Notes - Class 6

Object Oriented Programming

representing things as objects

an object has:

- properties: colour, bodywork, model, number of seats, engine size

Some properties don't change and are present in all cars (car has four wheels, has windows, has lights, has seats)
other properties are specific to only some cars (sat nav, tv, aircon). Think of a very basic car as a prototype. And all other cars will be built on top of that basic, they are what is called "instances" of a prototype with their own variants.

- functionality, behaviours: it drives, it reverses, indicates, it plays music, starting/stopping engine

Lets think of a toast:

Bread:
- properties: grain, crusty/soft, size, burntness
- methods: smells (lovely), tastes (even better), absorbs butter, fills us up

Toaster:
- properties:
    colour
    2-4 slices
    wattage
    timer

- methods (functions):
    switches on
    toasts
    cancels
    checks temperature

we worked through the process of making a toast

1. wake up2. get out of bed3. slice bread4. pop your slice in the toaster5. turn the toaster on6. check timer 6A if its right move to step 8 (skip 7) 6B if its not right move to step 7
7. adjust timer
8. turn on the toaster
9. wait till toaster stops
10. check the readiness
    10A. if its ready, move to step 11
    10B. if its not ready, repeat from step 4
11. butter and eat

This is a basic algorithm, a process of getting a certain result broken down to simple steps. In the described set of instructions we have conditionals (our if statements) which define a different outcome depending on the fulfilment of one or the other condition.
we also have a loop, i.e. an action (or set of actions) will continue repeating until condition calling that action is no longer true, in our case that condition is our check whether the toast is not ready. Until its not ready, i.e. condition "not ready" is true we will keep going through the steps 4-10

Computational thinking
in a layman's terms: computational thinking is about finding solutions to problems by breaking them down into smallest possible steps which lead to a desired outcome

JS is a programming language, as opposed to HTML, which is a markup language
program executes linearly one line of code thing at a time

JavaScript console:
you have one built in to Chrome and Firefox browsers, it will be one of the tabs in the developer tools.

for example, we can use console to output result of various mathematical operations that JavaScript can perform

console.log("some text");
will output

```
console.log("some text")
some text
```

we will be using swimmingpools.felixcohen.co.uk for practice using console we will be using [illegible] for practice using console

console.log is used a lot ad simple tool for debugging, ie understanding what went wrong if your code does't work, which is very important.

```
var name = prompt("what is your name?");
```

when above is run the browser will display a dialogue box with the input field, where you can enter your name which will the be saved in variable "name"
after the name entry was made into the dialogue box we can check that name variable stored your entered value, we can type by typing "name" in the console and hit enter:

```
name
"Jay"
```

next step: type the following

```
alert("hi there " + name)
undefined
```

so here we are using our variable name we set to concatenate (join) string "hi there " with string value stored under name variable. (Bear in mind that the entry from a dialogue box is stored as string)

Create another dialogue box
var age = prompt("how old are you, " + name + "?")

alert("hi "+ name + ", you are "+age+"years old")

arrow up will bring up text typed in previously in the console

we use plus for string concatenation, i.e. joining strings

it is a convention to use camel case for naming variable in JavaScript: start with lowercase and every new word should start with a capital letter, you can't have spaces in the variable names

```
var year
undefined
2013 - parseInt(age)
```

jQuery
a library that simplifies some low level JavaScript methods. It makes it more straightforward to manipulate the HTML elements. So everything you can do with jQuery, you can do with pure JS but it will be much more painful.

this is how we include jQuery library in the html document

<script type="text/javascript" src="js/jquery-1.8.0.min.js"></script>

script tags are included at the end of your html document to ensure that our DOM structure (your HTML) is fully present in your browser before we try to manipulate it

we went through the fruithover.js file in the exercise "ex adding scripts", this was mainly to practice the inclusion of the jQuery into your html file.

this exercise uses jQuery to create fade in effect on hover.

$ sign represents jQuery

We played around in the console to see how to manipulate elements usign jQuery:

```
$('h1').slideUp()
[
<h1 style="overflow: hidden; height: 36.81543766775631px; padding-top: 0px; margin-top: 21.33305414211514px;"]]>Fruit
Gallery</h1]]>
]
```

```
$('h1').slideDown()
[
<h1 style="display: block; overflow: hidden; height: 1px; padding-top: 0px; margin-top: 0px;"]]>Fruit Gallery</h1]]>
]
```

```
$('h1').slideToggle(5000)
[
<h1 style="display: block; overflow: hidden; height: 37px; padding-top: 0px; margin-top: 21.440000534057617px;"]]>Fruit
Gallery</h1]]>
]
```

```
$('h1').slideToggle(5000)
[
<h1 style="display: block; overflow: hidden; height: 1px; padding-top: 0px; margin-top: 0px;"]]>Fruit Gallery</h1]]>
]
```

****

Another practice directly in the console

```
var age = prompt("what is your age")
undefined
age
"27"

var year
undefined

year = 2013 - age
1986

parseInt(age)
27
```

Data types & variables

Variable is a bucket for data.
It could be a string, a number or a boolean, it doesn't care what it stores
programming language deals in variables, rather than actual text strings or numbers

Variable:

we use variable to notate our

Declaration/initialisation:
    var age;

Assignment to a variable declared previously:
    age = 21;

Declaration and assignment in one step:
    var age = 21;

Once the variable has been declared we can always reassign a different value to it:

declaration and assignment:
    var a = "value"
    if we console.log(a) we would get "value" returned

re-assignment:
    a = "new value"
    this time around if we console.log(a) we would get "new value" returned to us

Data types:

String
    stores text
    strings require quotes around the body of the string

    NB: escaping: backslash tells the program to take the next character literally

Number
    represents numerical data
    integers: 4
    floats: 3.2

Boolean
    variable whose value is either **true** or **false**

Object

An object is just a special kind of data, with **properties** and **methods**.

generally speaking objects are used everywhere in programming because they are great for organising information that describes our real life objects, problems etc.

Let's go back to our car example when we were trying to list its properties and methods, this is how the object literal notation for our car object would look like:

var car = {make: "ford", model: "Fiesta", cc: 100, colour: "blue"};

where:

- "car" is the variable name that stores our object, so anytime we need to refer to our object we can use car
- "make", "model", "cc", "colour" - we can refer to them as keys
- "ford", "Fiesta", "100", "blue" - are the values for corresponding keys

In the example above the keys of our car object happen to be the properties of the object. But they could also have been methods

Properties are variables created inside an object and methods are functions created inside an object

```
var myObject = {
    iAm : 'an object',
    whatAmI : function(){
        alert('I am ' + this.iAm);
    }
}
```

we have created a property 'iAm' which contains a string value that is used in our objects method 'whatAmI' which alerts a message.
Accessing Object Properties
Properties are the values associated with an object.
The syntax for accessing the property of an object is:
objectName.propertyName

To use a property first you type what object it belongs to – so in our example above this woulbe **myObject** –and then to reference its internal properties, you put a full stop and then the name of the
property so it will eventually look like myObject.iAm (this will return 'an object').

Following example uses the length property of the String object (i.e. in JavaScript all strings are instances of a String object, therefore they will have this property) to find the length of a string:

```
var message="Hello World!";var x=message.length;
```
The value of x, after execution of the code above will be:
```
12
```

Accessing Objects Methods
Methods are the actions that can be performed on objects.
You can call a method with the following syntax:
objectName.methodName()

so in the case of our example with myObject calling method will look the same except to execute the method, as with any function, you must put parenthesis after it; otherwise you will just be returning a reference to the function
and not what the function actually returns. So it will look like myObject.whatAmI() (this will alert 'I am an object').

This example uses the toUpperCase() method of the String object, to convert a text to uppercase:

```
var message="Hello world!";var x=message.toUpperCase();
```
The value of x, after execution of the code above will be:
```
HELLO WORLD!
```

If...else

if
Executes a statement if a specified condition is true. If the condition is false, another statement can be executed.

```
if (condition) {
   statements1
} else {
   statements2
}
```

for our condition we would usually use one of the logical operators, condition is essentially comparing one with another and performing corresponding on the actions accordingly

please also refer to the slide deck for the afternoon lesson