Functions

- Functions have inputs and outputs
- The inputs are passed into the function and are known as **arguments** or **parameters**
- Think of a function as a “black box” which performs an operation

Defining Functions

- The most common way to define a function is with the function statement.
- The function statement consists of the function keyword followed by the name of the function, a comma-separated list of parameter names in parentheses, and the statements which contain the body of the function enclosed in curly braces

Example: Function

```
function square(x)
{return x*x;}

z = 3;
sqr_z = square(z);
```

Name of Function: square

Input/Argument: x

Output: x*x

Example: Function

```
function sum_of_squares(num1,num2)

    {return (num1*num1) + (num2*num2);}
```

```
function sum_of_squares(num1,num2)

    {return (square(num1) + square(num2));}
```

The return Statement

- Used to specify the value that is returned from the function.
- Functions that are going to return a value must use the return statement.

Example

Function to return product of two numbers (a and b):

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(product(4,3));
</script>
</body>
</html>
```

The Lifetime of JavaScript Variables

- A variable declared, using "var", within a function, is only accessed within that function
- It is destroyed when you exit the function
- These variables are called **local variables**
- Declaring local variables with the same name in different functions is allowed since each is recognized only by the function in which it is declared.
- Variables declared outside a function are accessible to all the functions on your page can access it.
- Their lifetime starts when they are declared, and ends when the page is closed.

JavaScript Popup Boxes

- JavaScript has three kind of popup boxes:

1. Alert box
2. Confirm box
3. Prompt box

Alert Box

- Often used if you want to make sure information comes through to the user.
- When an alert box pops up, the user will have to click "OK" to proceed

Syntax : alert("sometext");

Example

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
alert("I am an alert box!");
}
</script>
</head>
<body>
<input type="button" onclick="show_alert()" value="Show alert box" />
</body>
</html>
```

Confirm Box

- Often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to 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.

Syntax: confirm("sometext");

Example

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button");
if (r==true)
{
alert("You pressed OK!");
}
else
{
alert("You pressed Cancel!");
}
}
</script>
```

Example cont..

```
</head>
<body>
<input type="button"
onclick="show_confirm()" value="Show
confirm box" />
</body>
</html>
```

Prompt Box

- Often used if you want the user to 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 an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax : prompt("sometext","defaultvalue");

Example

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
{
document.write("Hello " + name + "! How are you today?");
}
}
</script>
```

Example cont..

```
</head>
<body>
<input type="button" onclick="show_prompt()" value="Show prompt box" />
</body>
</html>
```

JavaScript Events

- By using JavaScript, we have the ability to create dynamic web pages.
- Events are actions that can be detected by JavaScript.
- Every element on a web page has certain events which can trigger a JavaScript.
- For example, we can use the onClick event of a button element to indicate what a function will run when a user clicks on the button.
- We define the events in the HTML tags.

Examples of events:

- Mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input field in an HTML form
- Submitting an HTML form
- A keystroke
- Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

Example

```
<html>
<head>
<script type="text/javascript">
function displayDate()
{
document.getElementById("demo").innerHTML=Date();
}
</script>
</head>
<body>
<h1>My First Web Page</h1>
<p id="demo"></p>
<button type="button" onclick="displayDate()">Display Date</button>
</body>
</html>
```

onLoad and onUnload

- These events are triggered when the user enters or leaves the page.
- The onLoad event:
  - Used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

onLoad and onUnload events:

- Are also often used to deal with cookies that should be set when a user enters or leaves a page.
- Example, a popup asking for the user's name upon his first arrival to your page.
- The name is then stored in a cookie.
- Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John!".

onFocus, onBlur and onChange

- Used in combination with validation of form fields.

Example of how to use the onChange event.

- The checkEmail() function will be called whenever the user changes the content of the field:

<input type="text" size="30" id="email"

onchange="checkEmail()">

onSubmit

- Used to validate ALL form fields before submitting it

Example

<form method="post" action="xxx.htm" onsubmit="return checkForm()">

- The checkForm() function will be called when the user clicks the submit button in the form.
- The function checkForm() returns either true or false.
- If it returns true the form will be submitted, otherwise the submit will be cancelled:

onMouseOver

- The onmouseover event can be used to trigger a function when the user mouses over an HTML element.

JavaScript - Catching Errors

- Expected errors when browsing Web pages on the internet

Runtime error and asking "Do you wish to debug?".

- Error message like id useful for developers but not for users.
- When users see errors, they often leave the Web page.

JavaScript Try...Catch Statement

- Allows you to test a block of code for errors.
- The try block contains the code to be run
- The catch block contains the code to be executed if an error occurs.

Syntax

```
try
{
//Run some code here
}
catch(err)
{
//Handle errors here
}
```

Note that try...catch is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Example

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try
{
adddlert("Welcome guest!");
}
```

Example cont..

```
catch(err)
{
txt="There was an error on this page.\n\n";
txt+="Error description: " + err.description + "\n\n";
txt+="Click OK to continue.\n\n";
alert(txt);
}
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

Explanation

- The example is supposed to alert "Welcome guest!" when the button is clicked.
- However, there's a typo in the message() function. alert() is misspelled as adddlert(). A JavaScript error occurs.
- The catch block catches the error and executes a custom code to handle it.
- The code displays a custom error message informing the user what happened

Learners' activity

- Create a confirm box to display a custom message telling users they can click OK to continue viewing the page or click Cancel to go to the homepage. If the confirm method returns false, the user clicked Cancel, and the code redirects the user. If the confirm method returns true, the code does nothing

JavaScript Throw Statement

- Allows the developer to create an exception.
- It can control program flow and generate accurate error messages when used together with the try...catch statement

  Syntax: throw exception

- Here the exception can be a string, integer, Boolean or an object.
- throw is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Example

```
<html>
<body>
<script type="text/javascript">
var x=prompt("Enter a number between 0 and 10:","");
try
{
if(x>10)
{
throw "Err1";
}
else if(x<0)
{
throw "Err2";
}
else if(isNaN(x))
{
throw "Err3";
}
}
catch(er)
{
if(er=="Err1")
```

Example cont…

```
{
alert("Error! The value is too high");
}
if(er=="Err2")
{
alert("Error! The value is too low");
}
if(er=="Err3")
{
alert("Error! The value is not a number");
}
}
</script>
</body>
</html>
```