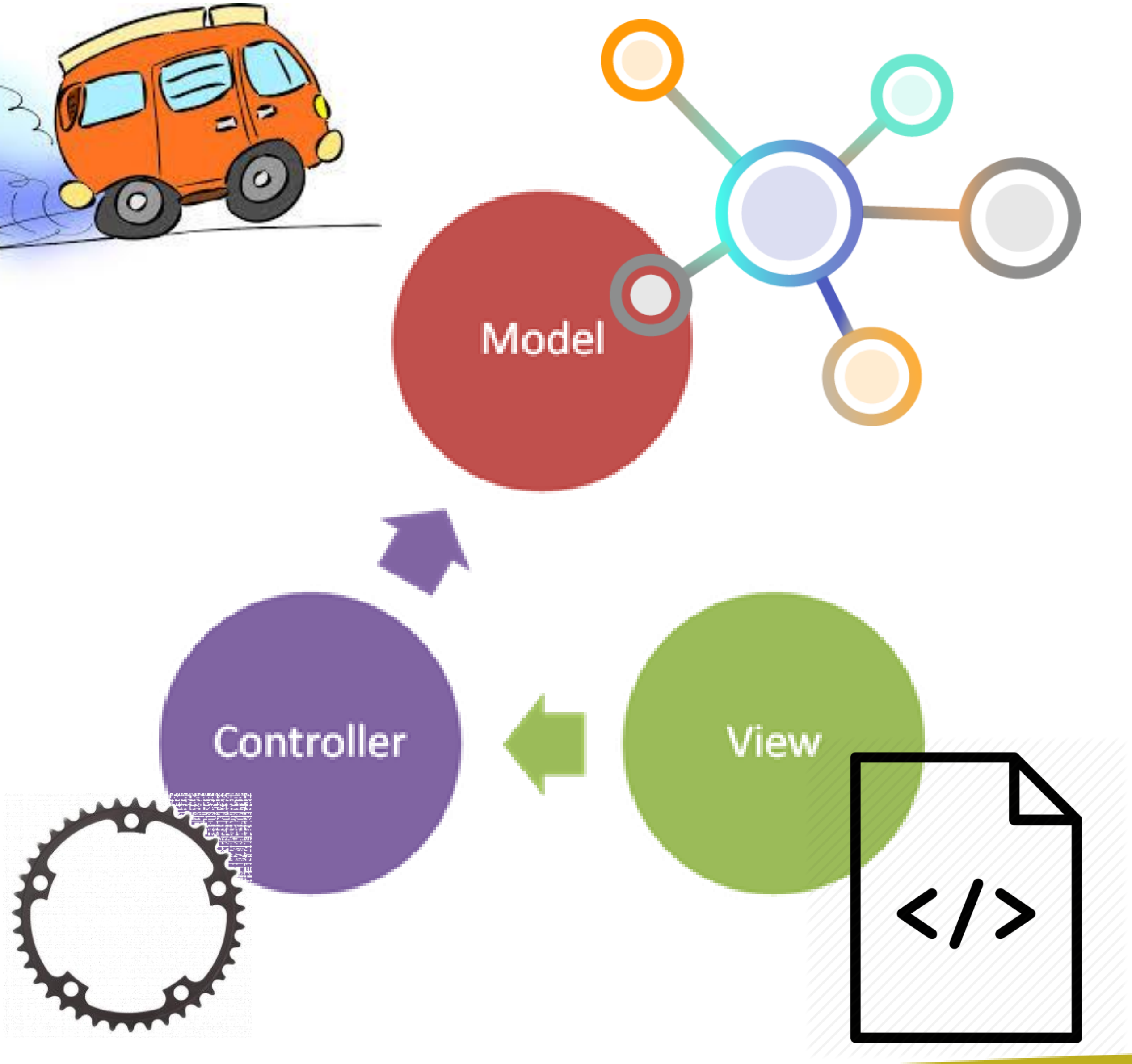
CRUDL with MVC on Cars



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)

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() {…
}
```

coding_academy

Let's focus on some features

Working with onerror

- Some elements (such as image, video, audio) may fire a special event: error
- We can listen to such an event and switch to some alternative content:

```
<img src="img/${car.vendor}.png"
     onerror="this.src='img/default.png'" />
```

Let's focus on some features

Working with Query-Params

Query parameters are **a defined set of parameters attached to the end of a url**. They are extensions of the URL that are used to help define specific content or actions based on the data being passed

In this case, we use to reflect our current car-filter:

```
http://mySite.com/index.html?vendor=audi&minSpeed=110
```

```
// Retrieve data from the current query-params
const queryStringParams = new URLSearchParams(window.location.search)
const filterBy = {
   vendor : queryStringParams.get('vendor') || '',
   minSpeed : +queryStringParams.get('minSpeed') || 0
}
```

Let’s focus on some features

Working with Query-Params

When the user change the filtering, we would like to reflect that in the query-params without re-loading the page, this can be achieved like this:

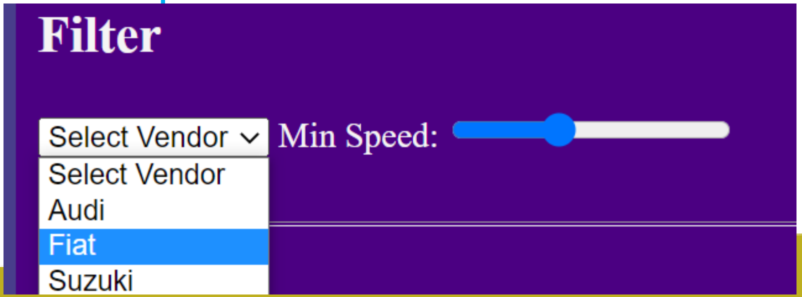
```
function onSetFilterBy(filterBy) {
    filterBy = setCarFilter(filterBy)
    renderCars()

    const queryStringParams =
          `?vendor=${filterBy.vendor}&minSpeed=${filterBy.minSpeed}`

    const newUrl = window.location.protocol + "//" + window.location.host +
                   window.location.pathname + queryStringParams

    window.history.pushState({ path: newUrl }, '', newUrl)
}
```

`http://mySite.com/index.html?vendor=audi&minSpeed=110`



coding_ academy

Let’s add some features

Improve the filter so vendors are retrieved from service

```
 const gVendors = ['audi', 'fiat', 'suzuki', 'honda’]

const strHTMLs = vendors.map(vendor =>
      `<option>${vendor}</option>`)
strHTMLs.unshift('<option value="">Select Vendor</option>')
```

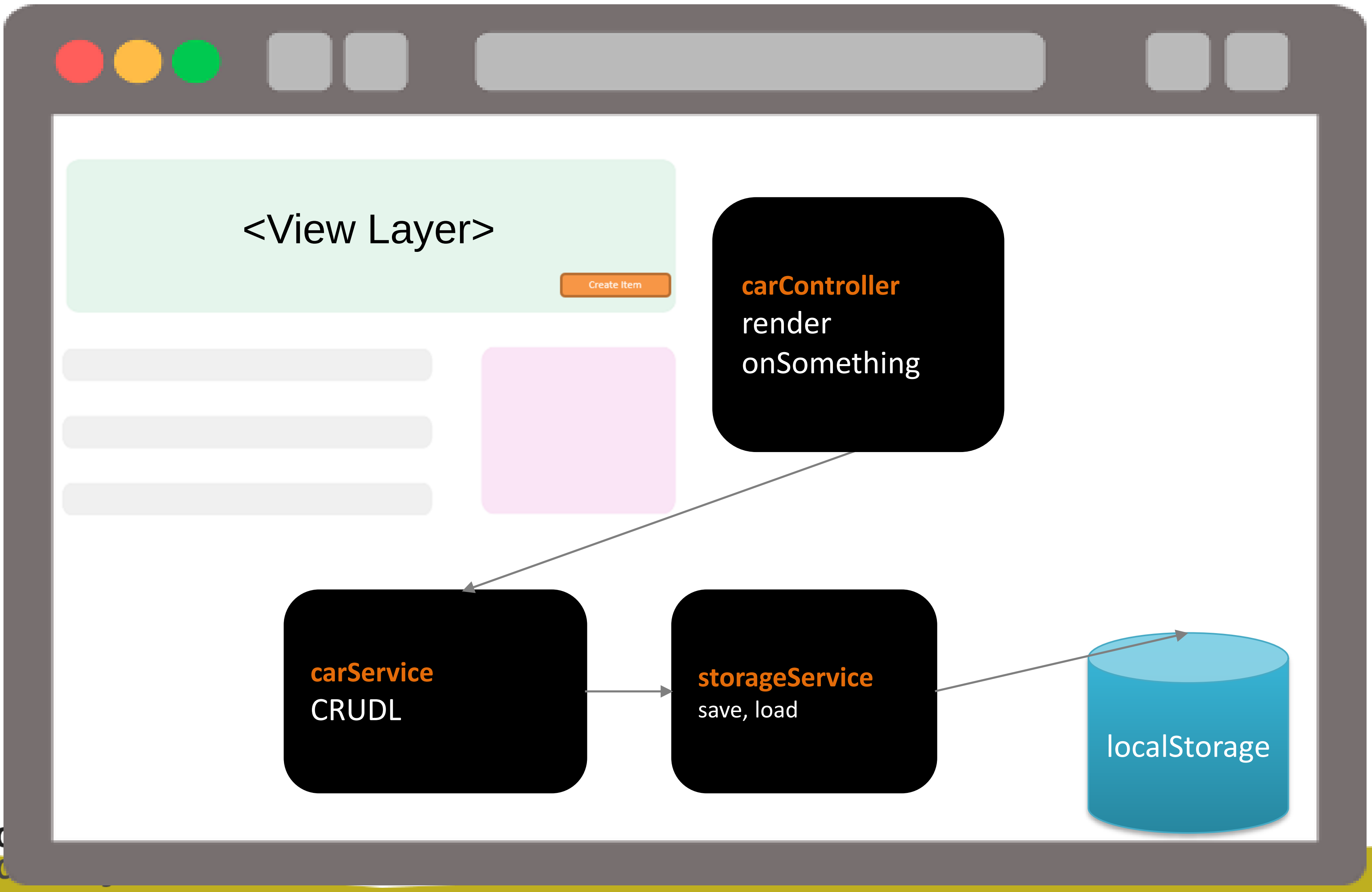
Let’s add some features

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
    }
}
```

Architecture block diagram



Some more stuff we should know about

Would you like some more

Some Extra Stuff

```css
#ninja {
  visibility: hidden;
  color: black;
}
```

Give me more

CSS precedence

- Element selector – 1 point
- Class / Attribute – 10 points
- Id – 100 points
- Inline style – 1000 points
- Important – Infinite

```
<h1 class="highlite now" style="color: blue;">Demos</h1>
```

```
h1 {
    color: red;
}
.highlite {
    color: pink
}
.highlite.now {
    color: brown
}
```

The `<h1>` will appear in Blue

coding_ academy

CSS precedence

- Element selector – 1 point
- Class / Attribute – 10 points
- Id – 100 points
- Inline style – 1000 points
- Important – Infinite

If 2 declarations have the same points – last wins

```css
h1 {
    color: red !important;
}
.highlite {
    color: pink
}
.highlite.now {
    color: brown
}
```

The `<h1>` will appear in Red

```html
<h1 class="highlite now" style="color: blue;">Demos</h1>
```

coding_ academy

Pseudo Classes

We can set each of the `<a>` (anchor) states to different CSS declarations:

- `a:hover {}` - when the mouse is hovering above
- `a:focus {}`– when a gets focus (e.g. using tab)
- `a:active {}` - when clicking the link
- `a:visited {}`- a link that we already viewed

Deprecated HTML Styling

- Deprecated means – it was supported in the past but removed later and should not be used.

- For example, the following are deprecated elements:
  <center>, <font>, <u>

- And here are some deprecated attributes:
  align, bgcolor, color

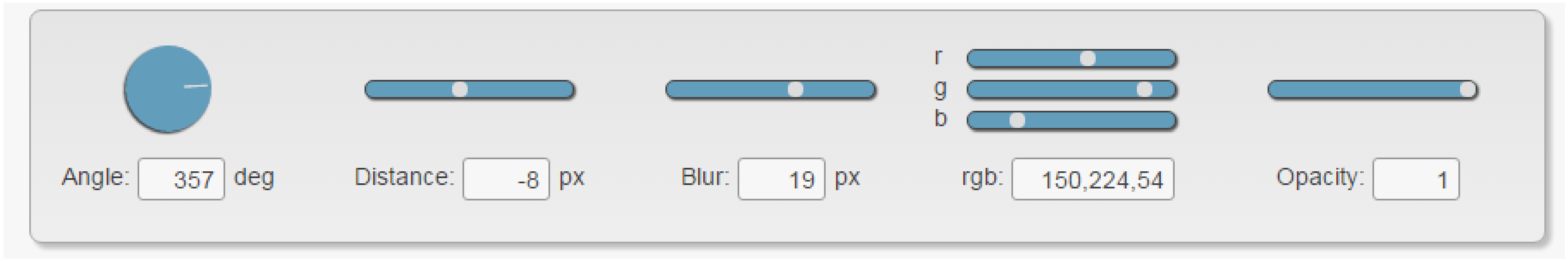
Web fonts

We can easily use various fonts today

Edwardian Script
Dobkin
Lucinda Calligraphy
Amaze
Brock Script
Freehand591 BT
Kristen ITC
Adler
Bernard Tango BT
BradleyHand ITC
French Script MT
Park Avenue
Pristina
T4C Beaulieux
Ghosthly
Harrington
Black Chancery
Rockwell
Americana BT
Blackadder ITC
Patrick
Elphinstone
Celtic Garamond
Gothikka
Cloister Black BT
Chalk
Childish Alpha
BoysRGross
Curlz MT
Burton's Nightmare
Comic Sans MS
Times New Roman
Haunt
Dracula
Psycho Poetry
Black Jack
Old English

Text effects

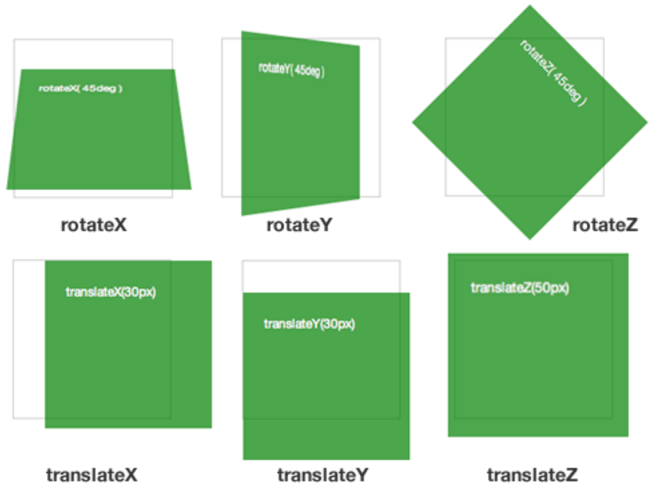
- text-shadow
- text-overflow
  - leave a visual “hint” to the user that text has been clipped ([demo]())


Angle: 357 deg
Distance: -8 px
Blur: 19 px
rgb: 150,224,54
Opacity: 1

coding_ academy

2d transforms

- translate()
- rotate()
- scale()
- skew()
- matrix()



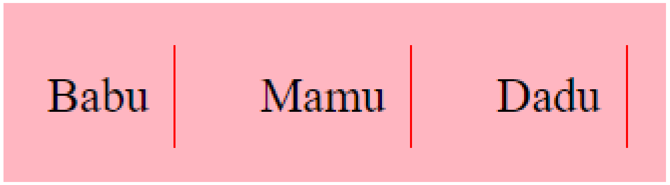
List items

- Two types of lists: <ul> and <ol>
  - <ol> – ordered list
  - <ul> – unordered list
- List items are used in many cases, examples:
  - Navigation menus
  - Shop Items

It is common to shutdown the default styling of lists with:

list-style:none

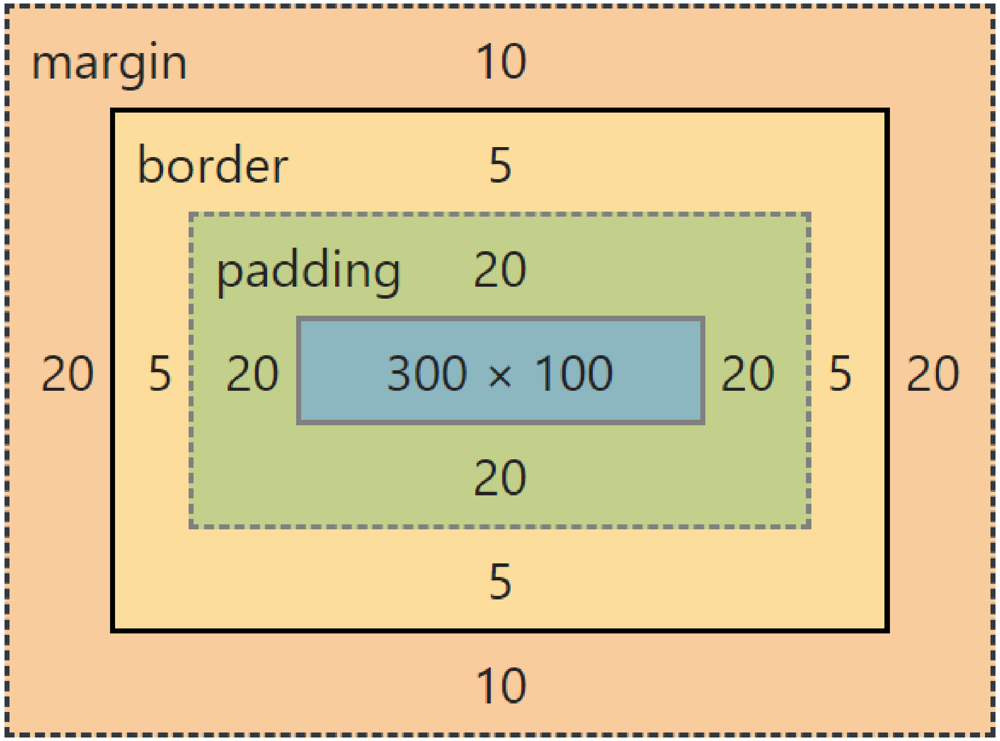
```
ul {
    list-style: none;
}
li {
    display: inline-block;
    border-right: 1px solid red;
    padding: 0.5em;
    margin: 0.5em;
}
```



Block elements use the Box Model

- The box model is composed from:
  - The content size (width, height)
  - The padding
  - The border
  - And the margin

```
div {
   width: 300px;
   height: 100px;
   padding: 20px;
   border: 5px solid orange;
   margin: 10px 20px;
}
```



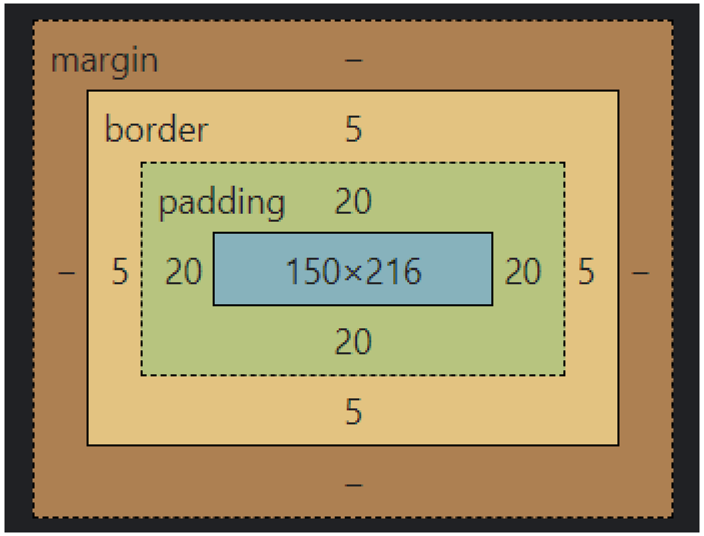
The box-sizing

- The default behavior in the box model is a bit confusing

  The default **box-sizing** is set to **content-box**

- It is common to change the box-sizing to border-box so our size definition will include the padding and the borders.

```
.box {
    box-sizing: border-box;
    padding: 20px;
    border: 5px solid black;
    width: 200px;
}
```

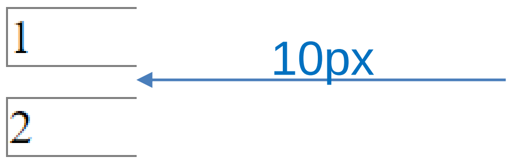


The box model and margins

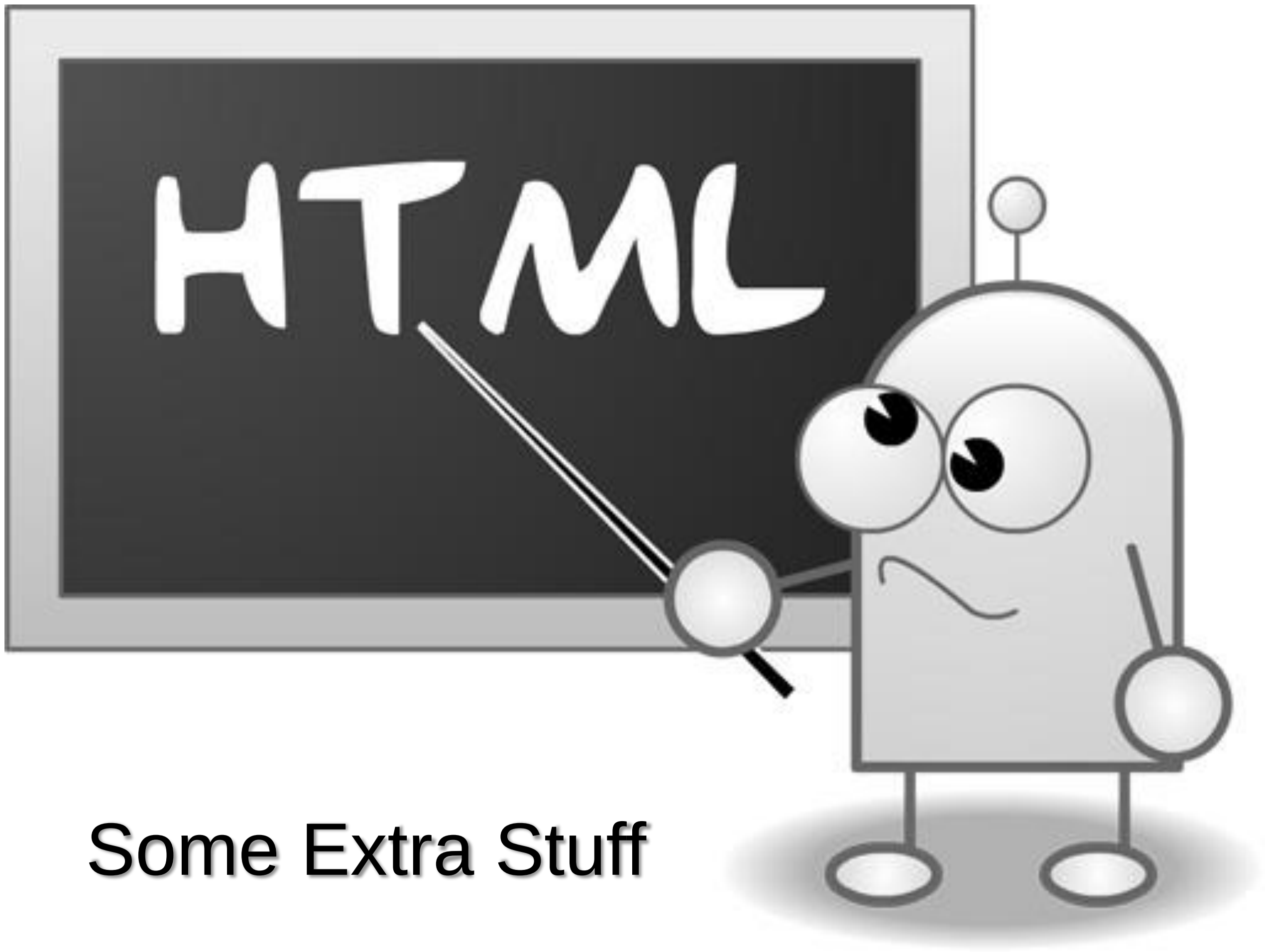
When two adjacent boxes request a margin, margin are collapsed (merged together)

```
<div class="box">1</div>
<div class="box">2</div>

.box {
    margin: 10px;
    border: 1px solid gray;
}
```



coding_ academy



Some Extra Stuff

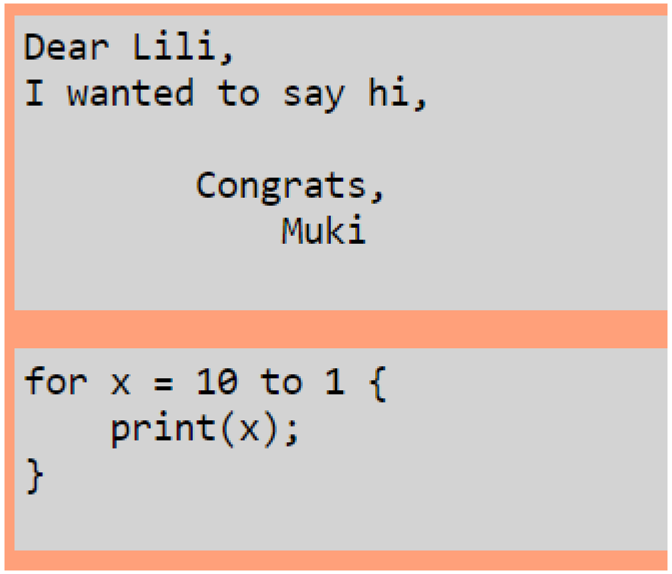
Preformatted Text

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

```
<pre>
Dear Lili,
I wanted to say hi,

        Congrats,
            Muki
</pre>

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



```
Dear Lili,
I wanted to say hi,

        Congrats,
            Muki
```

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

HTML Entities

- Some characters (like <) are better not placed inside the text (the browser might mistake them for tags).

- HTML Entities are used to output these special characters.
  For example: &lt; is the entity code for <.
- We can also use entity numbers, such as: &#165;

HTML Entities

Here are some examples:

Entity Output	Description	Code	Number
	non-breaking space	&nbsp;	&#160;
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
€	euro	&euro;	&#8364;
§	section	&sect;	&#167;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;

The <head> element

- The Head element contains information about the HTML document:

```html
<head>
    <title>My Page</title>
    <meta charset="utf-8" />
    <meta name="title" content="Frogi Pets Shop" />
    <meta name="description" content="Only the Best for your Pet" />
    <meta name="keywords" content="Frogi, Pets Food, miyahu" />
</head>
```

In a professional level HTML, the <head> contains much more, we will get to them later, lets have a look here…

More about Anchors

- The URL can point to any resource available on the web, e.g., picture, movie, etc:
- To open a link in a new tab, use:

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

Named Anchors

Named Anchors are used for linking to a different area in the same page:

```
<a name="services"></a>
<section class="container">
    <h2>Our Service</h2>
</section>
<a name="contact"></a>
<section class="container">
    <h2>Contact Us</h2>
</section>
```

```
<nav>
    <a href="#">Home</a> |
    <a href="#services">Services</a> |
    <a href="#contact">Contact Us</a> |
</nav>
```

Example - 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
- Getting access to the URL (Location), the previous browsed pages (History), etc.
- Setting timeouts and intervals.
- window is the default object, it can be used without specifying its name.

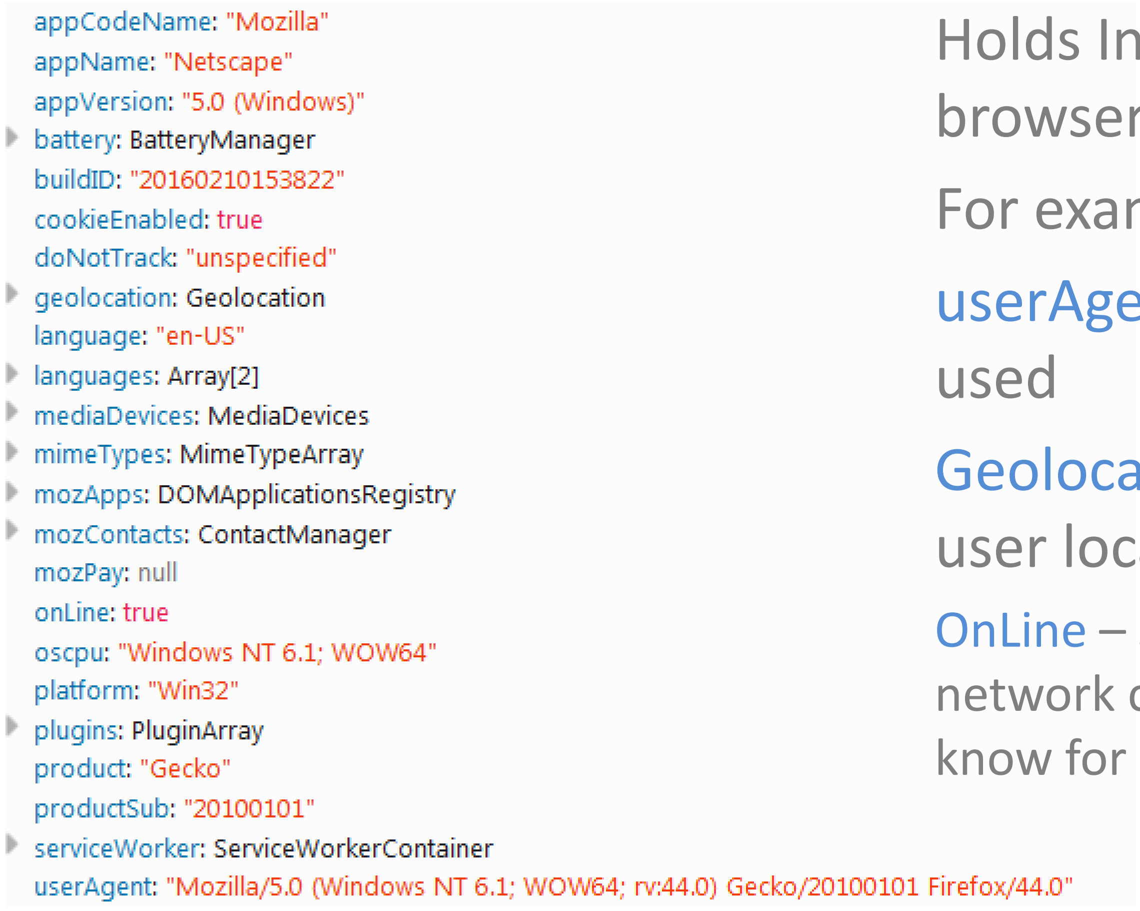
```
window.setTimeout(() => alert('aha!'), 3000)
// same as:
setTimeout(() => alert('aha!'), 3000)
```

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)

coding_ academy

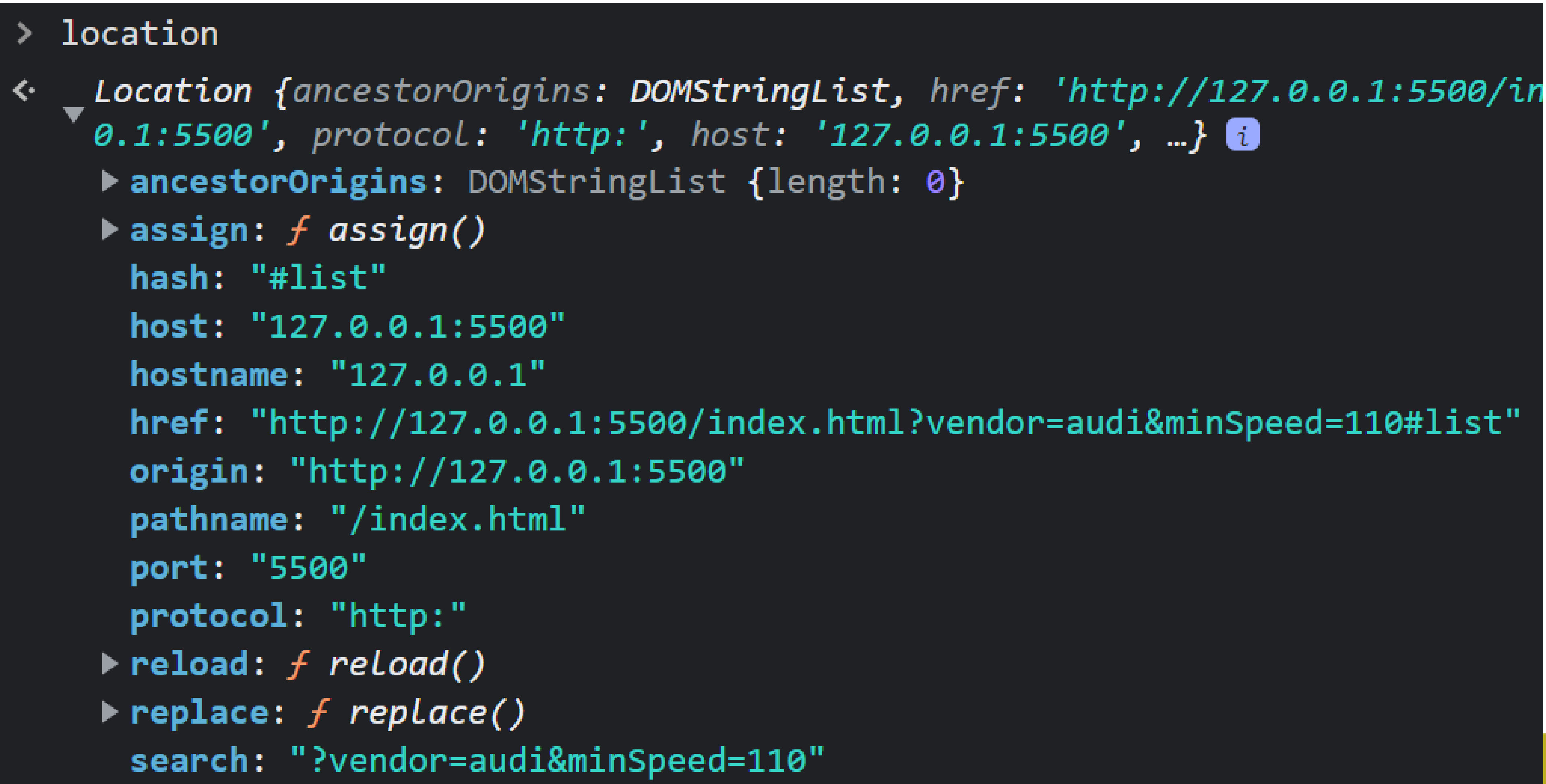
window.history

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)" />
```

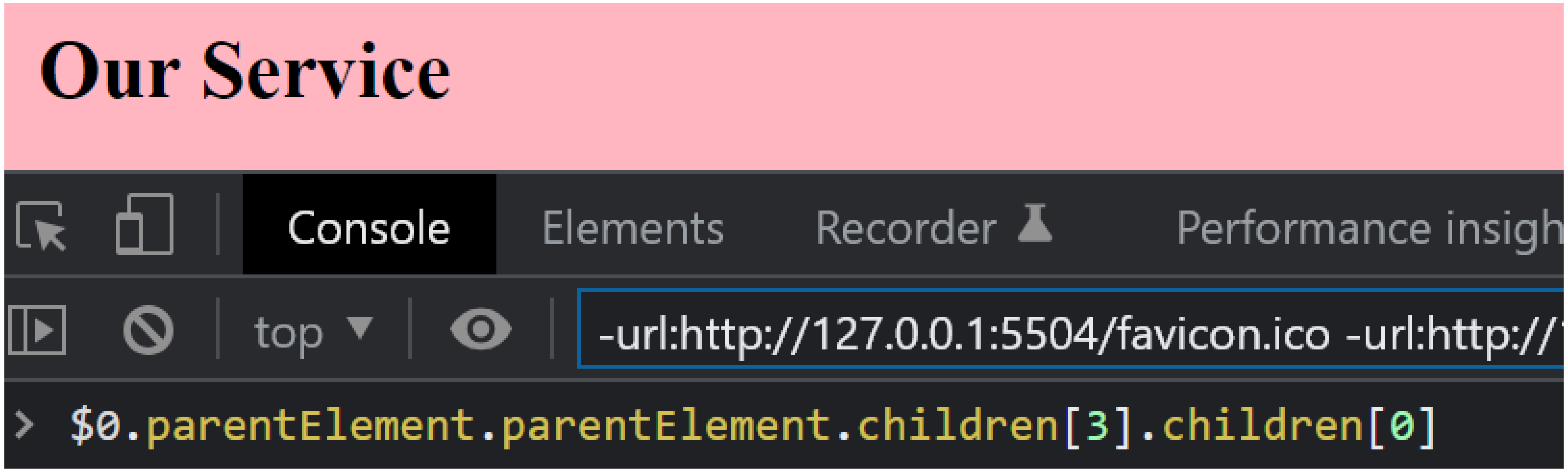
window.location

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



```
> location
<· ▼Location {ancestorOrigins: DOMStringList, href: 'http://127.0.0.1:5500/in
    0.1:5500', protocol: 'http:', host: '127.0.0.1:5500', …} i
   ▶ancestorOrigins: DOMStringList {length: 0}
   ▶assign: ƒ assign()
    hash: "#list"
    host: "127.0.0.1:5500"
    hostname: "127.0.0.1"
    href: "http://127.0.0.1:5500/index.html?vendor=audi&minSpeed=110#list"
    origin: "http://127.0.0.1:5500"
    pathname: "/index.html"
    port: "5500"
    protocol: "http:"
   ▶reload: ƒ reload()
   ▶replace: ƒ replace()
    search: "?vendor=audi&minSpeed=110"
```

Navigating in the DOM tree


Our Service

-url:http://127.0.0.1:5504/favicon.ico -url:http://

```
> $0.parentElement.parentElement.children[3].children[0]
```

Although possible, we will not write code that relies on a very specific DOM structure, we will prefer using more stable selectors

DOM – node's properties

- Every node in the DOM tree supports the following attributes:
  - e.parentNode - the parent node of e.
  - e.childNodes - the child nodes of e.
  - e.attributes - the attributes nodes of e.
  - e.innerHTML - the inner text value of e.
  - e.nodeName – read-only, the name of e.
    - For element – the tag name.
    - For attribute – the attribute name.
    - For text - #text.
  - e.nodeValue - the value of e.
    - For element – undefined.
    - For attribute – the attribute value.
    - For text – the text itself.
  - e.nodeType
    - The most useful types: 1 – element, 2 – attribute, 3 – text.

DOM – node's functions

- Every node in the DOM tree supports the following methods:
  - el.querySelector(selector)
  - el.getElementById(id) - get the element with a specified id under el.
  - el.getElementsByTagName(name) – get all elements of a specified tag under el.
  - el.appendChild(node) – adds a child node.
  - el.removeChild(node) – removes a child node.

Use *console.dir(el)* to examine the element

Variadic Functions

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

```
// 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('EXPECTED: -Infinity', myMax())
console.log('EXPECTED: 0', myMax(0, 0))
console.log('EXPECTED: 11', myMax(9, 11, 7, 1))
```

Variadic functions

- Here is another example:
- Note – the *arguments* object is not a real array: you cannot change it, it does not have functions such as *map* or *filter*, etc

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

Manipulating Dates

- The following code checks whether my birthday already occurred in the current year.
- Note that the month starts from 0, so 8 is September.

```
var now = new Date()
var myBirthday = new Date()
var inOneWeek = new Date()

myBirthday.setFullYear(now.getFullYear(), 8, 24)
inOneWeek.setDate(now.getDate() + 5)

if (now > myBirthday) {
    alert('After Birthday')
} else if (inOneWeek > myBirthday) {
    alert('Get Ready, birthday in 5 days')
} else {
    alert('Before Birthday')
}
```

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)
```

The Garbage Collector

Behind the scene, the JS Engine uses a process called garbage-collection to clean up *things* that are no longer accessiable

```
var movie = {
    name: 'Fight Club',
    actors: [{ name: 'Brad Pitt', salary: 10000 }]
}

console.log('movie', movie)

movie = null
// At this point we can no longer
// access the movie
// so it will be cleaned from memory
// by the garbage collector
```

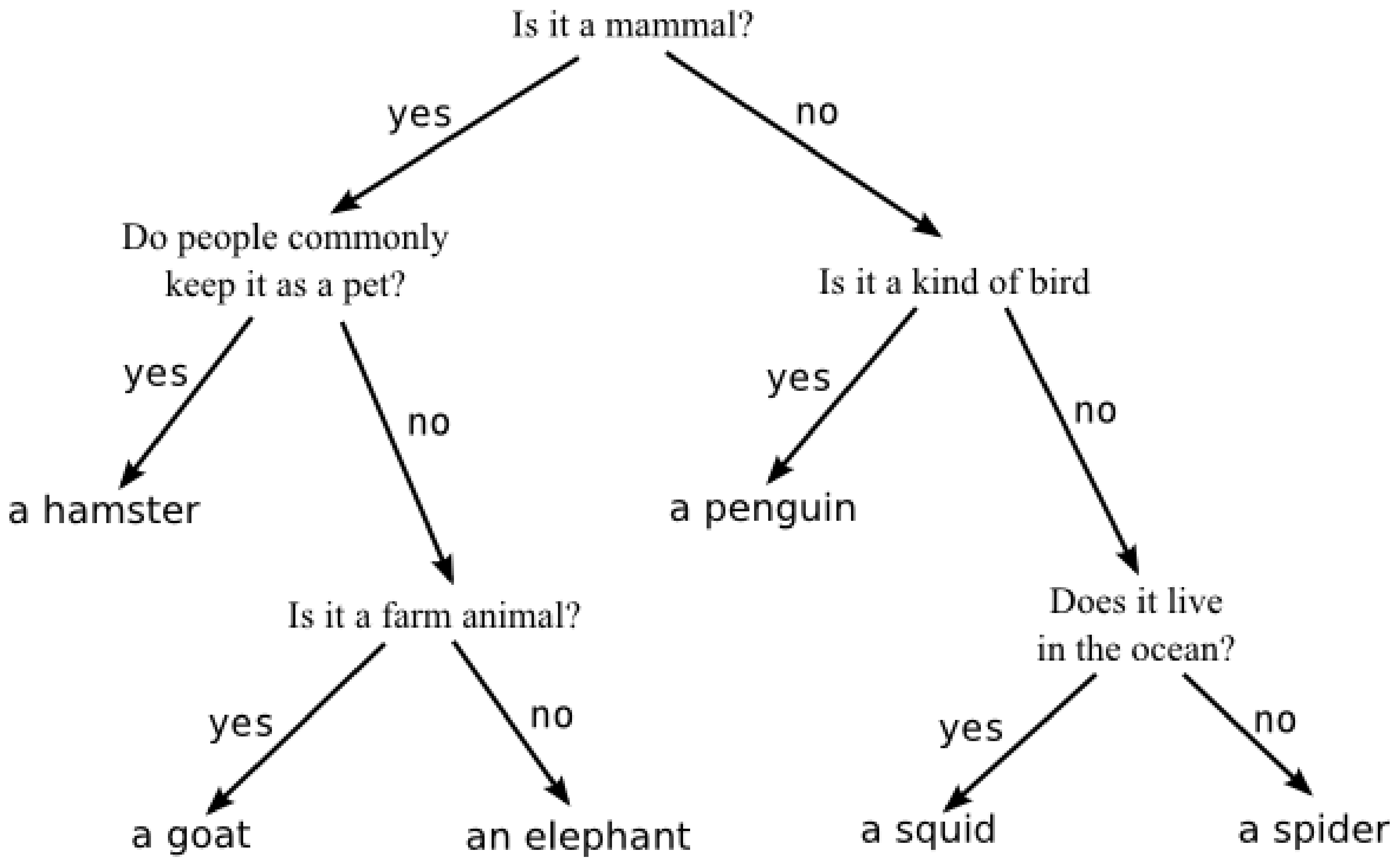
Tree data structure

Metushelah

- Build a data structure that represent a family tree
- 1)A person object that has:
  - i.an id (1001,1002, etc),
  - ii.a name,
  - iii.a birthdate (a date object)
  - iv.parents –array by size of 2
  - v.childs –array by any size
- 2)Build a data structure byId (an object) that will store all people by their ID
- 3)write functions:
  - i.addChild(toPersonId, childPerson)
  - ii.addParent(toPersonId, parentPerson)
- 4)Use those functions to create a tree with some data (i.e. your family, from Adam to Noah, etc.)
- 5)Write a function that prints the name of the parents that gave their son, the longest name.
- 6)Write a function getCousinsCount(personId) that returns number of cousins for a certain person

Guess Me Game

Lets build a simple game, based on a tree



```
var quest = {
    txt: 'a goat',
    yes : null,
    no : null
}
```

Victorious!

You now know some more

Victorious!

You now know some more