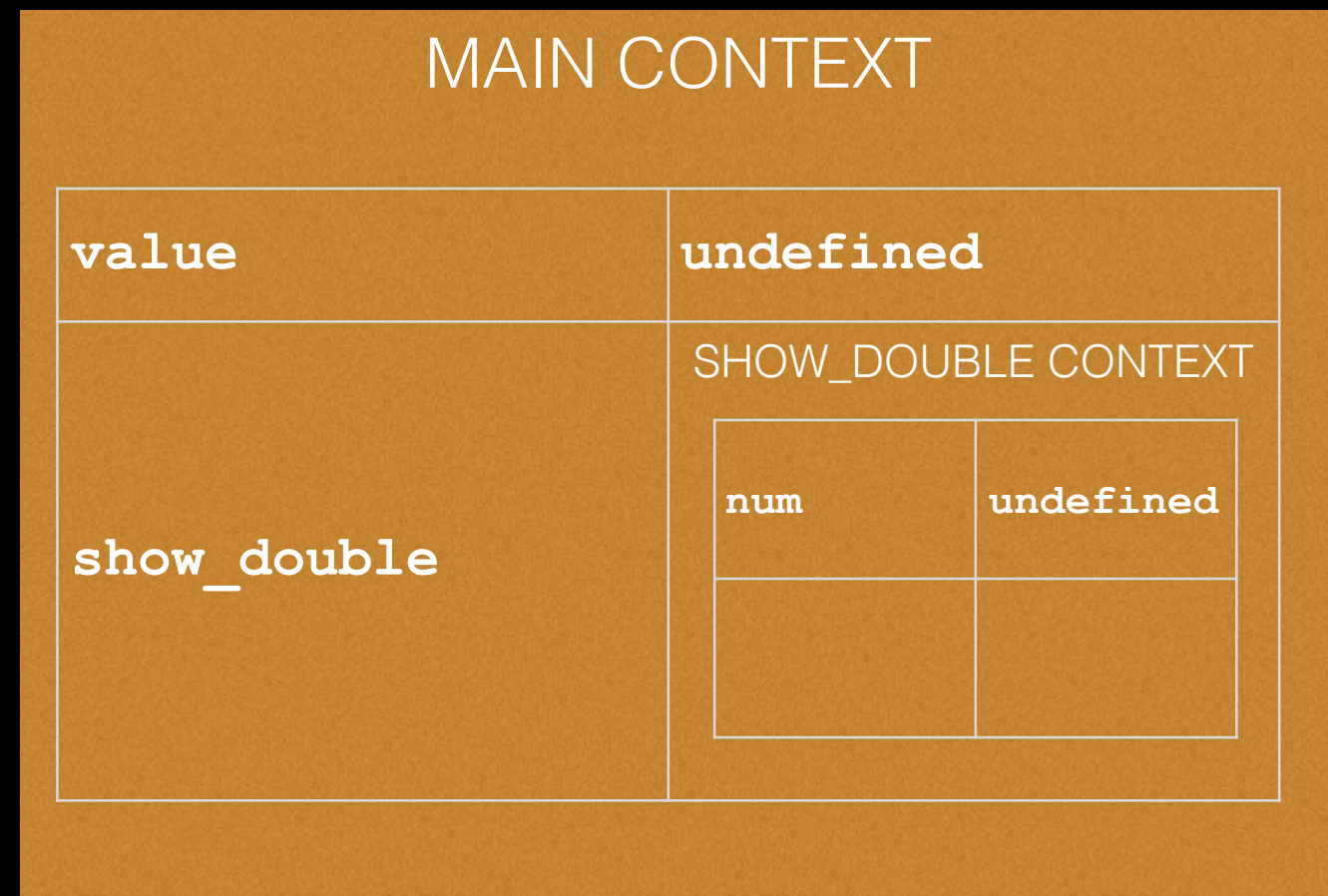
## double.js

### 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

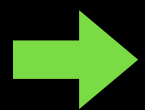
```
var value = 5;  
show_double(value);  
alert(value);  
  
function show_double(num) {  
    var result = num * 2;  
    alert(result);  
    value = "Hi!";  
}
```

#### MAIN CONTEXT

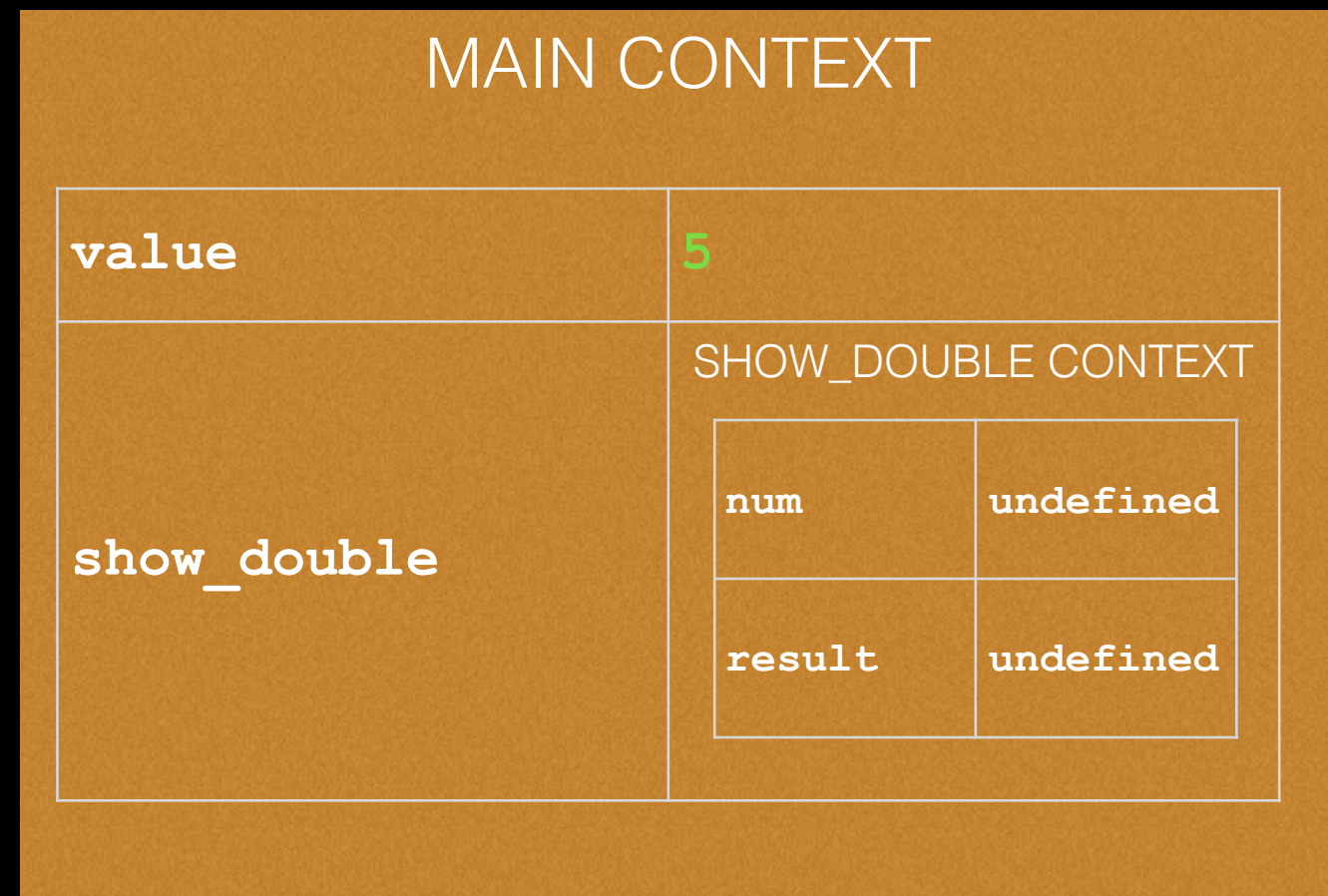
value	undefined				
show_double	SHOW_DOUBLE CONTEXT <table><tr><td>num</td><td>undefined</td></tr><tr><td>result</td><td>undefined</td></tr></table>	num	undefined	result	undefined
num	undefined				
result	undefined				

## double.js

### Executing



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

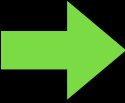


During execution, each line of code is run in turn.  
Variable assignments and function calls change the contexts.



## double.js

### Executing



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

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

#### MAIN CONTEXT

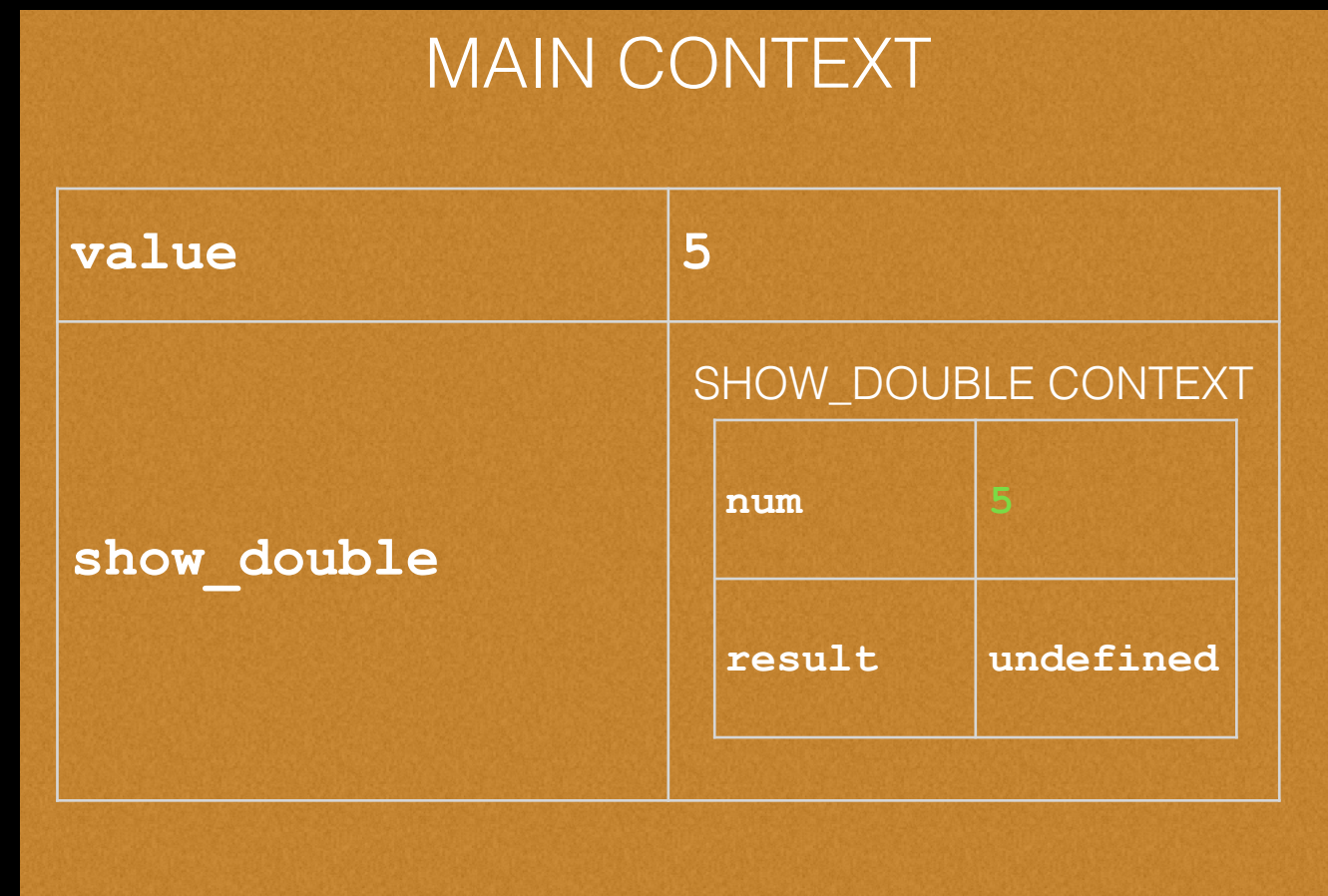
value	5				
show_double	<div>SHOW_DOUBLE CONTEXT<table><tr><td>num</td><td>undefined</td></tr><tr><td>result</td><td>undefined</td></tr></table></div>	num	undefined	result	undefined
num	undefined				
result	undefined				



## double.js

### Executing

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

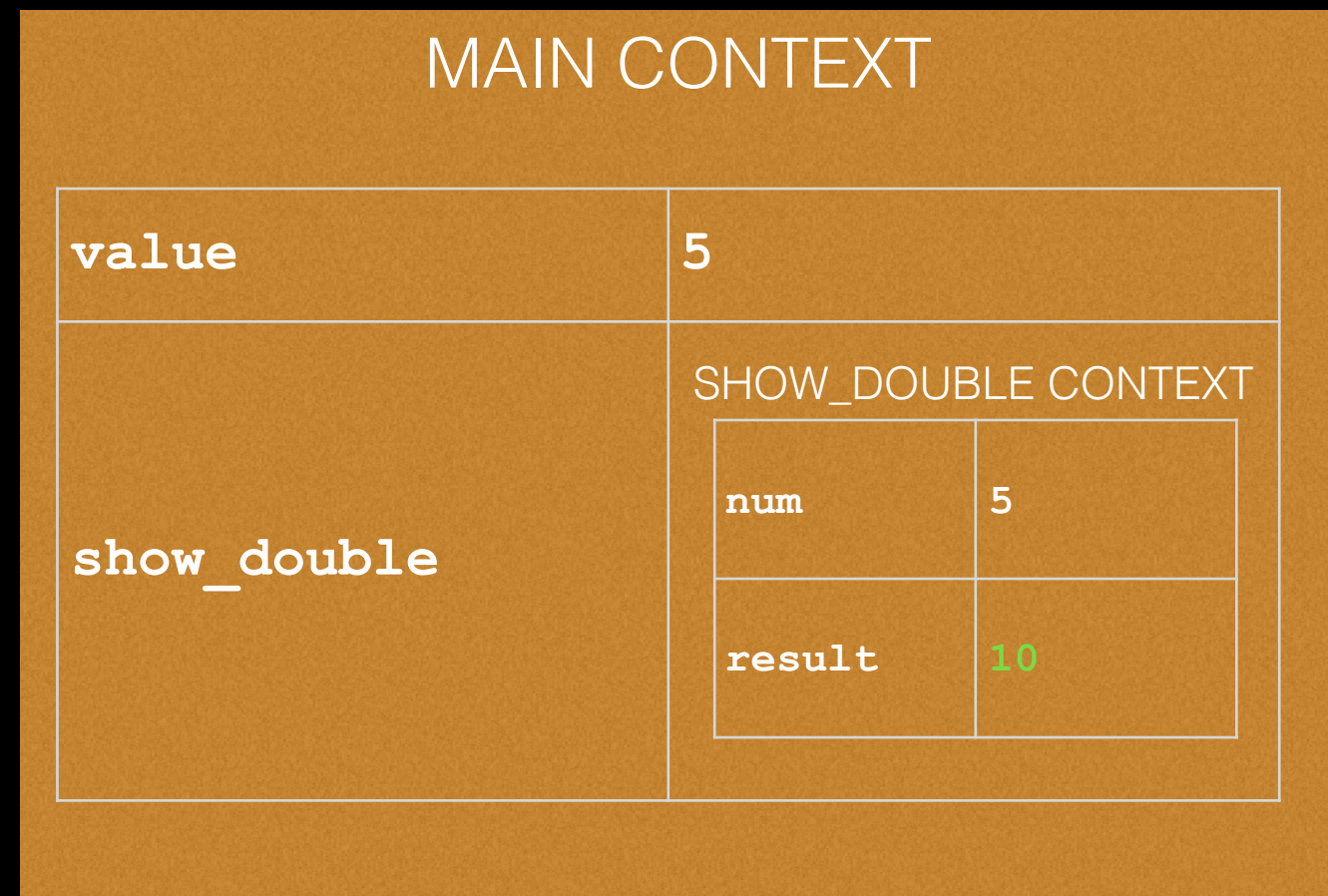


During a function call, values are passed as arguments into the function context.

## double.js

### Executing

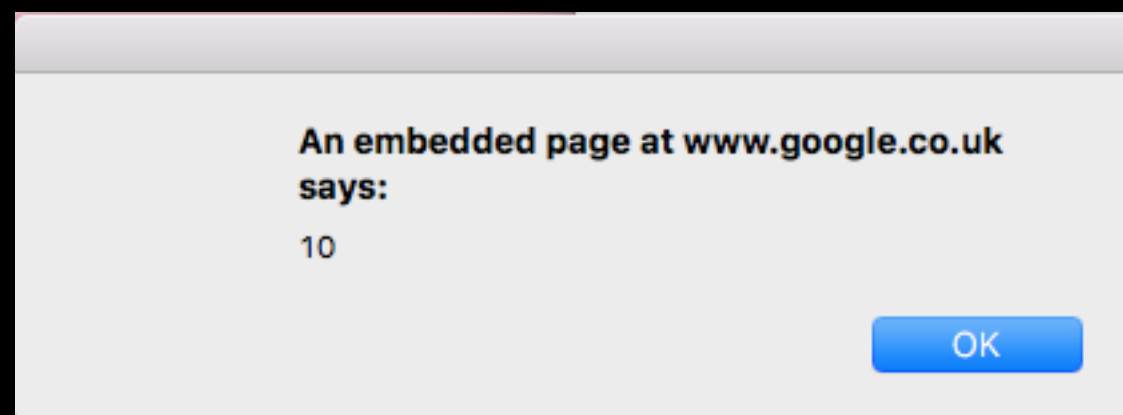
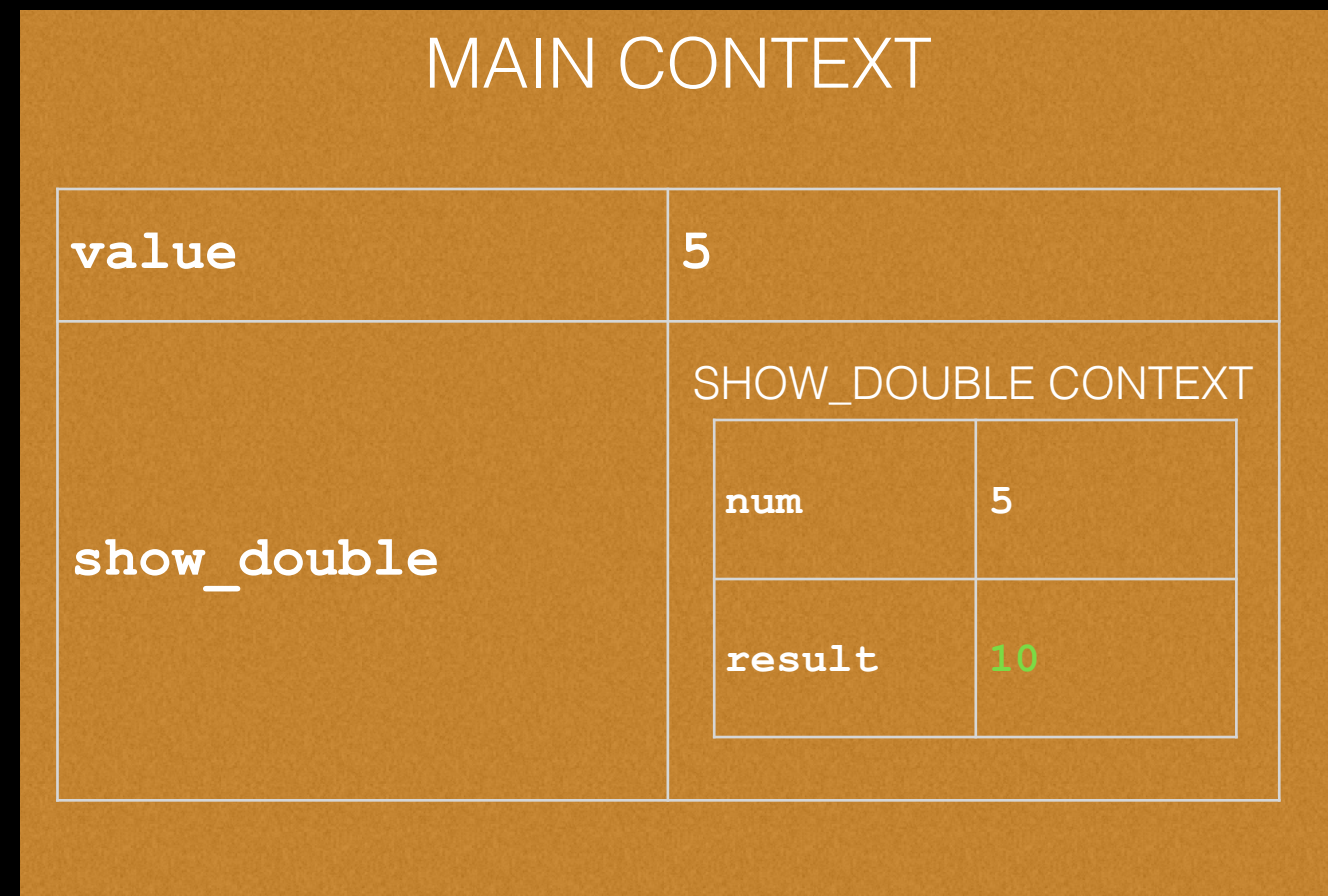
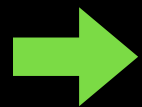
```
var value = 5;  
show_double(value);  
alert(value);  
  
function show_double(num) {  
    var result = num * 2;  
    alert(result);  
    value = "Hi!";  
}
```



## double.js

### Executing

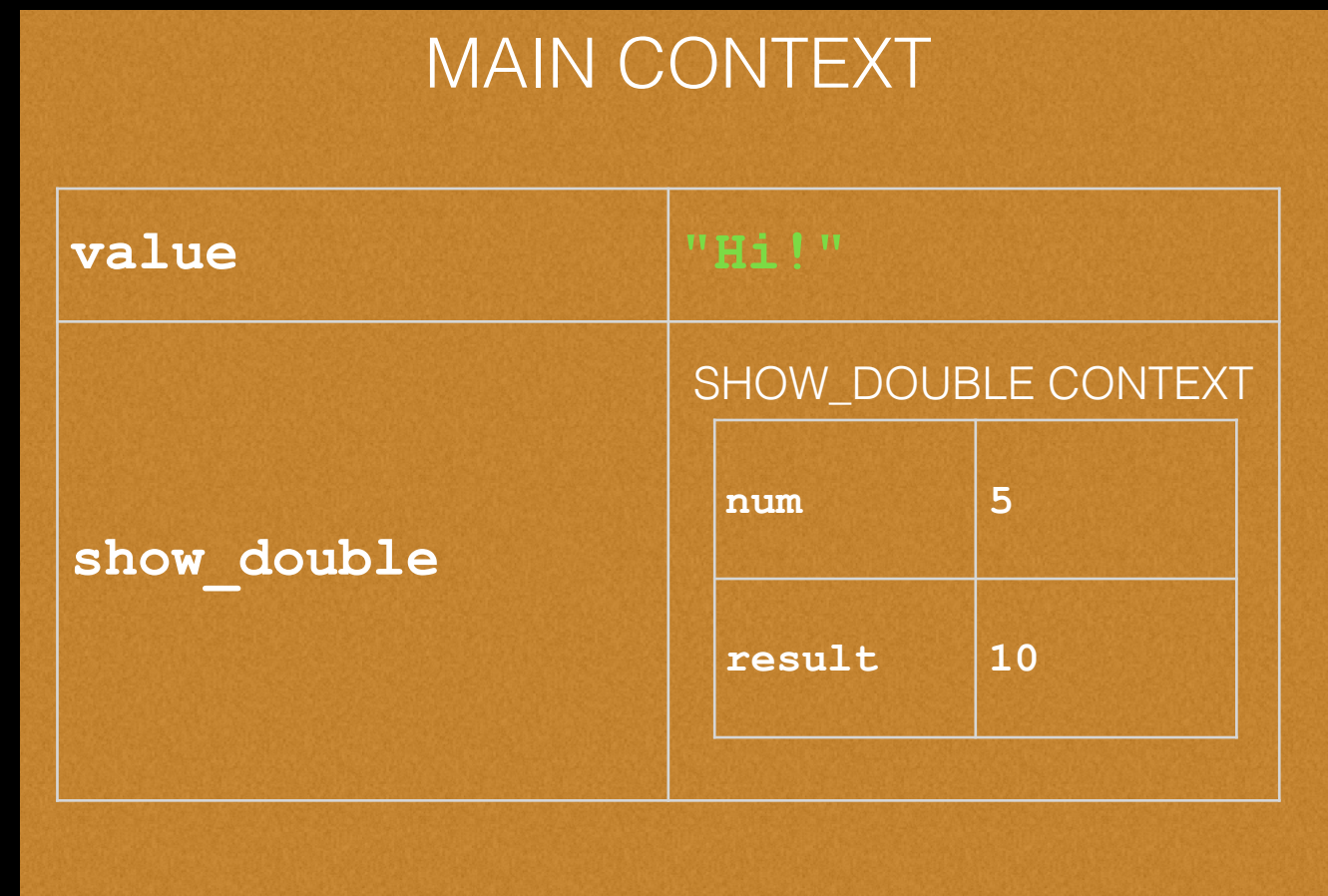
```
var value = 5;  
show_double(value);  
alert(value);  
  
function show_double(num) {  
    var result = num * 2;  
    alert(result);  
    value = "Hi!";  
}
```



## 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	<table><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>						
sum	<table><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>						



## 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;
}
```

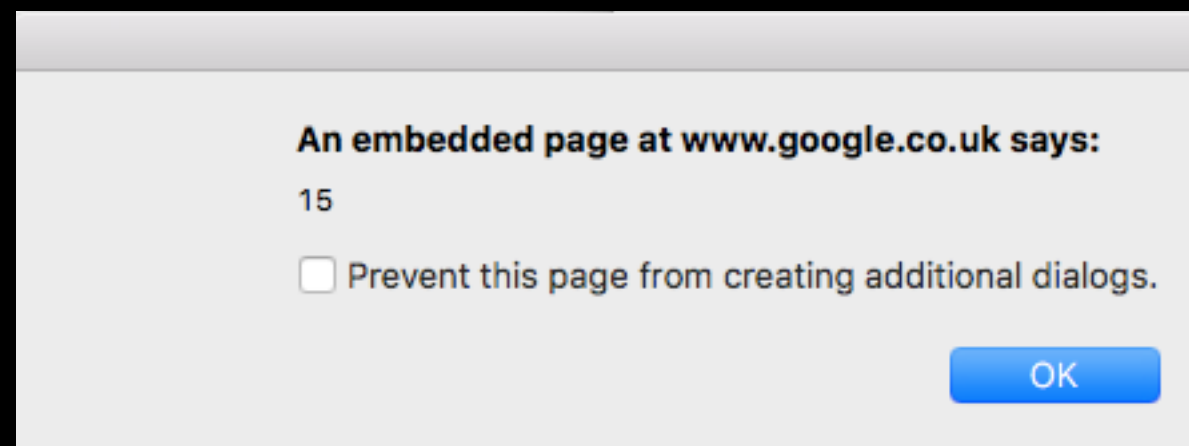
main	first_age	15
	second_age	16
	total	31
sum	num1	15
	num2	16
	result	31

## 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?"	

## 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

## Code





**HW**

**1. Complete Tasks 4-2 & 4-3**