**Functions:**

It allows you to store a piece of code that does a single task inside a defined block, and then call that code whenever you need it using a single short command

**Built-in browser functions:**

let myText = 'I am a string';

let newString = myText.**replace**('string', 'sausage');

console.log(newString);

let myArray = ['I', 'love', 'chocolate', 'frogs'];

let madeAString = myArray.**join**(' ');

console.log(madeAString);

let myNumber = Math.random();

**Build your own function:**

function displayMessage(msgText, msgType) {

}

**Calling the function:**

displayMessage('Woo, this is a different message!');

function random(number) {

return Math.floor(Math.random()\*number);

}

random(1);

**Invoking functions:**

function myFunction() {

alert('hello');

}

myFunction()

// calls the function once

**Anonymous functions**

function() {

alert('hello');

}

This is called an anonymous function — it has no name! It also won't do anything on its own.

You generally use an anonymous function along with an event handler

var myButton = document.querySelector('button');

myButton.onclick = function() {

alert('hello');

}

You can also assign an anonymous function to be the value of a variable, for example:

var myGreeting = function() {

alert('hello');

}

myGreeting();

**Function parameters**

Some functions require parameters to be specified when you are invoking them — these are values that need to be included inside the function parentheses,

let myText = 'I am a string';

let newString = myText.**replace**('string', 'sausage');

let myArray = ['I', 'love', 'chocolate', 'frogs'];

let madeAString = myArray.**join**(' ');

// returns 'I love chocolate frogs'

let madeAString = myArray.**join**(',');

// returns 'I,love,chocolate,frogs'

**Introduction to events:**

Events are actions or occurrences that happen in the system you are programming, which the system tells you about so you can respond to them in some way if desired

For example, if the user clicks a button on a webpage, you might want to respond to that action by displaying an information box.

**A simple example**

<button>Change color</button>

var btn = document.querySelector('button');

function random(number) {

return Math.floor(Math.random()\*(number+1));

}

btn.onclick = function() {

var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) + ')';

document.body.style.backgroundColor = rndCol;

}

**Inline event handlers — don't use these**

<button onclick="bgChange()">Press me</button>

function bgChange() {

var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) + ')';

document.body.style.backgroundColor = rndCol;

}

**addEventListener() and removeEventListener()**

var btn = document.querySelector('button');

function bgChange() {

var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) + ')';

document.body.style.backgroundColor = rndCol;

}

btn.addEventListener('click', bgChange);

btn.removeEventListener('click', bgChange); // to remove the events.

**Event objects**

Sometimes inside an event handler function, you might see a parameter specified with a name such as event, evt, or simply e.

This is called the event object, and it is automatically passed to event handlers to provide extra features and information.

function bgChange(e) {

var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) + ')';

e.target.style.backgroundColor = rndCol;

console.log(e);

}

btn.addEventListener('click', bgChange)

e.g.

var divs = document.querySelectorAll('div');

for (var i = 0; i < divs.length; i++) {

divs[i].onclick = function(e) {

e.target.style.backgroundColor = bgChange();

}

}

**Preventing default behavior**

Sometimes, you'll come across a situation where you want to stop an event doing what it does by default.

e.g.

<form>

<div>

<label for="fname">First name: </label>

<input id="fname" type="text">

</div>

<div>

<label for="lname">Last name: </label>

<input id="lname" type="text">

</div>

<div>

<input id="submit" type="submit">

</div>

</form>

<p></p>

var form = document.querySelector('form');

var fname = document.getElementById('fname');

var lname = document.getElementById('lname');

var submit = document.getElementById('submit');

var para = document.querySelector('p');

form.onsubmit = function(e) {

if (fname.value === '' || lname.value === '') {

e.preventDefault();

para.textContent = 'You need to fill in both names!';

}

}

**Handling text — strings in JavaScript**

let string = 'The revolution will not be televised.';

string;

**Escaping characters in a string**

let bigmouth = 'I\'ve got no right to take my place...';

bigmouth;

**Concatenating strings**

let one = 'Hello, ';

let two = 'how are you?';

let joined = one + two;

joined;

<button>Press me</button>

const button = document.querySelector('button');

button.onclick = function() {

let name = prompt('What is your name?');

alert('Hello ' + name + ', nice to see you!');

}

**Finding the length of a string**

let browserType = 'mozilla';

browserType.**length**;

**Retrieving a specific string character**

browserType[0];

browserType[browserType.**length**-1];

**Finding a substring inside a string and extracting it**

browserType.indexOf('zilla');

You could use this to find all instances of strings that don't contain the substring 'mozilla', or do,

if(browserType.indexOf('mozilla') !== -1) {

// do stuff with the string

}

When you know where a substring starts inside a string, and you know at which character you want it to end, slice() can be used to extract it.

let browserType = 'mozilla';

browserType.slice(0,3);

This returns "moz"

Changing case

let radData = 'My NaMe Is MuD';

radData.**toLowerCase**();

radData.**toUpperCase**();

**Updating parts of a string**

You can replace one substring inside a string with another substring using the replace()

browserType.replace('moz','van');

let myData = 'Manchester,London,Liverpool,Birmingham,Leeds,Carlisle';

let myArray = myData.**split**(',');

myArray;

myArray.**length**;

myArray[0]; // the first item in the array

myArray[1]; // the second item in the array

myArray[myArray.length-1]; // the last item in the array

let myNewString = myArray.**join**(',');

myNewString;

**Adding and removing array items**

let myArray = ['Manchester', 'London', 'Liverpool', 'Birmingham', 'Leeds', 'Carlisle'];

First of all, to add or remove an item at the end of an array we can use push() and pop() respectively.

myArray.push('Cardiff');

myArray.push('Bradford', 'Brighton');

myArray.pop();

unshift() and shift() work in exactly the same way as push() and pop(), respectively, except that they work on the beginning of the array, not the end.

myArray.unshift('Edinburgh');

let removedItem = myArray.shift();

**Object-oriented JavaScript**

**Object basics**

An object is a collection of related data and/or functionality (which usually consists of several variables and functions — which are called properties and methods when they are inside objects.)

const person = {};

const person = {

name: ['Bob', 'Smith'],

age: 32,

gender: 'male',

interests: ['music', 'skiing'],

bio: function() {

alert(this.name[0] + ' ' + this.name[1] + ' is ' + this.age + ' years old. He likes ' + this.interests[0] + ' and ' + this.interests[1] + '.');

},

greeting: function() {

alert('Hi! I\'m ' + this.name[0] + '.');

}

};

person.name

person.name[0]

person.age

person.interests[1]

person.bio()

person.greeting()

**What is "this"?**

The this keyword refers to the current object the code is being written inside — so in this case this is equivalent to person.

const person1 = {

name: 'Chris',

greeting: function() {

alert('Hi! I\'m ' + this.name + '.');

}

}

const person2 = {

name: 'Brian',

greeting: function() {

alert('Hi! I\'m ' + this.name + '.');

}

}

**What is Object-oriented programming (OOP)?**

The basic idea of OOP is that we use objects to model real world things that we want to represent inside our programs, and/or provide a simple way to access functionality that would otherwise be hard or impossible to make use of.

Objects can contain related data and code, which represent information about the thing you are trying to model, and functionality or behavior that you want it to have.

Object data (and often, functions too) can be stored neatly (the official word is encapsulated) inside an object package (which can be given a specific name to refer to, which is sometimes called a namespace),

Person

FirstName/LastName

Age

Phone

Email

Greeting()

**Constructors and object instances**

JavaScript uses special functions called constructor functions to define and initialize objects and their features.

function createNewPerson(name) {

var obj = {};

obj.name = name;

obj.greeting = function() {

alert('Hi! I\'m ' + obj.name + '.');

};

return obj;

}

var salva = createNewPerson('Salva');

salva.name;

salva.greeting();

**Replace your previous function with the following:**

function Person(name) {

this.name = name;

this.greeting = function() {

alert('Hi! I\'m ' + this.name + '.');

};

}

var person1 = new Person('Bob');

var person2 = new Person('Sarah');

person1.name

person1.greeting()

person2.name

person2.greeting()

e.g.

function Person(first, last, age, gender, interests) {

this.name = {

first : first,

last : last

};

this.age = age;

this.gender = gender;

this.interests = interests;

this.bio = function() {

alert(this.name.first + ' ' + this.name.last + ' is ' + this.age + ' years old. He likes ' + this.interests[0] + ' and ' + this.interests[1] + '.');

};

this.greeting = function() {

alert('Hi! I\'m ' + this.name.first + '.');

};

}

var person1 = new Person('Bob', 'Smith', 32, 'male', ['music', 'skiing']);

person1['age']

person1.interests[1]

person1.bio()

**Other ways to create object instances**

The Object() constructor

You can use the Object() constructor to create a new object

var person1 = new Object();

person1.name = 'Chris';

person1['age'] = 38;

person1.greeting = function() {

alert('Hi! I\'m ' + this.name + '.');

};

You can also pass an object literal to the Object() constructor as a parameter

var person1 = new Object({

name: 'Chris',

age: 38,

greeting: function() {

alert('Hi! I\'m ' + this.name + '.');

}

});

**Using the create() method**

You can create a new object based on any existing object

var person2 = Object.create(person1);

person2.name

person2.greeting()

**Object prototypes**

Prototypes are the mechanism by which JavaScript objects inherit features from one another.

JavaScript is often described as a prototype-based language — to provide inheritance, objects can have a prototype object, which acts as a template object that it inherits methods and properties from, and so on.

function Person(first, last, age, gender, interests) {

// property and method definitions

this.first = first;

this.last = last;

//...

}

var person1 = new Person('Bob', 'Smith', 32, 'male', ['music', 'skiing']);

You can check out existing prototype properties for yourself :

Person.prototype

person1.constructor

**Modifying prototypes**

Person.prototype.farewell = function() {

alert(this.name.first + ' has left the building. Bye for now!');

};

person1.farewell();

e.g.

// Constructor with property definitions

function Test(a, b, c, d) {

// property definitions

}

// First method definition

Test.prototype.x = function() { ... };

// Second method definition

Test.prototype.y = function() { ... };

// etc.

**Prototypal inheritance**

function Person(first, last, age, gender, interests) {

this.name = {

first,

last

};

this.age = age;

this.gender = gender;

this.interests = interests;

};

Person.prototype.greeting = function() {

alert('Hi! I\'m ' + this.name.first + '.');

};

function Teacher(first, last, age, gender, interests, subject) {

Person.call(this, first, last, age, gender, interests);

this.subject = subject;

}

**Setting Teacher()'s prototype and constructor reference**

Teacher.prototype = Object.create(Person.prototype);

Teacher.prototype.greeting = function() {

var prefix;

if (this.gender === 'male' || this.gender === 'Male' || this.gender === 'm' || this.gender === 'M') {

prefix = 'Mr.';

} else if (this.gender === 'female' || this.gender === 'Female' || this.gender === 'f' || this.gender === 'F') {

prefix = 'Mrs.';

} else {

prefix = 'Mx.';

}

alert('Hello. My name is ' + prefix + ' ' + this.name.last + ', and I teach ' + this.subject + '.');

};

var teacher1 = new Teacher('Dave', 'Griffiths', 31, 'male', ['football', 'cookery'], 'mathematics');

teacher1.name.first;

teacher1.interests[0];

teacher1.bio();

teacher1.subject;

teacher1.greeting();

teacher1.farewell();