CS7025 Programming for Digital Media

Lesson 8 – Storage

Recap
Classes

Classes are blueprints for JavaScript objects. They instruct a JavaScript interpreter what variables and methods are available.

```
class Animal{
  constructor(name){this.name=name;}

  move(){…}
  eat(){…}
  sleep(){…}
}
```

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Object Oriented Programming

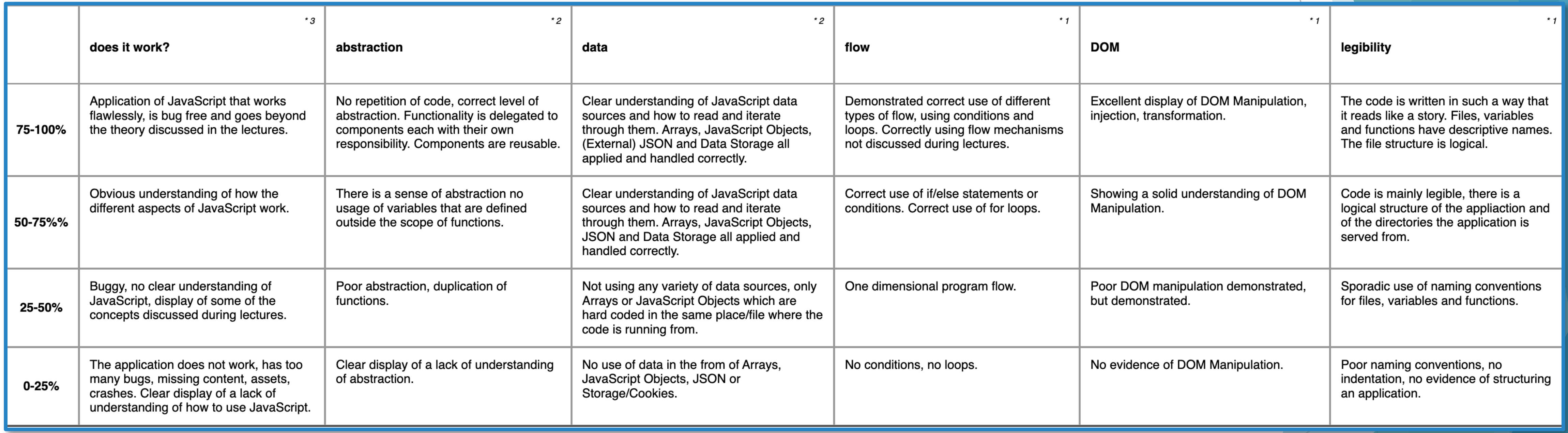
- Principles of OOP
  - Abstraction
  - Inheritance
  - Encapsulation
  - Polymorphism
  - Method Overloading / Overriding

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Grading

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Grading 1st Semester



	* 3 **does it work?**	* 2 **abstraction**	* 2 **data**	* 1 **flow**	* 1 **DOM**	* 1 **legibility**
75-100%	Application of JavaScript that works flawlessly, is bug free and goes beyond the theory discussed in the lectures.	No repetition of code, correct level of abstraction. Functionality is delegated to components each with their own responsibility. Components are reusable.	Clear understanding of JavaScript data sources and how to read and iterate through them. Arrays, JavaScript Objects, (External) JSON and Data Storage all applied and handled correctly.	Demonstrated correct use of different types of flow, using conditions and loops. Correctly using flow mechanisms not discussed during lectures.	Excellent display of DOM Manipulation, injection, transformation.	The code is written in such a way that it reads like a story. Files, variables and functions have descriptive names. The file structure is logical.
50-75%%	Obvious understanding of how the different aspects of JavaScript work.	There is a sense of abstraction no usage of variables that are defined outside the scope of functions.	Clear understanding of JavaScript data sources and how to read and iterate through them. Arrays, JavaScript Objects, JSON and Data Storage all applied and handled correctly.	Correct use of if/else statements or conditions. Correct use of for loops.	Showing a solid understanding of DOM Manipulation.	Code is mainly legible, there is a logical structure of the appliaction and of the directories the application is served from.
25-50%	Buggy, no clear understanding of JavaScript, display of some of the concepts discussed during lectures.	Poor abstraction, duplication of functions.	Not using any variety of data sources, only Arrays or JavaScript Objects which are hard coded in the same place/file where the code is running from.	One dimensional program flow.	Poor DOM manipulation demonstrated, but demonstrated.	Sporadic use of naming conventions for files, variables and functions.
0-25%	The application does not work, has too many bugs, missing content, assets, crashes. Clear display of a lack of understanding of how to use JavaScript.	Clear display of a lack of understanding of abstraction.	No use of data in the from of Arrays, JavaScript Objects, JSON or Storage/Cookies.	No conditions, no loops.	No evidence of DOM Manipulation.	Poor naming conventions, no indentation, no evidence of structuring an application.

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Data Storage

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Data Storage
Client-side

Browsers allow you to store data on the client. There are three types of storage available to us:

Cookies

- Have an expiry date

Session Storage

- Gets discarded when the browser is closed

Local Storage

- Persist for longer (until local storage is cleared)

Cookies

A cookie is a string containing a semicolon-separated list of all cookies (i.e. key=value pairs). Note that each *key* and *value* may be surrounded by whitespace.

Attributes:

- ;path=path
- ;domain=domain
- ;max-age=max-age-in-seconds
- ;expires=date-in-GMTString-format
- ;secure
- ;samesite

https://developer.mozilla.org/en-US/docs/Web/API/Document/cookie accessed 3 Nov 2024

Cookies

```
function setCookie(cookieKey, cookieValue, exdays) {
    const cookieDate = new Date();
    const expiryDaysInMs = exdays * 24 * 60 * 60 * 1000;
    cookieDate.setTime(cookieDate.getTime() + expiryDaysInMs);
    let expires = "expires="+ cookieDate.toUTCString();
    document.cookie = cookieKey + "=" + cookieValue + ";" + expires +
";path=/";
}

setCookie('first_name', 'Jim', 30);
```

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Cookies

```
function getCookie(cookieKey) {
  let name = cookieKey + "=";
  let myCookie = document.cookie.split(';');
  for(let i = 0; i < myCookie.length; i++) {
    let theCookie = myCookie[i];
    while (theCookie.charAt(0) == ' ') {
       theCookie = theCookie.substring(1);
    }
    if (theCookie.indexOf(name) == 0) {
      return theCookie.substring(name.length, theCookie.length);
    }
  }
  return "";
}


getCookie('first_name') // -> 'Jim'
```

Session Storage

- Whenever a document is loaded in a particular tab in the browser, a unique page session gets created and assigned to that particular tab. That page session is valid only for that particular tab.
- A page session lasts as long as the tab or the browser is open and survives over page reloads and restores.
- **Opening a page in a new tab or window creates a new session with the value of the top-level browsing context, which differs from how session cookies work.**
- Opening multiple tabs/windows with the same URL creates sessionStorage for each tab/window.
- Duplicating a tab copies the tab's sessionStorage into the new tab.
- Closing a tab/window ends the session and clears objects in sessionStorage

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Session Storage

```
sessionStorage.setItem("lastname", "Smith");
sessionStorage.getItem("lastname");
sessionStorage.removeItem(<key>); // sessionStorage.removeItem("lastname");
sessionStorage.clear();
```

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Local Storage

- Accessible via the Window interface in JavaScript.
- `window.localStorage` is a read-only property that returns a reference to the local storage object used to store data that is only accessible to the origin that created it.
- Stores data in key-value pairs as strings

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Local Storage

Local Storage has 5 methods

- `setItem()`: Add key and value to `localStorage`
- `getItem()`: This is how you get items from `localStorage`
- `removeItem()`: Remove an item by key from `localStorage`
- `clear()`: Clear all `localStorage`
- `key()`: Pass an index to retrieve the key of a `localStorage`

Local Storage

Storing Content

An example of storing data in local storage with JavaScript is:

```
window.localStorage.setItem('name', 'John Doe');
```

`localStorage` only wants Strings. You can store an object in `localStorage`, but you'll have to convert it to a string first. JSON has a function for that.

```
let person = { name: "John", age: 28 };
window.localStorage.setItem('person', JSON.stringify(person));
// '{"name": "John", "age": "28" }'
```

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Local Storage

Retrieving Content

An example of retrieving data in local storage with JavaScript is:

```
window.localStorage.getItem('name');


JSON.parse(window.localStorage.getItem('person'));
```

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Try it yourself

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Thank You

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin