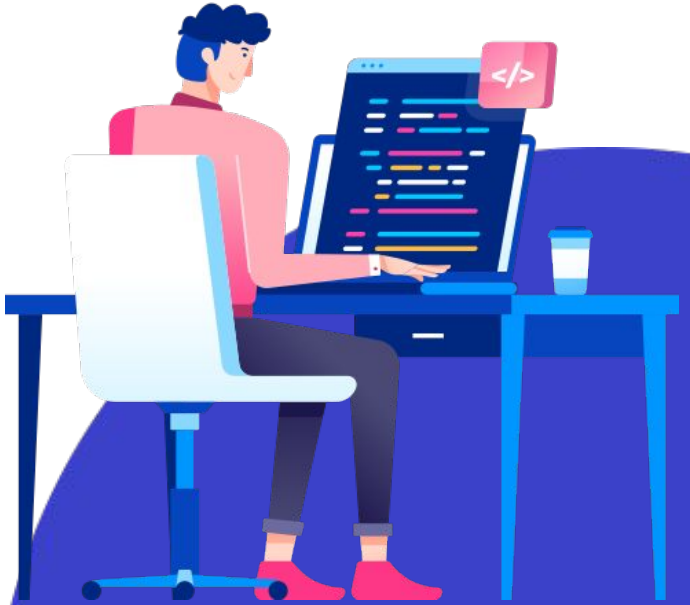
PROGRAMMING FUNDAMENTALS

Local Storage

DIPLOMA IN FULL-STACK DEVELOPMENT

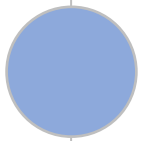
Certificate in **Computing Fundamentals**



LOCAL STORAGE

Types of Storage

Cookies, Local Storage



Using Local Storage

JSON, set/get LocalStorage items

WEB STORAGE API
LOCAL STORAGE

Source:
https://www.w3.org

Local Storage

TL;DR
HTML5 Storage is based on **named key/value pairs**. You store data based on a named key, then you can retrieve that data with the same key. The named key is a string.

The data can be any type supported by JavaScript, including strings, Booleans, integers, or floats. However, the data is actually **stored as a string**.

If you are storing and retrieving anything other than strings, you will need to use functions like parseInt() or parseFloat() to coerce your retrieved data into the expected JavaScript data type.

Web storage (or HTML5 storage) lets you **store data** in the browser.

The data it stores can only be **accessed by the domain** that set the data.

There are two types of storage: **local** and **session** storage.

They are implemented using the `localStorage` and `sessionStorage` objects.

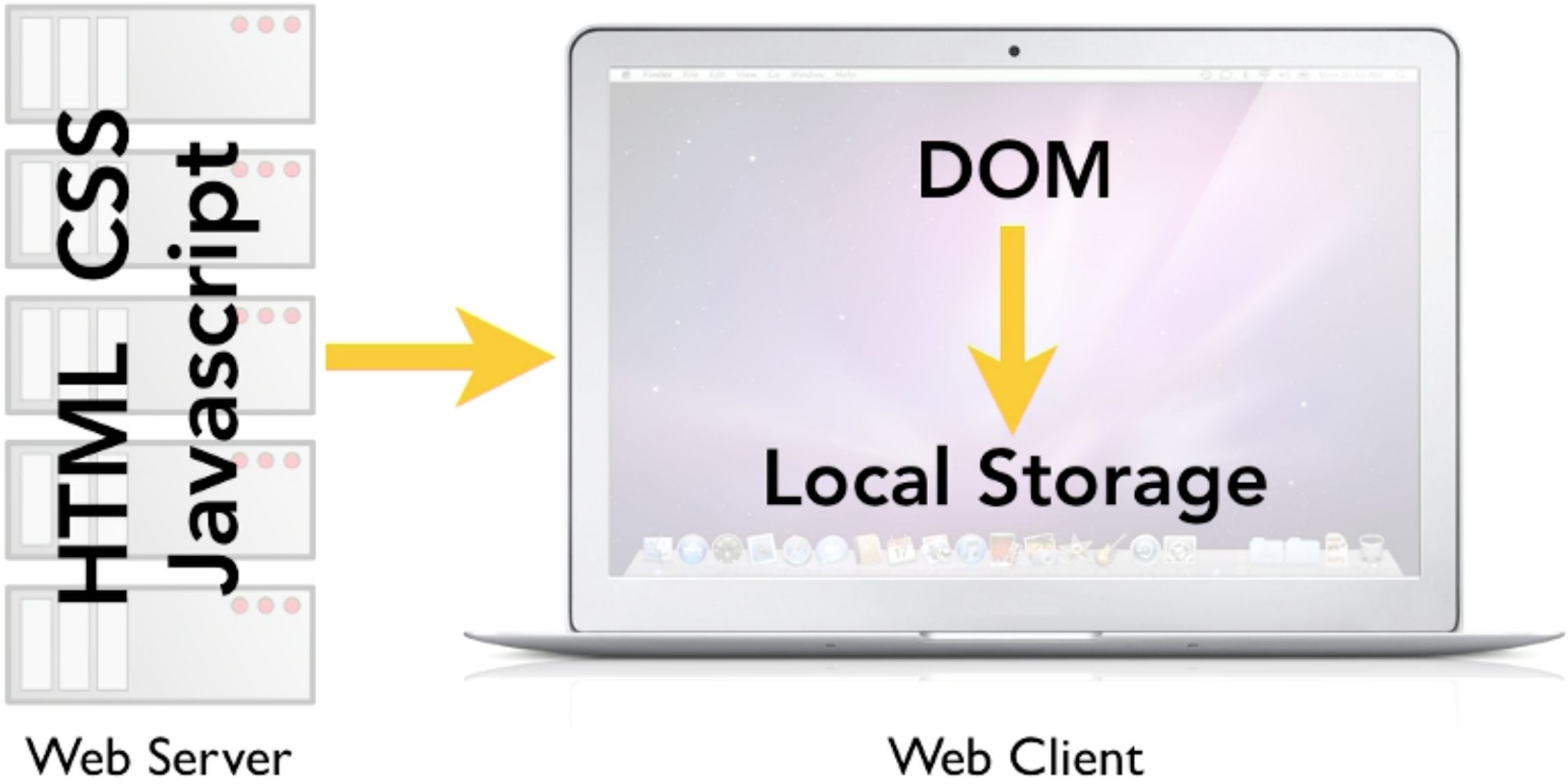
Commonly used in place of **cookies**

Both local and session storage have the same methods to get and set data, but each lasts a **different amount of time.**

STORAGE	LOCAL	SESSION
Stored when tab closed	Y	N
All open windows / tabs can access the data	Y	N

Most browser store up to **5MB** of data **per domain** in a storage object.

Should a site require more than the 5MB of data, typically the browser will ask the user for permission to store more data
(never rely on users agreeing to give a site more space)



For security protection, browsers employ a **same origin policy**, which means that data can only be accessed by other pages in the same origin.

Sidenote: this is also how iframes are being restricted

https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

1. **Protocol**: The protocol must be a match. If data was stored by a page that starts **http**, the storage object cannot be accessed via **https**.

2. **Subdomain**: The subdomain name must match.
For example, **maps.**google.com cannot access
data stored by **www.**google.com

3. **Domain**: The domain name must match.
For example, google. com cannot access local storage from facebook.com.

4. **Port**: The port number must match. Webservers can have many ports. Usually a port number is not specified in a URL, and the site uses port 80 for web pages, but the port number *can* be changed.

Accessing the storage API:

```
localStorage.setItem(key, value)
sessionStorage.setItem(key, value)

localStorage.getItem(key)
sessionStorage.getItem(key)
//key means identifier
```

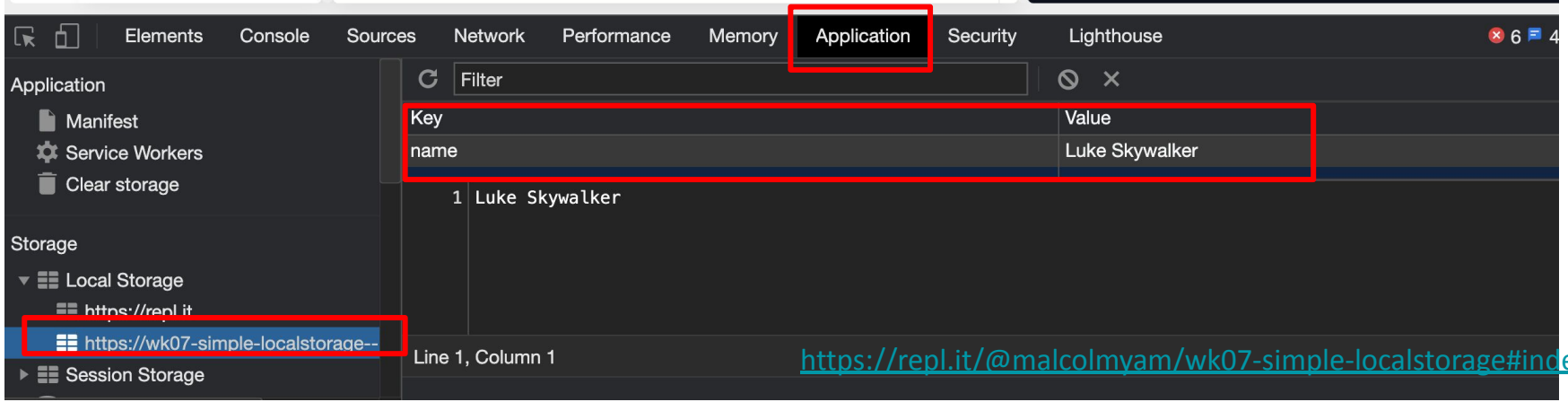
```
localStorage.setItem('name', 'Luke Skywalker');
let temp  = localStorage.getItem('name');
alert(localStorage['name']);
alert(localStorage.name);
```

https://repl.it/@malcolmyam/wk07-simple-localstorage#index.html

CHECKING WHETHER LOCALS STORAGE STORED IN CHROME BROWSER debugger

```
11  <body>
12    <script>
13      localStorage.setItem('name', 'Luke
        Skywalker');
14      var temp  = localStorage.getItem('name');
15      console.log(temp);
16      console.log(`Using array to access: $
        {localStorage['name']}`);
17    </script>
18  </body>
19
20  </html>
```

```
Luke Skywalker
Using array to access: Luke Skywalker
```



https://repl.it/@malcolmyam/wk07-simple-localstorage#index.html

Local Storage

Like other JavaScript objects, you can treat the localStorage object as an associative array (much easier).

Instead of using the getItem() and setItem() methods, you can simply use square brackets. For example, this snippet of code:

```
let foo = localStorage.getItem("bar");
....
....
localStorage.setItem("bar", foo);
```

==SAME==

```
let foo = localStorage["bar"];
....
....
localStorage["bar"] = foo;
```

https://repl.it/@malcolmyam/wk07-basic-localstorage#script.js

Accessing the storage API

Using **setters/getters**

```
//Store information
localStorage.setItem('age' , '12') ;
localStorage.setltem('color' , 'blue');

//Access information and store in variable
let age = localStorage.getItem('age');
let color= localStorage.getItem('color');

//Number of items stored
let items= localStorage.length;
```

Using the **dot notation**

```
//Store information (object notation)
localStorage.age = 12;
localStorage.color  = 'blue';

//Access information(object notation)
let age = localStorage.age;
let color= localStorage.color;

//Number of items stored
let items= localStorage.length;
```

METHODS & PROPERTIES

METHOD	DESCRIPTION
setItem(key,value)	Creates a new key/value pair
getItem(key)	Gets the value for the specified key
removeItem(key)	Removes the key/value pair for the specified key
clear()	Clears all information from that storage object

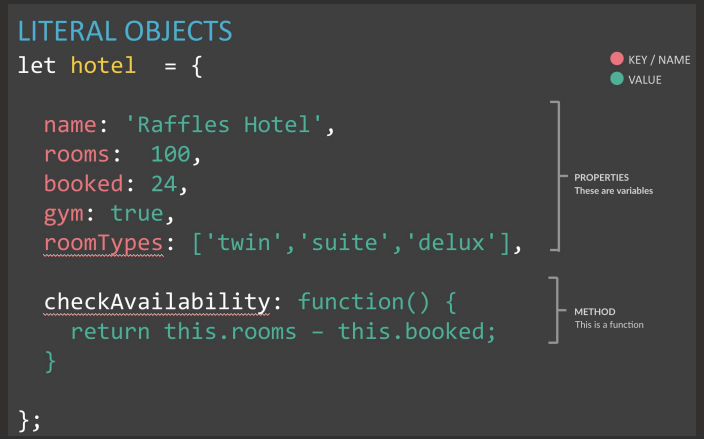
PROPERTY	DESCRIPTION
length	Number of keys

WHAT ABOUT OBJECTS?

DATA FORMATS
JSON

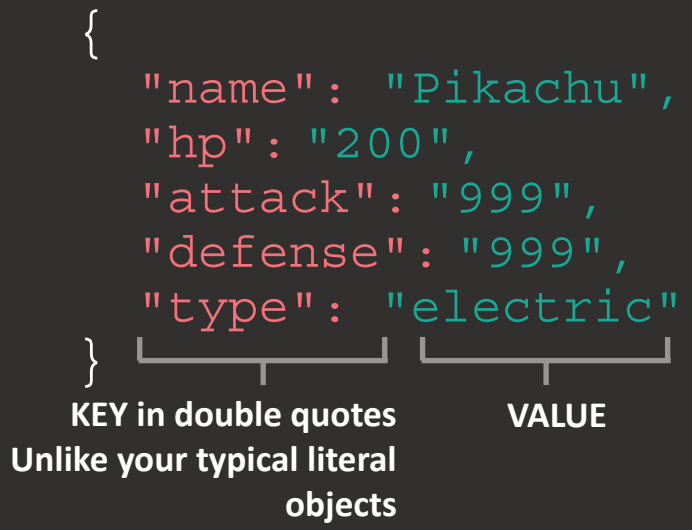
JSON looks like object literal syntax but it is just **data**, not an object:

```
{
   "name": "Pikachu",
   "hp": 200,
   "attack": 999,
   "defense": 999,
   "type": "electric"
}
```



https://www.w3schools.com/js/js_json_intro.asp
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON

JSON data is made up of **keys** and **values**:



In JSON, the **key** should be placed in **double quotes** (not single quotes).

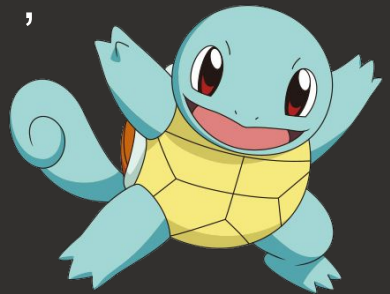
The key (or name) is separated from its value by a **colon**.

Each key/ value pair is separated by a comma. However, note that there is *no* comma after the last key/value pair

The value can be a string, number, Boolean, array, **object** or null.

You can nest objects.

```
let pokemon = [
    {
        "name":"Pikachu",
        "hp":100,"attack":9
        "defense":8,
        "color":"yellow",
        "type":"electric"
    },
    {
        "name":"squirtle",
        "hp":"47",
        "attack":"39",
        "defense":"38",
        "color":"blue",
        "type":"water"
    }
    ]
```



NESTING OBJECTS with JSON

pokemon[0].name //pikachu
pokemon[1].name //squirtle

Image Source:
http://pokemon.wikia.com/wiki/

JavaScript has a `JSON` object with two important methods:

1: Convert a JavaScript object to a string:

```
JSON.stringify();
```

2: Convert a string to a JavaScript object:

```
JSON.parse();
```

JSON as long Strings (objects)

```
let contact = function (fname) {
  this.firstName = fname;
};
let example = new contact('Johnny');
localStorage.setItem('user1', JSON.stringify(example));

localStorage.getItem('user1');??//object?
```

Parse it instead!

```
let exampleObj = JSON.parse(localStorage.getItem('user1'));
```

We use JSON.parse() to convert JSON into a Javascript object

https://repl.it/@malcolmyam/wk07-simplejson#script.js
https://www.w3schools.com/js/js_json_parse.asp

OBJECTS?

```
function Pokemon(name, hp, attack, defense, color, type) {
    this.name = name;
    this.hp = hp;
    this.attack = attack;
    this.defense = defense;
    this.color = color;
    this.type = type;
}
//let's create our base and store pikachu
let pikachu = new Pokemon("Pikachu", 100, 9, 8, "yellow", "electric");

//
let giantPokeBall = [];
giantPokeBall.push(pikachu);

localStorage.setItem("pokemonList", JSON.stringify(giantPokeBall));

//creates an array
let tempList = JSON.parse(localStorage.getItem("pokemonList"));
```

Image Source:
http://pokemon.wikia.com/wiki/Pikachu

Clearing Storage

```
localStorage.clear();
localStorage.removeItem(<item key>);
```

https://repl.it/@malcolmyam/wk07-sample-obj-localstorage#script.js

STORAGE LIMITATIONS

Browser limitation 5 MB

Able to request the browser for more storage space

“QUOTA_EXCEEDED_ERR” is the exception that will get thrown if you exceed your storage quota of 5 megabytes

When Quota Exceeds?

```
try {
  localStorage.setItem(data.name,JSON.stringify(data));
} catch(domException) {
  if (domException.name === 'QuotaExceededError' ||
      domException.name === 'NS_ERROR_DOM_QUOTA_REACHED')
  {
    // Fallback code comes here.
  }
}
```

ACTIVITY

“STORE Basic” (30min)

Follow the instructions below before starting the questions
https://replit.com/@immalcolm/form-localstorage

1. Either fork the repl.it or copy out the source code to your local machine
2. Based on the JS code in the HTML file, modify the TODO portion.
3. check whether there's local storage data 'username'
4. if there is, retrieve it and display first when the page is loaded
5. if not, set the localstorage item 'username' when the user submits the name
6. Download your final answer into your computer.

Sidenote: Use the browser’s debugger or repl.it console for quick testing

SUBMISSION
Simply download and save your own codebase

ACTIVITY

“STORE Advanced” (20min)

Follow the instructions below before starting the questions
https://replit.com/@immalcolm/form-obj-localstorage

1. Either fork the repl.it or copy out the source code to your local machine
2. Follow the README.md file to start on your project

Sidenote: Use the browser’s debugger or repl.it console for quick testing
Look at how the pokemon code is done :)

SUBMISSION
Simply download and save your own codebase

Sample Codes

Local Storage

Basic Local Storage example
https://replit.com/@Jian_Ting_Donov/localStorage-Simple

Advanced Local Storage with combined use of JSON data
https://replit.com/@Jian_Ting_Donov/localStorage-Advanced

Summary

What have we covered?

JSON

LocalStorage

Key Takeaway?

Practise & Practise

The usage of localstorage and JSON is versatile and can be applied to create and develop more complex applications. A simple example is a Create, Read, Update & Delete (CRUD) system, like a to-do list that persist even when you close the browser

TIDBITS

http://www.html5rocks.com/en/tutorials/offline/whats-offline/
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript
http://www.html5rocks.com/en/tutorials/offline/quota-research/