Data Parsing

- Introduction to JSONs
- Reading/Writing to JSONs
- JSON for Web Development
- Session Storage Overview
- Accessing Stored Values Between Pages

J A V A

S C R I P T

O B J E C T

N O T A T I O N

Example of a JSON

- ~~Introduction to JSONs~~
- Reading/Writing to JSONs
- JSON for Web Development
- Session Storage Overview
- Accessing Stored Values Between Pages

Opening JSON

Parses the file '*data.json*' and stores the resulting object in variable *data*

```
$.getJSON( "data.json", function( data ) {} )
```

Robust: if 'data.json' is located and can be opened, function(data) is called. Otherwise getJSON() terminates and passes over function(data).

Reading JSON

What is printed as a result of running the code?

```
{
    "orders": [
        {
            "orderno": "740745375",
            "date": "June 30, 2088 1:54:23: AM",
            "tracking": "IN039291",
            "custid": "11045",
            "customer": [
                {
                    "custid": "11045",
                    "fname": "Bob",
                    "lname": "Ross",
                    "address": "5032 Forbes Ave",
                    "city": "Pittsburgh",
                    "state": "PA",
                    "zip": "15213"
                }
            ]
        }
    ]

}
```

```
$.getJSON( “data.json", function( data ) {

    var name = data[“orders”][0][“customer”][0][“fname”];
    name += “ “;
    name = data[“orders”][0][“customer”][0][“lname”];
    console.log(name);

});
```

Reading JSON

The '*customer*' field is an array of objects, which is why we access the [0] index before ["fname"]


```
$.getJSON( "data.json", function( data ) {

    var name = data["orders"][0]["customer"][0]["fname"];
    name += " ";
    name = data["orders"][0]["customer"][0]["lname"];
    console.log(name);

});
```

Writing JSON

Writing is the opposite of reading, we assign the field a variable rather than a variable the field.


```
$.getJSON( “data.json", function( data ) {

    data[“orders”][0][“customer”][0][“fname”] = “guy”;
    data[“orders”][0][“customer”][0][“lname”] = “fieri”;

});
```

- ~~Introduction to JSONs~~
- ~~Reading/Writing to JSONs~~
- JSON for Web Development
- Session Storage Overview
- Accessing Stored Values Between Pages

Benefits of using JSONs

- Great for storing large amounts of data
- Easy to parse
- Sounds like “Jason”, so you won’t forget it

Downsides of using JSONs

- Comparatively slow read-write times
- Easily breachable (don’t store personal data in it).

Scanning JSON

Separates each value in the upper-level region of *data* into key-and-value pairs

```
$.each( data, function( key, val ) {});
```

Scanning JSON

What will be printed as a result?

```
{
    "orders": [
        {
            "orderno": "740745375",
            "date": "June 30, 2088 1:54:23: AM",
            "tracking": "IN039291",
            "custid": "11045",
            "customer": [
                {
                    "custid": "11045",
                    "fname": "Bob",
                    "lname": "Ross",
                    "address": "5032 Forbes Ave",
                    "city": "Pittsburgh",
                    "state": "PA",
                    "zip": "15213"
                }
            ]
        }
    ]
}
```

```
$.getJSON( “data.json", function( data ) {

    $.each( data, function( key, val ) {

        console.log(val);
    });

});
```

‘orders’ is the only upper-level data entry, so:

key = ‘orders’

val = (everything between […])

Scanning JSON

```
{
    "video-01": ["cool", "awesome", "nice", "happy"],
    "video-02": ["sad", "depressed", "dark"],
    "video-03": ["confusing", "long"]
}
```

We can use JSONs to store related terms for element tags and retrieve them if a search matches one of its terms.

```
$.getJSON( “data.json", function( data ) {

    $.each( data, function( key, val ) {

        for( var k = 0; k < val.length; k++ ) {
            if( val[k] == tag )
                console.log( key );
        }
    });

});
```

- ~~Introduction to JSONs~~
- ~~Reading/Writing to JSONs~~
- ~~JSON for Web Development~~
- Session Storage Overview
- Accessing Stored Values Between Pages

Session Storage Library

Store item ***click-count*** with value 0

```
sessionStorage.setItem(”click-count”, 0)
```

Retrieve item ***click-count***

```
sessionStorage.getItem(”click-count")
```

Check that item ***click-count*** is set

```
sessionStorage.getItem(”click-count") != null
```

Clear item ***click-count***

```
sessionStorage.removeItem(”click-count")
```

Session Storage Library

Keeping track of clicks with sessionStorage:

```
$('body').click( function () {

    if ( sessionStorage.getItem("click-count") != null ) {
            var count = sessionStorage.getItem("click-count");
            sessionStorage.setItem("click-count", count++);
    }
    else {
            sessionStorage.setItem("click-count", 0);
    }

});
```

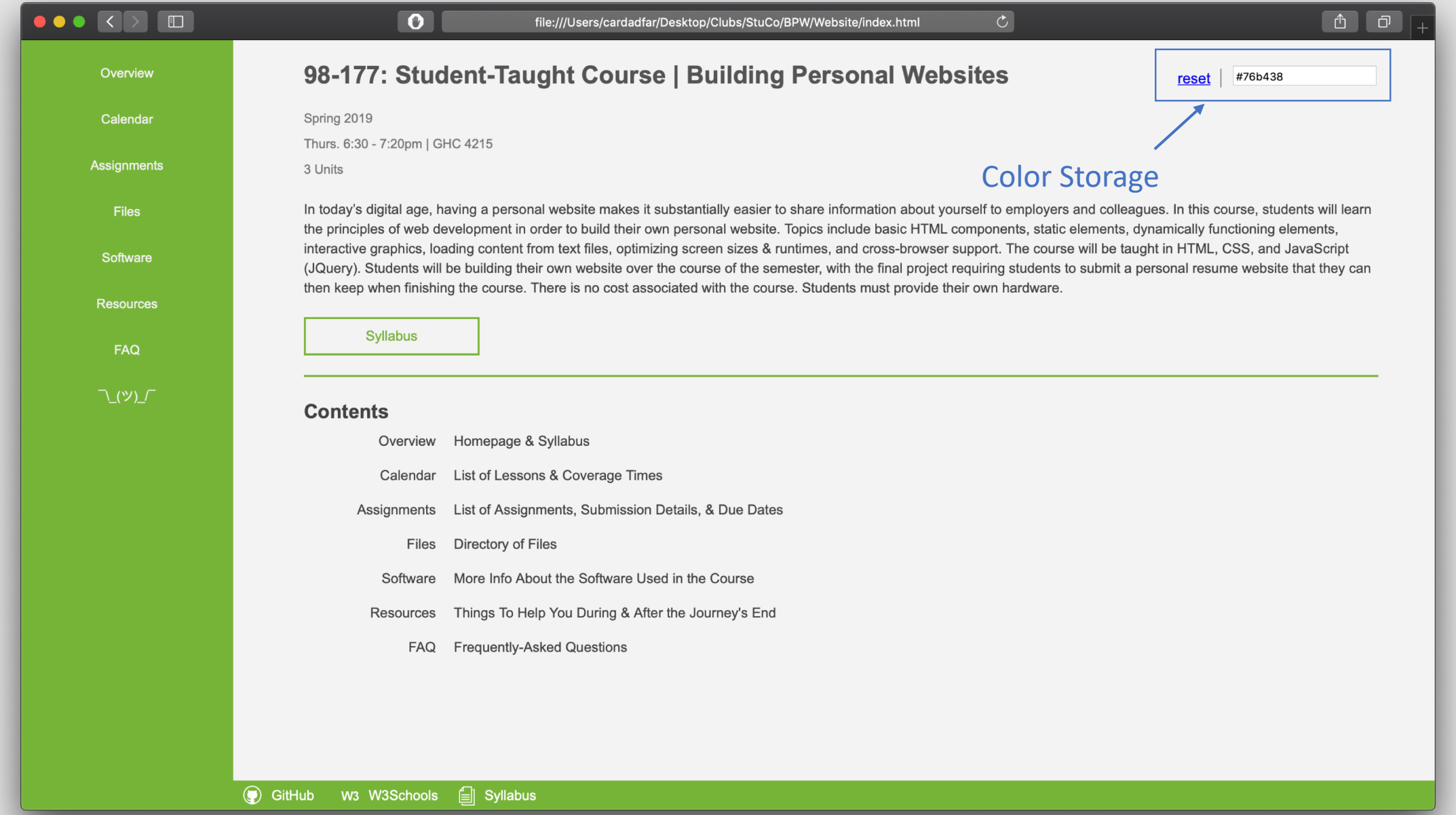
What is sessionStorage?

- A simple way of storing values in JavaScript for repetitive use.
- Optimal use is when wanting to share values between pages.

Why can't I use a global variable instead?

- Global variables are reset each time the page is reloaded.
- sessionStorage values are saved in the browser's cache and are not cleared until either:
  - The browser's cache is cleared
  - The browser is closed.

- ~~Introduction to JSONs~~
- ~~Reading/Writing to JSONs~~
- ~~JSON for Web Development~~
- ~~Session Storage Overview~~
- Accessing Stored Values Between Pages


file:///Users/cardadfar/Desktop/Clubs/StuCo/BPW/Website/index.html
Overview
Calendar
Assignments
Files
Software
Resources
FAQ
¯_(ツ)_/¯
98-177: Student-Taught Course | Building Personal Websites
reset
#76b438
Color Storage
Spring 2019
Thurs. 6:30 - 7:20pm | GHC 4215
3 Units
In today's digital age, having a personal website makes it substantially easier to share information about yourself to employers and colleagues. In this course, students will learn the principles of web development in order to build their own personal website. Topics include basic HTML components, static elements, dynamically functioning elements, interactive graphics, loading content from text files, optimizing screen sizes & runtimes, and cross-browser support. The course will be taught in HTML, CSS, and JavaScript (JQuery). Students will be building their own website over the course of the semester, with the final project requiring students to submit a personal resume website that they can then keep when finishing the course. There is no cost associated with the course. Students must provide their own hardware.
Syllabus
Contents
Overview Homepage & Syllabus
Calendar List of Lessons & Coverage Times
Assignments List of Assignments, Submission Details, & Due Dates
Files Directory of Files
Software More Info About the Software Used in the Course
Resources Things To Help You During & After the Journey's End
FAQ Frequently-Asked Questions
GitHub
W3 W3Schools
Syllabus

```
/*------------------COLOR-------------------*/

var colorWell;
var curColor;
var defaultColor;
if(sessionStorage.getItem("bg_color") != null) {
    defaultColor = sessionStorage.getItem("bg_color");
}
else {
    defaultColor = '#76b438';
}
curColor = defaultColor;
-----------------------------------------------------------------

window.addEventListener("load", startup, false);
function startup() {
    $(':root').css("--purple", defaultColor);
    colorWell = document.querySelector("#colorWell");
    if(colorWell != null) {
        colorWell.value = defaultColor;
        colorWell.addEventListener("change", updateAll, false);
    }

    $(".tab").css("transition", "all 0.25s");

    var footer_hgt = $(".footer").position().top;
    var window_hgt = $(window).height();
    if(footer_hgt < window_hgt) {
        diff = window_hgt - footer_hgt;
        $(".footer").css("margin-top", diff);
    }
-----------------------------------------------------------------

}
function updateAll(event) {
    curColor = event.target.value;
    $(':root').css("--purple", curColor);
    sessionStorage.setItem("bg_color", curColor);
}
-----------------------------------------------------------------

function resetColor() {
    curColor = defaultColor;
    colorWell.value = curColor;
    $(':root').css("--purple", curColor);
    sessionStorage.setItem("bg_color", curColor);
}
```

Instantiate Variables

Set sessionStorage

Create Color Selector on Statrtup

Update Color (Color Selector Event Handler)

Reset Color (Reset Button Event Handler)

```
3  /*------------------COLOR-------------------*/
4
5  var colorWell;
6  var curColor;
7  var defaultColor;
8  if(sessionStorage.getItem("bg_color") != null) {
9      defaultColor = sessionStorage.getItem("bg_color");
10 }
11 else {
12     defaultColor = '#76b438';
13 }
14 curColor = defaultColor;
15
16
17 window.addEventListener("load", startup, false);
18 function startup() {
19     $(':root').css("--purple", defaultColor);
20     colorWell = document.querySelector("#colorWell");
21     if(colorWell != null) {
22         colorWell.value = defaultColor;
23         colorWell.addEventListener("change", updateAll, false);
24     }
25
26     $(".tab").css("transition", "all 0.25s");
27
28     var footer_hgt = $(".footer").position().top;
29     var window_hgt = $(window).height();
30     if(footer_hgt < window_hgt) {
31         diff = window_hgt - footer_hgt;
32         $(".footer").css("margin-top", diff);
33     }
34
35 }
36 function updateAll(event) {
37     curColor = event.target.value;
38     $(':root').css("--purple", curColor);
39     sessionStorage.setItem("bg_color", curColor);
40 }
41
42 function resetColor() {
43     curColor = defaultColor;
44     colorWell.value = curColor;
45     $(':root').css("--purple", curColor);
46     sessionStorage.setItem("bg_color", curColor);
47 }
```

If shared between HTML pages, each page can access bg_color variable

Homework Ideas

- Use JSONs to create searchable keywords for HTML elements.
- Use JQuery to toggle between elements of keywords similar to was inputted in a searchbar.
- Store values from your site into a JSON and have JQuery load them on startup.

Live Demo