Functions

JavaScript Functions

A JavaScript **function** is a block of code designed to perform a particular task.
The main advantages of using functions:
Code **reuse**: Define the code once, and use it many times.
Use the same code many times with different **arguments**, to produce different results.

> A JavaScript function is executed when "something" invokes, or calls, it.

Defining a Function

To define a JavaScript function, use the **function** keyword, followed by a **name**, followed by a set of **parentheses** ().

The code to be executed by the function is placed inside curly brackets {}.

```
function name() {□
  //code to be executed□
}
```

> Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

Calling a Function

To execute the function, you need to call it.
To call a function, start with the name of the function, then follow it with the arguments in parentheses.
Example:

```
function myFunction() {
  alert("Calling a Function!");
}

myFunction();
//Alerts "Calling a Function!"
```

Try It Yourself

> Always remember to end the statement with a **semicolon** after calling the function.

Calling Functions

Once the function is defined, JavaScript allows you to call it as many times as you want to.

```
function myFunction() {
  alert("Alert box!");
}

myFunction();
//"Alert box!"

// some other code

myFunction();
//"Alert box!"
```

Try It Yourself

> You can also call a function using this syntax: myFunction.call(). The difference is that when calling in this way, you're passing the 'this' keyword to a function. You'll learn about it later.

Function Parameters

Functions can take **parameters**.
Function **parameters** are the names listed in the function's definition.

Syntax:

```
functionName(param1, param2, param3) {
  // some code
}
```

> As with variables, parameters should be given **names**, which are **separated by commas** within the parentheses.

Using Parameters

After defining the parameters, you can use them inside the function.

```
function sayHello(name) {
  alert("Hi, " + name);
}

sayHello("David");
//Alerts "Hi, David"
```

Try It Yourself

This function takes in one parameter, which is called **name**. When calling the function, provide the parameter's value (argument) inside the parentheses.

> Function **arguments** are the real values passed to (and received by) the function.

Function Parameters

You can define a single function, and pass different parameter values (arguments) to it.

```
function sayHello(name) {
  alert("Hi, " + name);
}
sayHello("David");
sayHello("Sarah");
sayHello("John");
```

Try It Yourself

> This will execute the function's code each time for the provided argument.

Multiple Parameters

You can define multiple parameters for a function by **comma-separating** them.

```
function myFunc(x, y) {
  // some code
}
```

> The example above defines the function **myFunc** to take two parameters.

Multiple Parameters

The parameters are used within the function's definition.

```
function sayHello(name, age) {
 document.write( name + " is " + age + " years old.");
}
```

> Function parameters are the names listed in the function definition.

Multiple Parameters

When calling the function, provide the arguments in the same order in which you defined them.

```
function sayHello(name, age) {
 document.write( name + " is " + age + " years old.");
}

sayHello("John", 20)
//Outputs "John is 20 years old."
```

Try It Yourself

> If you pass more arguments than are defined, they will be assigned to an array called arguments. They can be used like this: arguments[0], arguments[1], etc.

Multiple Parameters

After defining the function, you can call it as many times as needed.
JavaScript functions do not check the number of arguments received.

> If a function is called with missing arguments (fewer than declared), the missing values are set to **undefined**, which indicates that a variable has not been assigned a value.

Function Return

A function can have an optional **return** statement. It is used to return a value from the function.

This statement is useful when making calculations that require a result.

> When JavaScript reaches a **return** statement, the function stops executing.

Function Return

Use the **return** statement to return a value.

For example, let's calculate the product of two numbers, and return the result.

```
function myFunction(a, b) {
  return a * b;
}

var x = myFunction(5, 6);
// Return value will end up in x
// x equals 30
```

Try It Yourself

> If you do not return anything from a function, it will return **undefined**.

Function Return

Another example:

```
function addNumbers(a, b) {
  var c = a+b;
  return c;
}
document.write( addNumbers(40, 2) );
//Outputs 42
```

Try It Yourself

> The document.write command outputs the value returned by the function, which is the sum of the two parameters.

The Alert Box

JavaScript offers three types of popup boxes, the **Alert**, **Prompt**, and **Confirm** boxes.

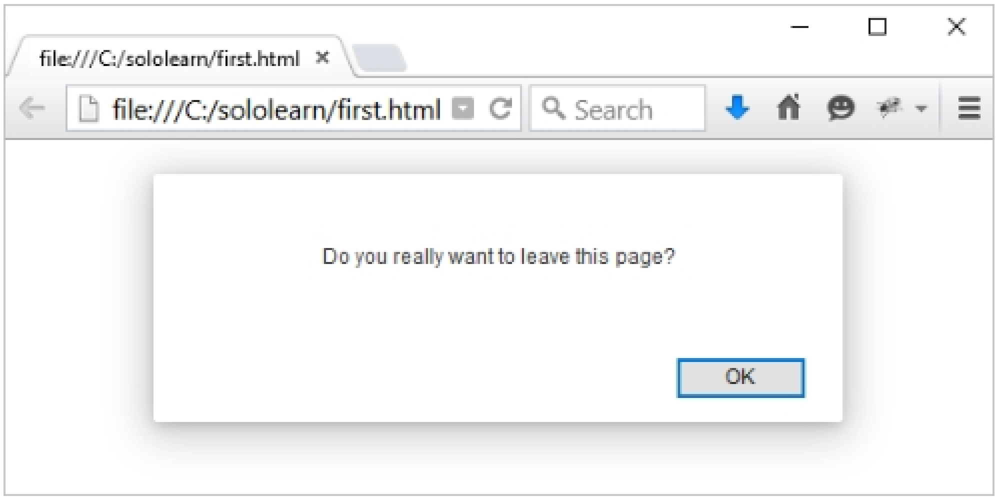
Alert Box

An **alert box** is used when you want to ensure that information gets through to the user.
When an alert box pops up, the user must click OK to proceed.
The **alert** function takes a single parameter, which is the text displayed in the popup box.
Example:

```
alert("Do you really want to leave this page?");
```

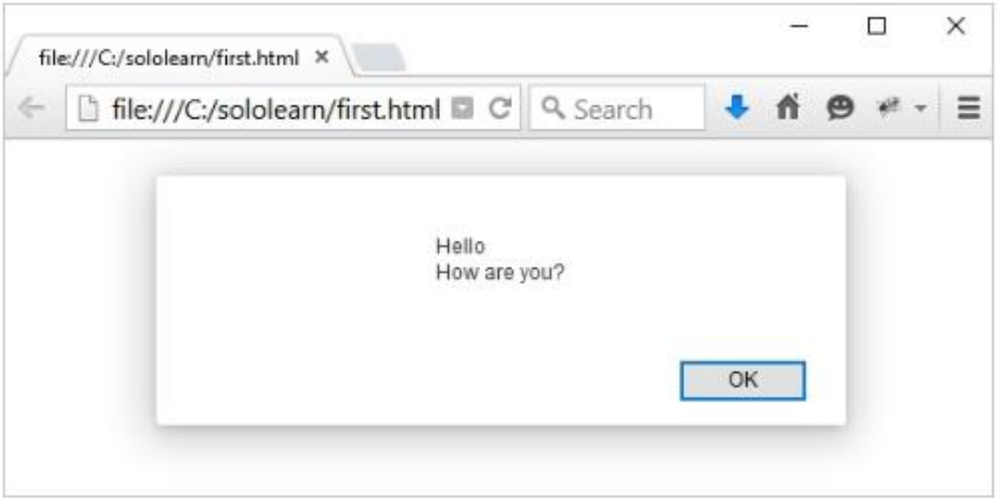
Try It Yourself

Result:


file:///C:/sololearn/first.html
file:///C:/sololearn/first.html
Search
OK

Do you really want to leave this page?

Result:


file:///C:/sololearn/first.html
file:///C:/sololearn/first.html
Search
OK

Hello
How are you?

> Be careful when using alert boxes, as the user can continue using the page only after clicking OK.

Prompt Box

A **prompt box** is often used to have the user input a value before entering a page.
When a prompt box pops up, the user will have to click either OK or Cancel to proceed after entering the input value.
If the user clicks OK, the box **returns the input value**. If the user clicks Cancel, the box returns **null**.

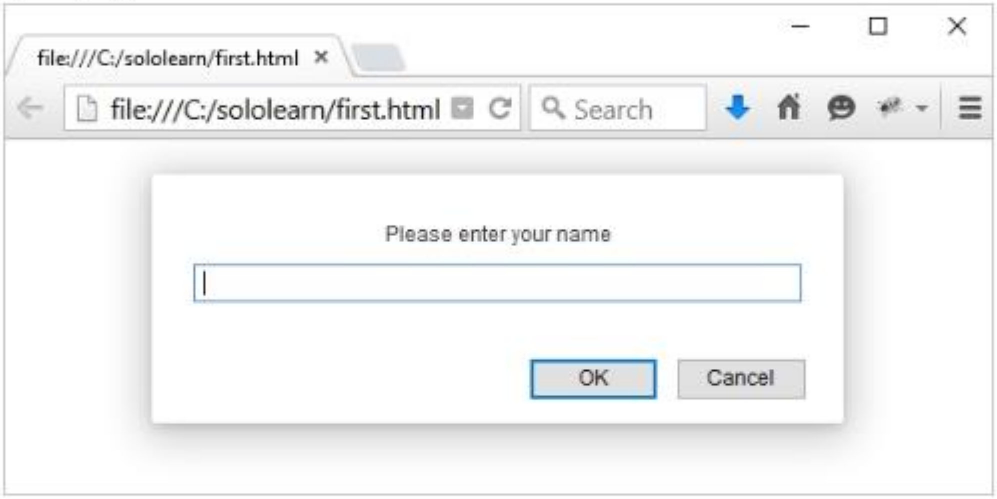
The **prompt()** method takes **two parameters**.
- The first is the label, which you want to display in the text box.
- The second is a default string to display in the text box (optional).

Example:

```
var user = prompt("Please enter your name");
alert(user);
```

Try It Yourself

The prompt appears as:


file:///C:/sololearn/first.html
file:///C:/sololearn/first.html
Search
OK
Cancel

Please enter your name

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. Do not overuse this method, because it prevents the user from accessing other parts of the page until the box is closed.

Confirm Box

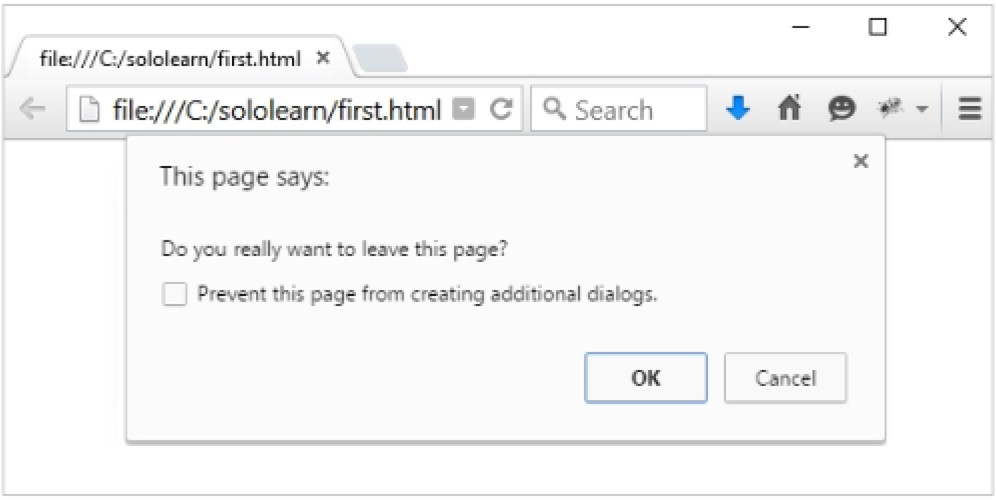
A **confirm box** is often used to have the user verify or accept something.
When a confirm box pops up, the user must click either OK or Cancel to proceed.
If the user clicks OK, the box returns **true**. If the user clicks Cancel, the box returns **false**.

Example:

```
var result = confirm("Do you really want to leave this page?");
if (result == true) {
  alert("Thanks for visiting");
}
else {
  alert("Thanks for staying with us");
}
```

Try It Yourself

Result:

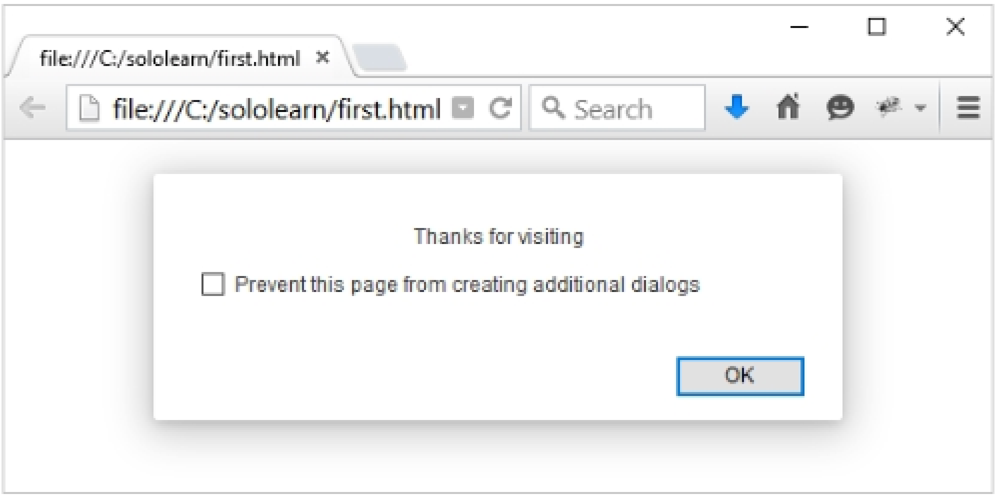

file:///C:/sololearn/first.html
file:///C:/sololearn/first.html
Search
OK
Cancel

This page says:

Do you really want to leave this page?

□ Prevent this page from creating additional dialogs.

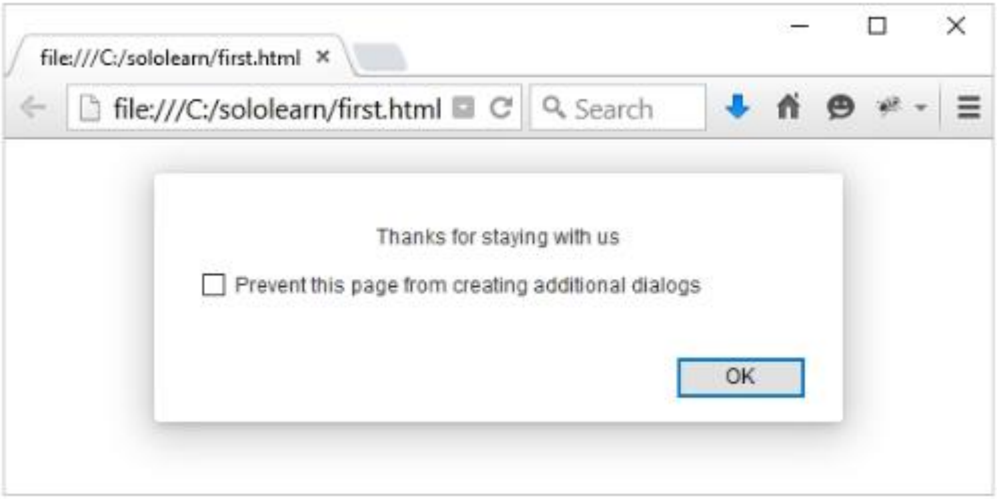
The result when the user clicks **OK**:


file:///C:/sololearn/first.html
file:///C:/sololearn/first.html
Search
OK

Thanks for visiting

□ Prevent this page from creating additional dialogs

The result when the user clicks **Cancel**:


file:///C:/sololearn/first.html
file:///C:/sololearn/first.html
Search
OK

Thanks for staying with us

□ Prevent this page from creating additional dialogs

Do not overuse this method, because it also prevents the user from accessing other parts of the page until the box is closed.

End.