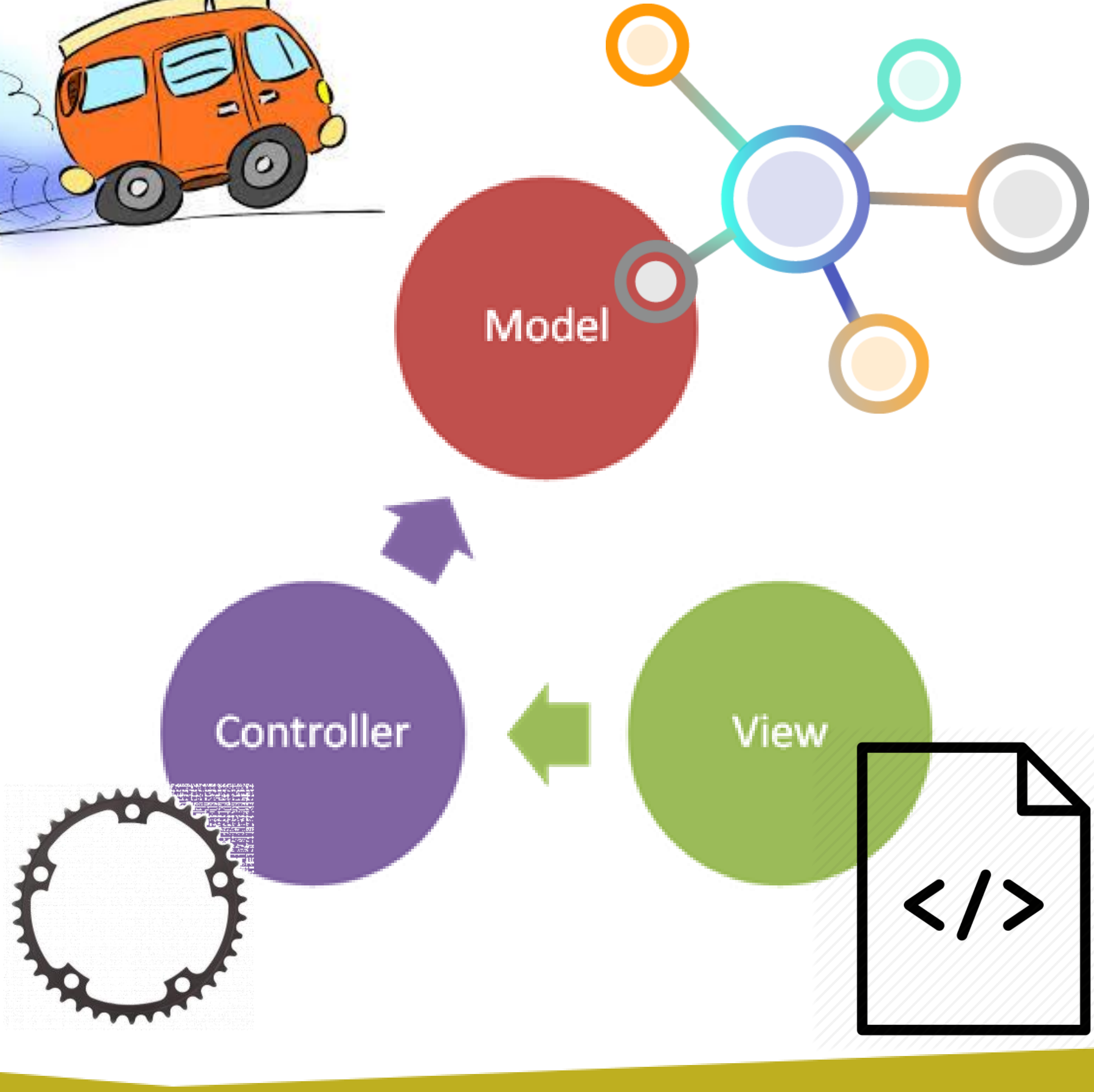
# CRUDL with MVC on Cars



coding_
academy

## CRUDL

When building apps, we usually have some entities that the application manages:

We usually need to:

- Create – add a new entity
- Read – read the entire details of the entity
- Update – update the entity
- Delete – remove the entity
- List – Read a list of the entity preview (filtered / ordered / paging / etc)

coding_ academy

# CRUDL on Cars

Lets review the starter:

Controller

```
function onInit() {⋯
}

function renderCars() {⋯
}

function onDeleteCar(carId) {⋯
}

function onAddCar() {⋯
}

function onUpdateCar(carId) {⋯
}

function onReadCar(carId) {⋯
}

function onCloseModal() {⋯
}
```

Model

```
const KEY = 'cars';
var gCars;

_createCars();

function getCars() {⋯
}

function deleteCar(carId) {⋯
}

function addCar(vendor) {⋯
}

function getCarById(carId) {⋯
}

function updateCar(carId, newSpeed) {⋯
}

function _createCar(vendor) {⋯
}

function _createCars() {⋯
}

function _saveCarsToStorage() {⋯
}
```

# Let's add some features

- When adding a car, show a section, use a select of vendors

```
const gVendors = ['audi', 'fiat', 'suzuki', 'honda’]
var vendor = gVendors[getRandomIntInclusive(0, gVendors.length-1)]
```

- Add Paging:

```
const PAGE_SIZE = 5;
var gPageIdx = 0;

function getCars() {
    var startIdx = gPageIdx*PAGE_SIZE;
    return gCars.slice(startIdx, startIdx + PAGE_SIZE)
}
function nextPage() {
    gPageIdx++;
    if (gPageIdx * PAGE_SIZE >= gCars.length ) {
        gPageIdx = 0;
    }
}
```

{coding_ academy

## Let’s add some features

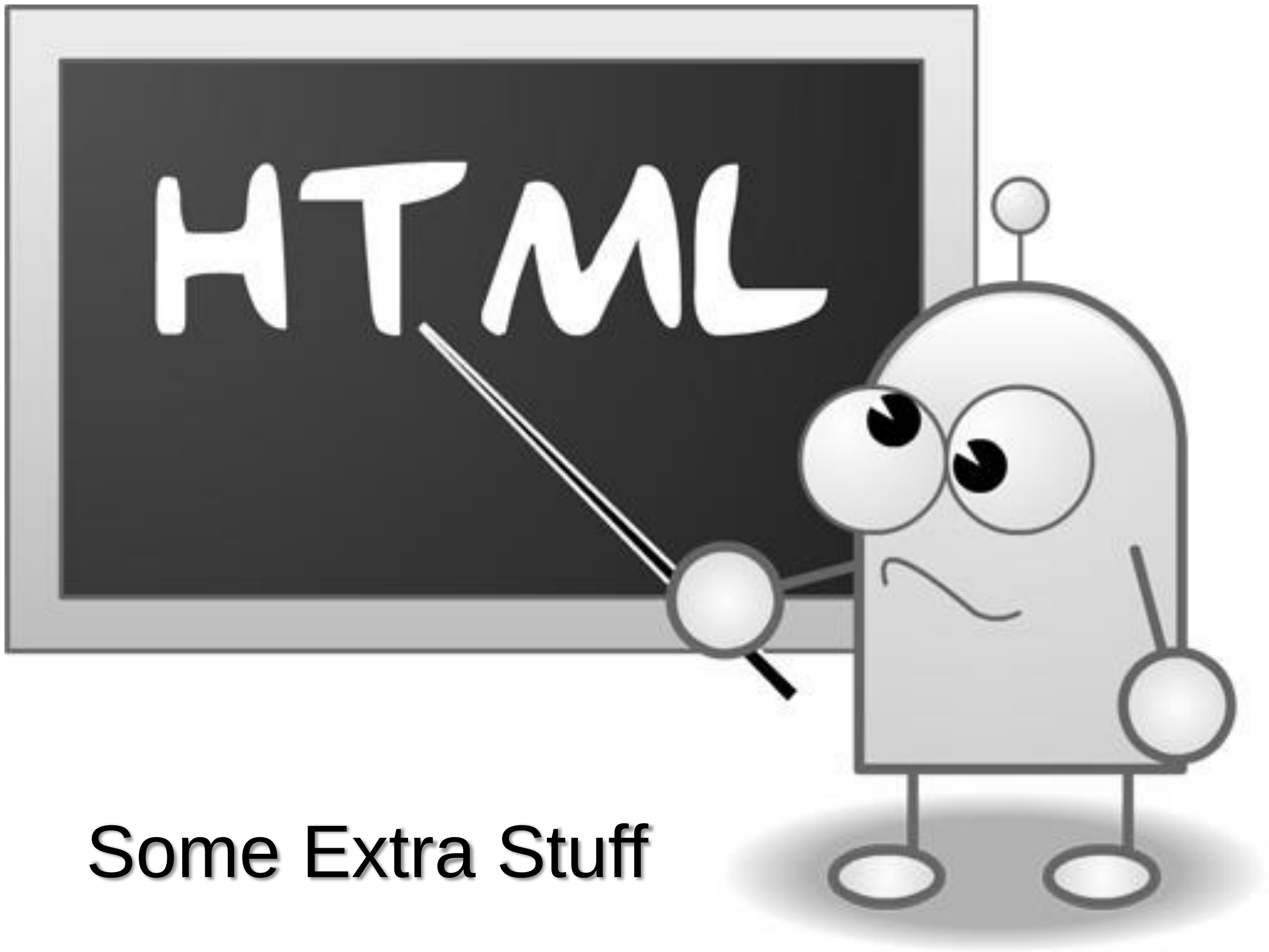
- ITP Add filtering and Sorting:

```
var gFilterBy = {…}
var gSortBy = {…}
```

coding_
academy

## Some more stuff
## we should know about

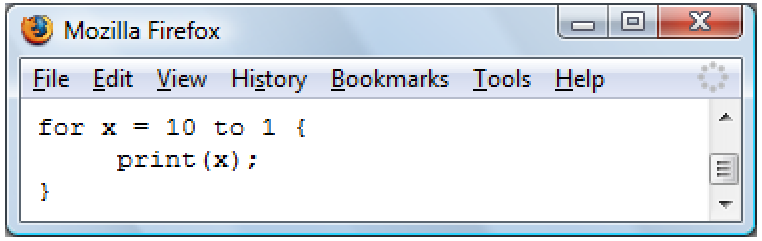
Would you like some more



Some Extra Stuff

## Preformatted Text

```
<pre>
for x = 10 to 1 {
     print(x);
}
</pre>
```



```
for x = 10 to 1 {
     print(x);
}
```

- Normally, new lines and multiple white spaces are ignored.
- Preformatted text is displayed with respect to white spaces and new lines.

## More about Anchors

- To open a link in a new tab, use:

```html
<a href="http://www.gmail.com/" target="_blank">
   Open Gmail (opens a new tab)
</a>
```

- The URL can point to any resource available on the web, e.g., picture, movie, etc:

## Named Anchors

- Named Anchors are used for linking to a different area in the same page (note the differences):

```
<a name="pageTop"><h1>Samples</h1></a>
...
...
<a href="#pageTop">Goto Top</a>
```

  - Note that the *pageTop* anchor is not displayed differently.

- Linking to a specific section in another page:

```
<a href="http://www.MyDomain.com/MyPage.html#someSection">
Goto Page at Specific Section</a>
```

coding_
academy

# window

- Represents the browser window, used for:
- Getting access to the URL (Location), the previous browsed pages (History), etc.
- Setting timeouts and intervals.
- Opening popup windows.
- **window is the default object, it can be used without specifying its name.**

```
window.setTimeout('alert("aha!")', 3000);

// same as:
setTimeout('alert("aha!")', 3000);
```

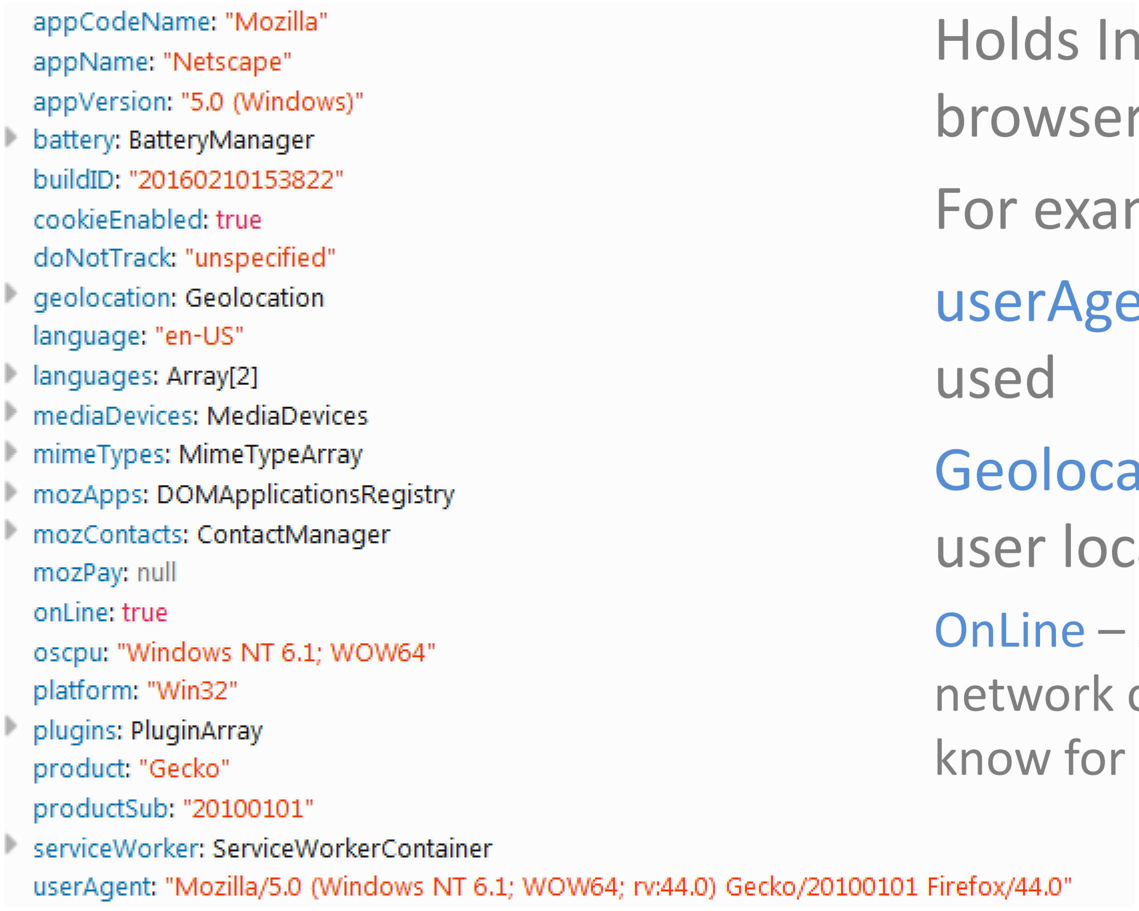
# Opening a window

You can open a window in JS (not a common thing to do )

- Note that popup blockers tends to block such popups, specially when the window is not opened as a result of user click

```
var popup = window.open('','','width=100,height=80')
popup.document.write("a Popup")
popup.focus()
```

# window.navigator



```
  appCodeName: "Mozilla"
  appName: "Netscape"
  appVersion: "5.0 (Windows)"
▸ battery: BatteryManager
  buildID: "20160210153822"
  cookieEnabled: true
  doNotTrack: "unspecified"
▸ geolocation: Geolocation
  language: "en-US"
▸ languages: Array[2]
▸ mediaDevices: MediaDevices
▸ mimeTypes: MimeTypeArray
▸ mozApps: DOMApplicationsRegistry
▸ mozContacts: ContactManager
  mozPay: null
  onLine: true
  oscpu: "Windows NT 6.1; WOW64"
  platform: "Win32"
▸ plugins: PluginArray
  product: "Gecko"
  productSub: "20100101"
▸ serviceWorker: ServiceWorkerContainer
  userAgent: "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:44.0) Gecko/20100101 Firefox/44.0"
```

Holds Information about the browsers and environment:

For example:

userAgent – which browser is used

Geolocation – get the current user location

OnLine – some indication about network connectivity (not enough to know for sure)

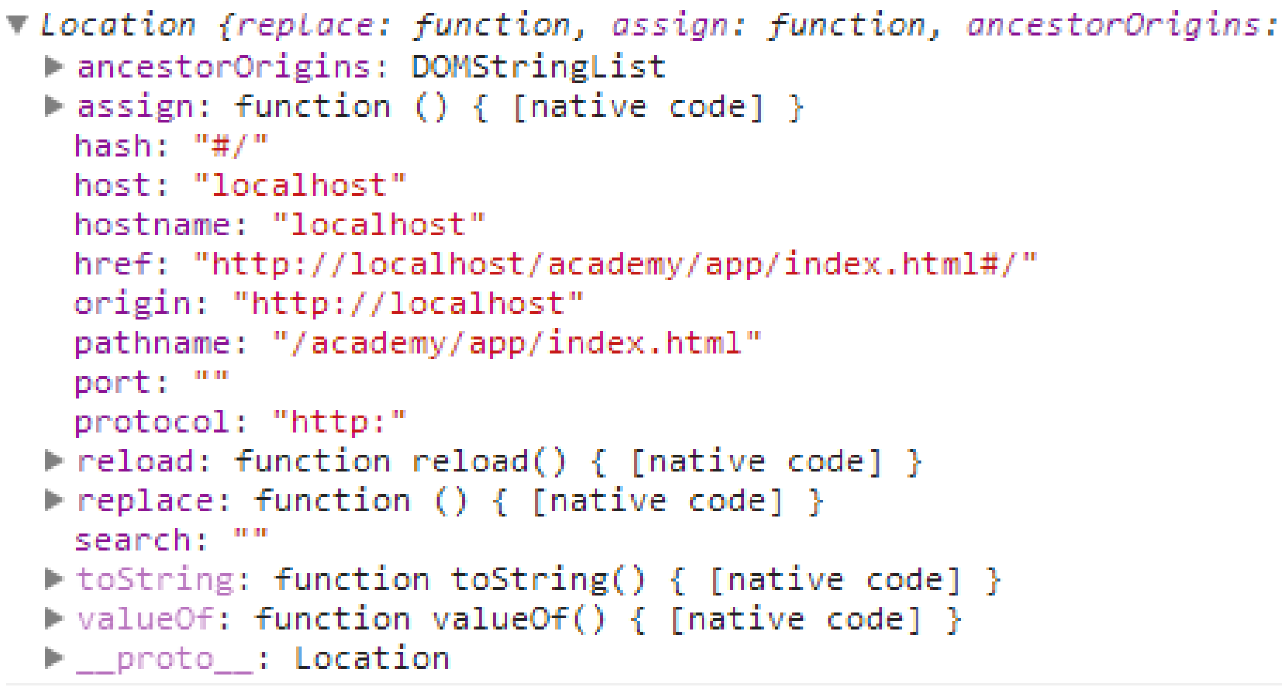
## window.history

Using the history object it is possible to simulate a click on the next/previous buttons.

```
<input type="button" value="Go Back"
onclick="window.history.go(-1)" />
```

# window.location

- The Location object holds information and methods regarding the current URL:



```
▼ Location {replace: function, assign: function, ancestorOrigins:
  ► ancestorOrigins: DOMStringList
  ► assign: function () { [native code] }
    hash: "#/"
    host: "localhost"
    hostname: "localhost"
    href: "http://localhost/academy/app/index.html#/"
    origin: "http://localhost"
    pathname: "/academy/app/index.html"
    port: ""
    protocol: "http:"
  ► reload: function reload() { [native code] }
  ► replace: function () { [native code] }
    search: ""
  ► toString: function toString() { [native code] }
  ► valueOf: function valueOf() { [native code] }
  ► __proto__: Location
```

## Navigating in the DOM tree

Note: We will not write code like that, this is just to help visualize the DOM

```
<html>
<head>
<title>Pets R Us</title>
<script>
function init () {
   document.body.children[0].childNodes[0].parentElement.parentElement
}
</script>
</head>
<body onload="init ()">
   <h1>We sell Pets</h1>
   <img src="pic1.jpg"/>
</body>
</html>
```

## Variadic Functions

When inside a function, we can access the function’s arguments using the special parameter *arguments*. This helps when creating variadic functions:

```
// function that receives an unknown parameters count  and returns the maximum:
function myMax() {
    var max = -Infinity;
    for (var i =0; i< arguments.length; i++) {
        if (arguments[i] > max) max = arguments[i];
    }
    return max;
}
console.log('Expecting: -Infinity', myMax());
console.log('Expecting: 0', myMax(0, 0));
console.log('Expecting: 11', myMax(9, 11, 7, 1));
```

## Variadic functions

- Use the implicit *arguments* object to implement variable-arity functions.
- Note – the *arguments* object is not a regular array, never try to shift it or modify it in any way (you can make a copy if needed: [].slice.call(arguments))

```
function calcAvg() {
   for (var i = 0, sum = 0, n = arguments.length;i < n; i++) {
      sum += arguments[i];
    }
   return sum / n;
}
```

## Math with floats

Just a useful technique

```
var totalPrice = 5.981;
var price = +Math.random().toFixed(3);

totalPrice += +price.toFixed(3);
totalPrice  = +totalPrice.toFixed(3);

console.log(totalPrice);
```

## Numerical Bases

0123456789

٠١٢٣٤٥٦٧٨٩

I II III IV V VI VII VIII IX X

০১২৩৪৫৬৭৮৯

൦൧൨൩൪൫൬൭൮൯

०१२३४५६७८९

〇一二三四五六七八九

## Numerical Bases

0123456789

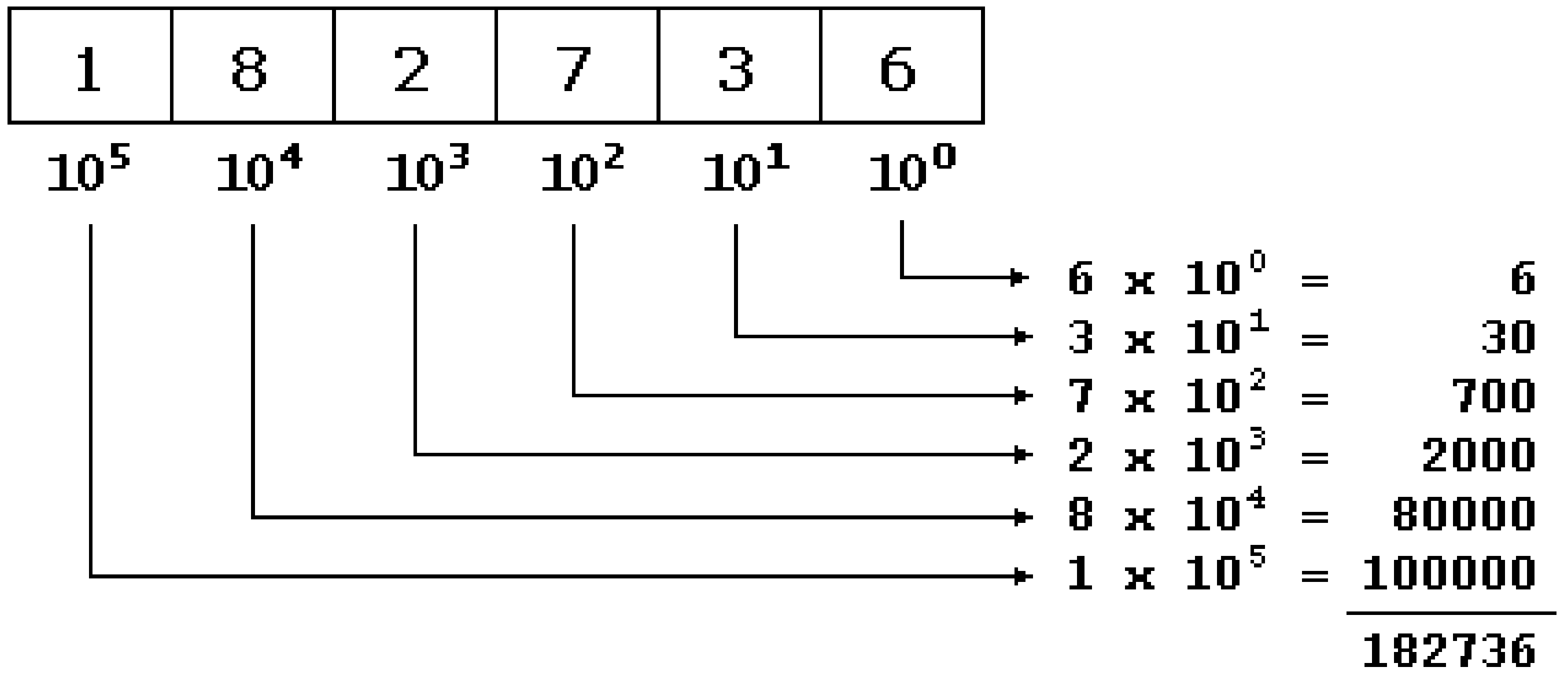
```
// Counting in Base 3:
0, 1, 2, 10, 11, 12, 20, 21, 22, 100

// Counting in Base 2:
0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010

// Counting in Base 16:
0, 1, 2, ... 9, A, B, C, D, E, F, 10
```

## Numerical Bases

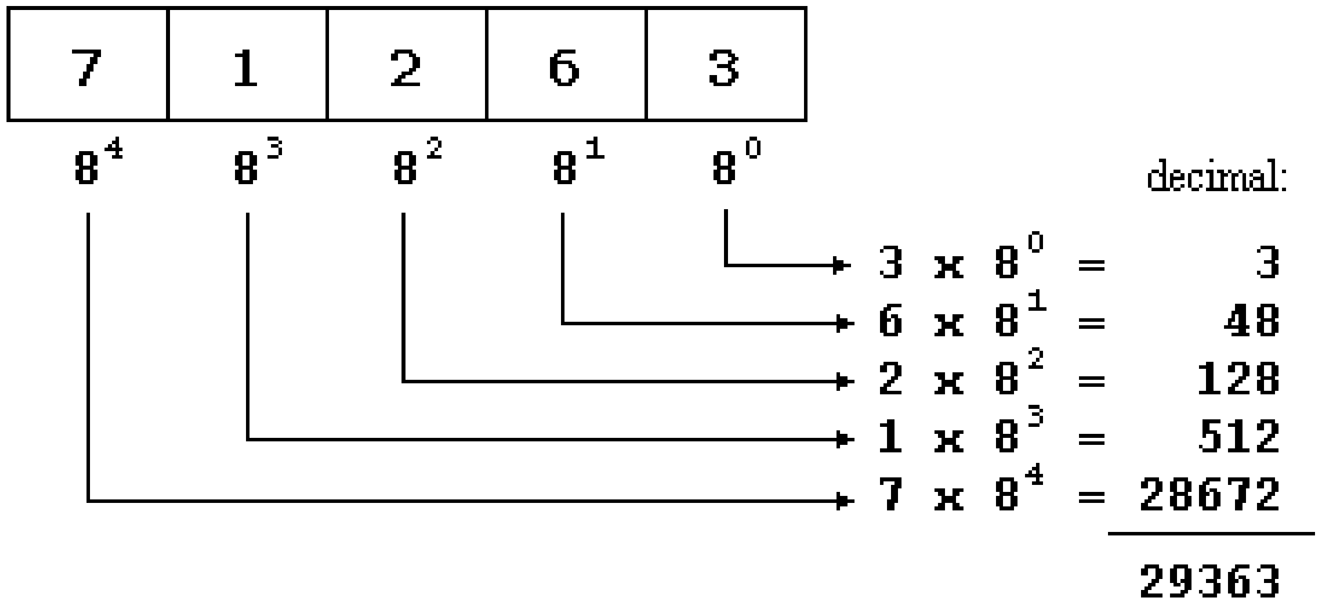
Here is how we build a number in base 10:



coding_ academy

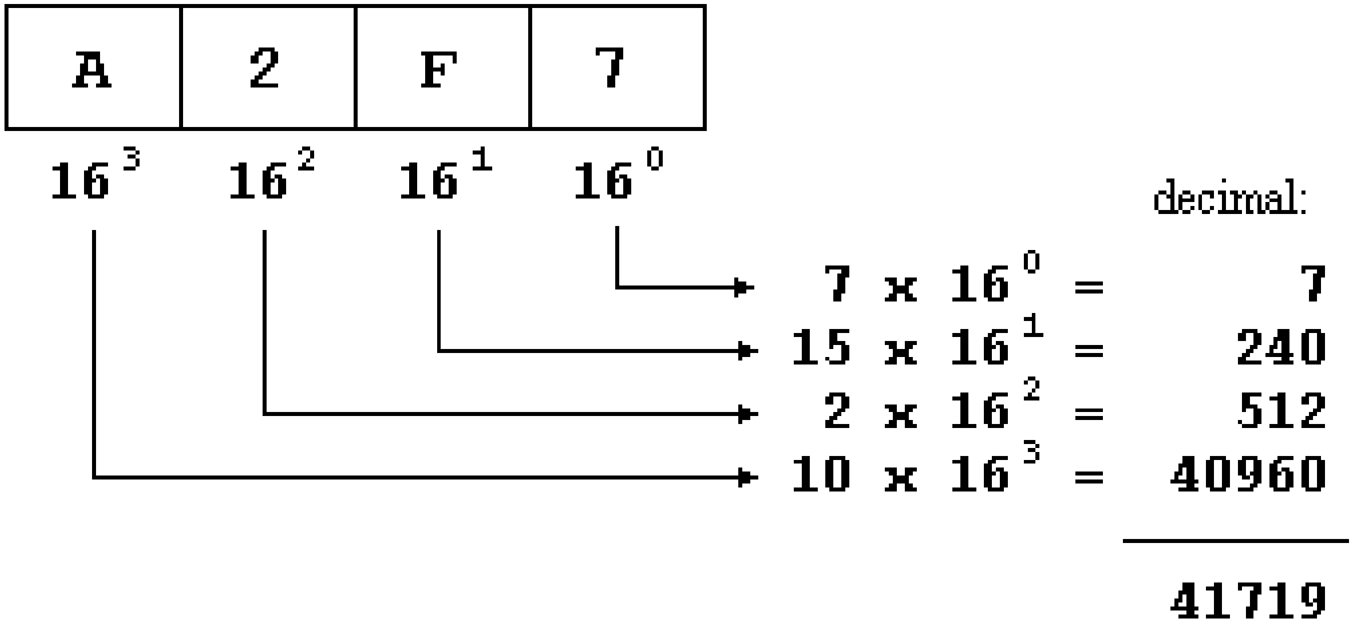
## Numerical Bases

Here is how we build a number in other bases:



# Numerical Bases

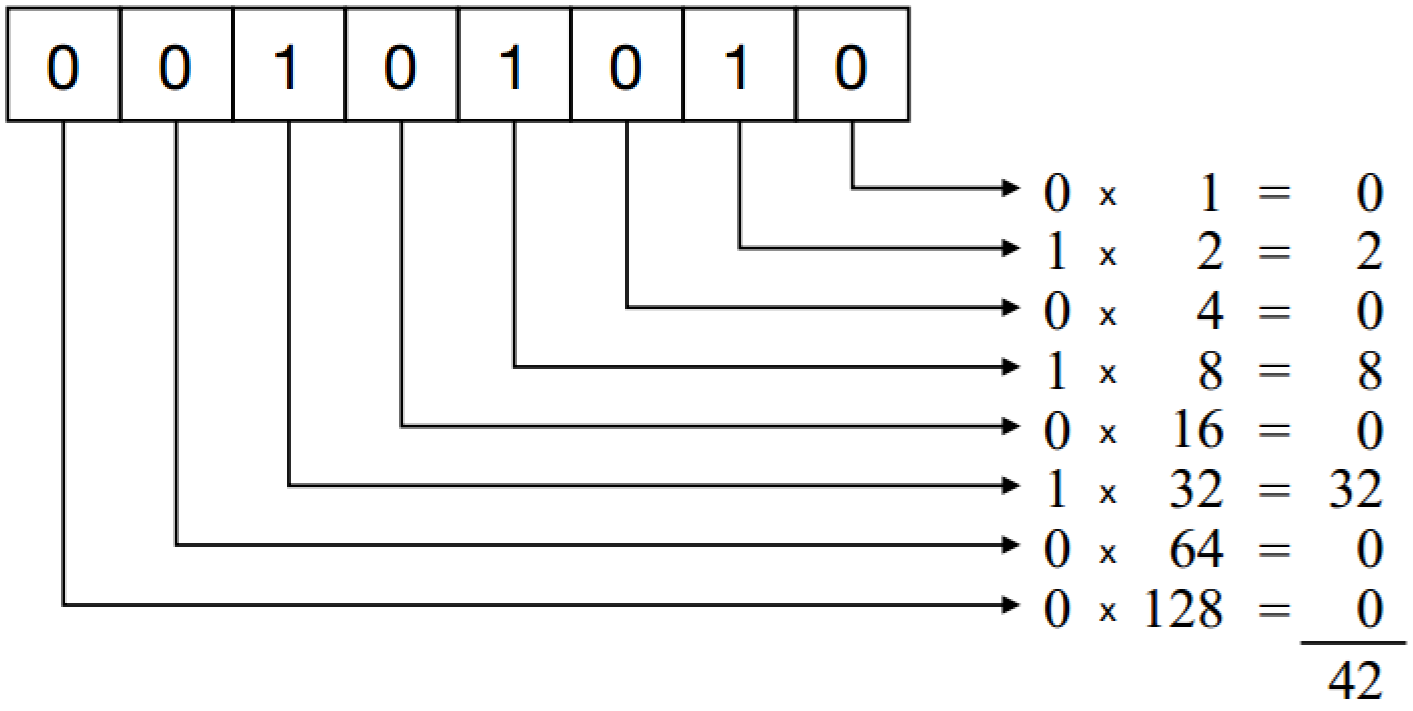
Here is how we build a number in other bases:



coding_ academy

## Numerical Bases

Here is how we build a number in base 2:



# Converting Decimal to Hexadecimal

## Here are the steps:

1. Divide the decimal number by 16. Treat the division as an integer division.
2. Write down the remainder (in hexadecimal).
3. Divide the result again by 16. Treat the division as an integer division.
4. Repeat step 2 and 3 until result is 0.
5. The hex value is the digit sequence of the remainders from the last to first.

Convert the number **256** DECIMAL to HEXADECIMAL

| DIVISION | RESULT | REMAINDER (in HEX) |
|---|---|---|
| 256 / 16 | 16 | 0 |
| 16 / 16 | 1 | 0 |
| 1 / 16 | 0 | 1 |
| | | |
| ANSWER | | 100 |

## More Examples

Convert the number **188** DECIMAL to HEXADECIMAL

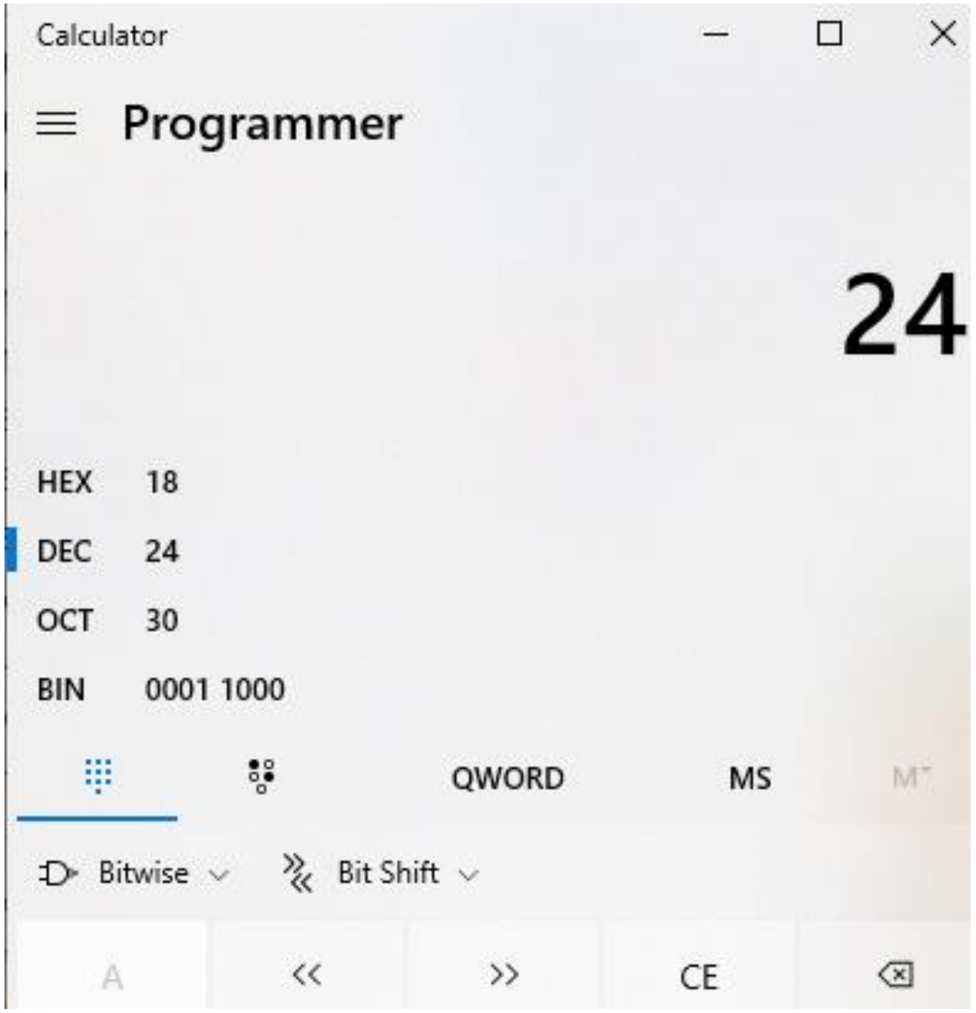
| **DIVISION** | **RESULT** | **REMAINDER (in HEX)** |
|---|---|---|
| 188 / 16 | 11 | C (12 decimal) |
| 11 / 16 | 0 | B (11 decimal) |
| | | |
| ANSWER | | BC |

Convert the number **590** DECIMAL to HEXADECIMAL

| **DIVISION** | **RESULT** | **REMAINDER (HEX)** |
|---|---|---|
| 590 / 16 | 36 | E (14 decimal) |
| 36 / 16 | 2 | 4 (4 decimal) |
| 2 / 16 | 0 | 2 (2 decimal) |
| | | |
| ANSWER | | 24E |

## Switching Bases

We can easily convert between common bases using the calculator

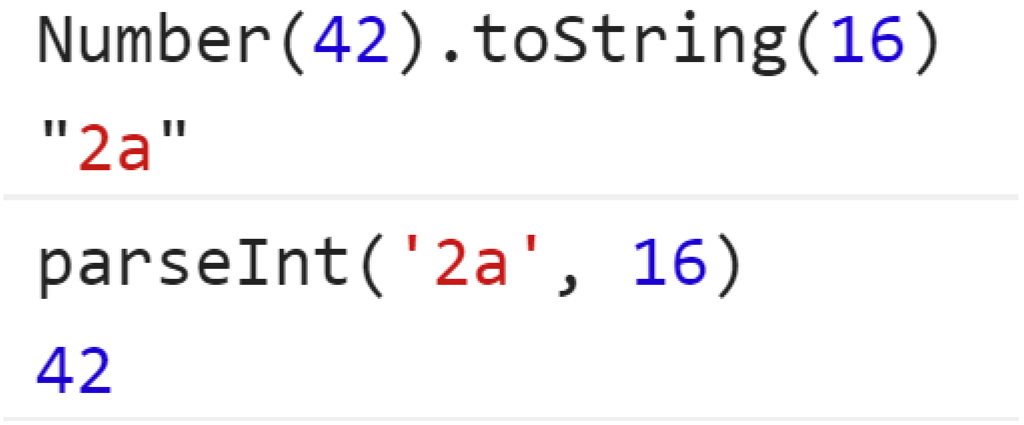


HEX 18
DEC 24
OCT 30
BIN 0001 1000

coding_
academy

## Switching between bases

Is easy…

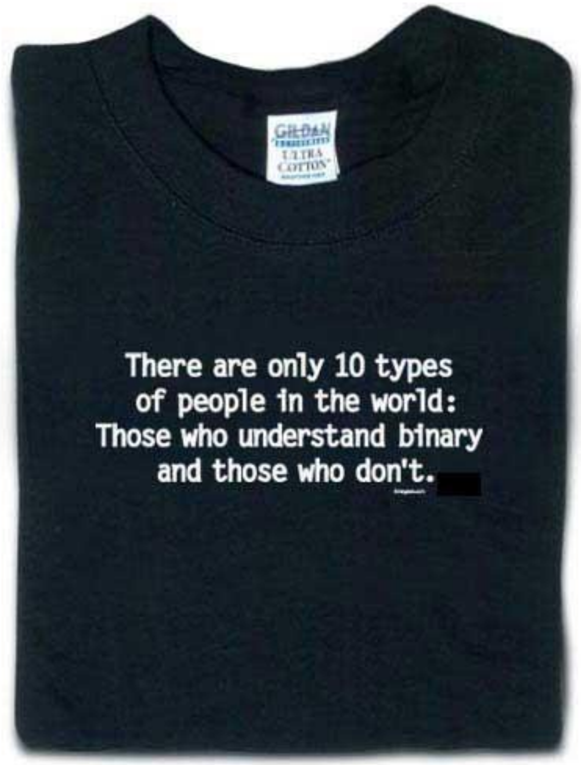


```
Number(42).toString(16)
"2a"
```

```
parseInt('2a', 16)
42
```

## Base 2

Dor is
00011110
29.10.17


GILDAN
ULTRA COTTON
There are only 10 types
of people in the world:
Those who understand binary
and those who don't. [redacted]

# Victorious!

You now know some more