More Javascript

cs380

(yes, javascript is the third PL — not counting html)

G Towell
Oct 1

Presenations

- Andy -- PHP single and double quotes
  - https://docs.google.com/presentation/d/1Jvlp8HFY_KIk8nXnkd4hEhDUH6k1Wlo5ISEtBlKVfFQ/edit?usp=sharing
- Joseph -- CSS and why is is so named
  - https://docs.google.com/presentation/d/12a8ioYQJanoQyEhJlKtJglUNbNwq_zPtsTSAxRRpUh4/edit?usp=sharing
- Nisha -- get and post
- Dylan -- Typescript

GET and POST

Nisha Choudhary

HTTP Methods

- Used for communication between clients and servers
- 7 types –
  - GET
  - POST
  - PUT
  - HEAD
  - DELETE
  - PATCH
  - OPTIONS
- Using HTTPS does not make them more secure

GET

- Used to request data from a specified resource
- Query string is seen in the URL*

  Example – /test/demo_form.php?name1=value1&name2=value2

* Example from w3schools

POST

- Used to send data to a server to create or update a resource
- Example

  POST /test/demo_form.php HTTP/1.1

  Host: w3schools.com

  name1=value1&name2=value2
- Data will be resubmitted upon reloading a page or going back
  - An alert to the user that data is about to be resubmitted (browser alert)

* Example from w3schools

Comparison

	GET	POST
Stored in cache	Yes	No
Visibility	In the query of a URL	Not visible
Browser History	Remains in browser history	Not found in browser history
Able to be bookmarked	Yes	No
Length	Restricted	Unlimited
Data Type Allowed	ASCII characters only	ASCII characters & Binary

Sources

- w3schools
- Medium Post

Typescript

Dylan Soemitro

Typescript vs Javascript

- Extension of Javascript
- Javascript has **no type system**
  - provides primitive types like string,number, object etc., but it doesn't check assigned values
  - Does not support classes/other object-oriented features
  - Dynamically typed

Benefits of using Typescript

- Use of a type system
  - Readability
  - Error-checking
- Static typing
- Supports object oriented programming features
  - Enums, data types, classes
- Compiles into simple Javascript

Typescript

```typescript
class Greeter {
    greeting: string;
    constructor (message: string) {
        this.greeting = message;
    }
    greet() {
        return "Hello, " + this.greeting;
    }
}
```

Javascript

```javascript
var Greeter = (function () {
    function Greeter(message) {
        this.greeting = message;
    }
    Greeter.prototype.greet = function () {
        return "Hello, " + this.greeting;
    };
    return Greeter;
})();
```

Sources

- https://www.tutorialsteacher.com/typescript/typescript-overview#:~:text=TypeScript%20is%20an%20open%2Dsource,that%20compiles%20to%20plain%20JavaScript.
- https://www.pcmag.com/encyclopedia/term/type-system
- https://stackoverflow.com/questions/12694530/what-is-typescript-and-why-would-i-use-it-in-place-of-javascript
- https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html
- https://stackoverflow.com/questions/1517582/what-is-the-difference-between-statically-typed-and-dynamically-typed-languages

Functions

- function name(args) { ...}
  - creates a “name” in global namespace
  - then name(args) to use it.
- Anonymous functions
  - var name = function(args) { ... };
    - effectively the same as function name(args) {...}
    - But this means you can put a function anywhere (e.g., inside array or object ...)
    - Recursion on these can be done, but harder
- Optional Args (also known as default args)
  - function aa(dd, ee, ff=‘gg’, ii=7)
  - optional args must be after required args
    - aa(1,2); or aa(1,2,3); or aa(1,2,3,4);
    - filled in order
- rest args
  - function sum(...args) { for (aa of args) { } }
    - that is args is an array holding all (or the remaining) args

Arrays

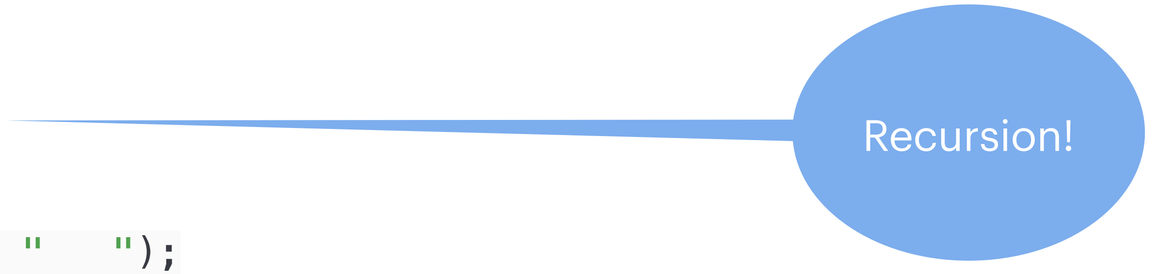
- creation:
  - a= new Array();
    - a[1]=1;
    - a.push(“dog”);
  - a=[1, “dog”, 2, “cat”]; // yes you can mix types in an array.
- access:
  - a[1], a[2], ...
- length: a.length
- arrays cannot have “holes”
  - no preallocation of space
  - Nothing like java int[] aa = new int[10];
- Iteration
  - for (var i = 0; i < a.length; i++) {
  - for (currentValue of a) {
    - currentValue has the values
  - for (index in a) {
    - index has indices
  - a.forEach(function(currentv, idx, arr) { ... });
- 2 d — or more!
  - a[3]=[1,2,3,4]

see array.html for examples

Array Example

File: array2.html

```
<html>
<head>
  <title>Arrays</title>
</head>
<body>
    <form action="javascript:builder()">
    <table>
      <tr><td colspan="2"><center><button id="mbutton">Log it!</button></center></td></tr>
    </table>
    </form>
    <script>
        function docArray(arr, indnt) {
            for (aa of arr) {
                if (Array.isArray(aa)) {
                    docArray(aa, indnt + "   ");
                } else {
                    document.write(indnt+aa+"<br>");
                }
            }
        }
        function builder() {
            a = [[1], [1,2], [1,2,3], 4, [1,2,3,4,[1,2,3]]];
            a.forEach(function(cv, idx, arr) {
                console.log("FUNCC " + idx + " " + cv);
            });
            docArray(a, "");
        }
    </script>
    </body>
    </html>
```



Demo in chrome.

See effect on DOM

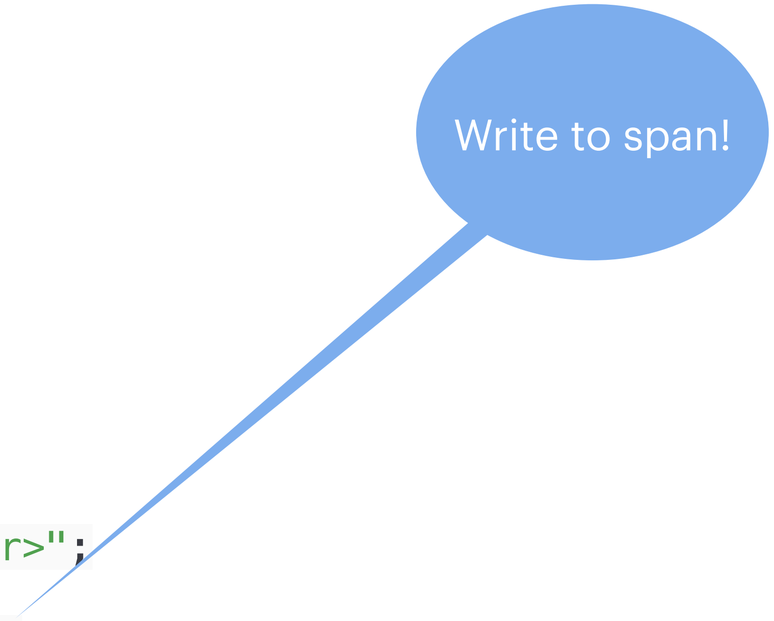
Objects

- At their simplest, equivalent to Java Maps, Python dictionaries, ...
- Creation:
  - var obj = {};
  - var obj=new Object();
- Setting
  - at creation: obj = { f:"FFF", g:"gggg"};
  - other times obj["f"]="ffff";
    - or obj.f="fff";
- Iteration
  - for (key in obj) { ... }
- Arrays are special objects so
  - for(key in array) {} works for arrays
  - for (ob of obj) {} does NOT work for objects

- Javascript objects and Object Oriented Programing

Object Example

```
<html>
<head>
  <title>Objects</title>
</head>
<body>
    <form action="javascript:builder()">
    <table>
      <tr><td colspan="2"><center><button id="mbutton">Log it!</button></center></td></tr>
    </table>
    </form>
    <span id="spanned"></span>
    <script>
        var trial=0;
        function builder() {
            trial= trial+1;
            let string="Trial " + trial + "<br>";
            o = {f:"ff", g:"ggg"};
            o.h="hhhh";
            o["i"]="iiiii";
            string = string + "f:" + o.f + "<br>";
            string = string + "h:" + o["h"] + "<br>";
            for (k in o) {
                string = string + "obj " + k + " " + o[k] + "<br>";
            }
            document.getElementById("spanned").innerHTML=string;
        }
    </script>
    </body>
    </html>
```



Homework 4

Due Oct 25

Groups of 2 -3 but can do it solo

Let me know groups by Oct 11

Implement a game in Javascript (mostly)

4 is a crowd — show video — discuss underlying javascript

Other games on my approval

Multiplication Table, v1

```
<html> <head>
  <title>Multiplication Table</title>
  <script type="text/javascript">
    var rows = prompt("How many rows for your multiplication table?");
    var cols = prompt("How many columns for your multiplication table?");
    if(rows == "" || rows == null)
       rows = 10;
    if(cols== "" || cols== null)
       cols = 10;
    createTable(rows, cols);
    function createTable(rows, cols) {
      var j=1;
      var output = "<table border='1' width='500' cellspacing='0'cellpadding='5'>";
      for(i=1;i<=rows;i++) {
        output = output + "<tr>";
        while(j<=cols) {
          output = output + "<td>" + i*j + "</td>";
          j = j+1;
        }
        output = output + "</tr>";
        j = 1;
      }
      output = output + "</table>";
      document.write(output);
    }
  </script>
</head>
<body>
</body></html>
```

Javascript directly asks user, in practice this is almost never used

Gratuitous function

document is the javascript variable for the enclosing page. This will overwrite everything

NO BODY!

Multiplication Table, v2, part 1

```
<html>
<head>
  <title>Multiplication Table</title>
</head>
<body>
    <table>
      <tr><td>Number of Rows</td><td><input type="text" id="rows" name="cols"></td></tr>
      <tr><td>Number of columns</td><td><input type="text" id="cols" name="cols"></td></tr>
      <tr><td colspan="2"><center><button id="mbutton">Make table</button></center></td></tr>
    </table>
    <table id="mytable">
        <tr><td>Hello</td></tr>
        <tr><td>Goodbye</td></tr>
    </table>
```

Multiplication Table v2, part 2

Set handler for onclick event
Could have done in html

```
  <script type="text/javascript">

document.getElementById("mbutton").onclick = function() {
    let rows = document.getElementById("rows").value;
    let cols = document.getElementById("cols").value;
    console.log("HHH " + rows + "  " + cols);
    multable(rows, cols)
};

function multable(rows, cols) {
    if(rows == "" || rows == null)
        rows = 10;
    if(cols== "" || cols== null)
        cols = 10;
    fillTable(rows, cols);
}
```

Building table one element at a time and adding each to DOM.

```
function fillTable(rows, cols)
{
    // first remove all old rows
    var table = document.getElementById("mytable");
    table.innerHTML = "";
    // then create new rows and fill them
    for (rr=0; rr<=rows; rr++) {
        let row = document.createElement("tr");
        table.append(row);
        for (cc=0; cc<=cols; cc++) {
            if (rr==0) {
                let th = document.createElement('th');
                th.innerHTML = ""+(cc==0?"":cc);
                row.appendChild(th);
            } else {
                if (cc==0) {
                    let th = document.createElement('th');
                    th.innerHTML = ""+rr;
                    row.appendChild(th);
                } else {
                    let td = document.createElement("td");
                    td.innerHTML = ""+(rr*cc);
                    row.appendChild(td);
                }
            }
        }
    }
}
```

Multiplication Table, v3

Just the diffs

```
<body>

    <form action="javascript:builder()">
    <table>
      <tr><td>Number of Rows</td><td><input type="text" id="rows" name="cols"></td></tr>
      <tr><td>Number of columns</td><td><input type="text" id="cols" name="cols"></td></tr>
      <tr><td colspan="2"><center><button id="mbutton">Make table</button></center></td></tr>
    </table>
    </form>
    <table id="mytable">
        <tr><td>Hello</td></tr>
        <tr><td>Goodbye</td></tr>
    </table>

  <script type="text/javascript">

function builder() {
    let rows = document.getElementById("rows").value;
    let cols = document.getElementById("cols").value;
    console.log("HHH " + rows + "  " + cols);
    multable(rows, cols)
};
```

rather than giving an "onclick" function to the button, put the button into a form and point the form at a JS function

the function executed by the form action. It is otherwise identical to the onclick

Multiplication Table, v4

just do it on the server side!

```
<html><head> <title>Multiplication Table</title></head>
<body>
    <form action="mtable.php" type="post">
    <table>
      <tr><td>Number of Rows</td><td><input type="text" id="rows" name="rows"></td>
      <tr><td>Number of columns</td><td><input type="text" id="cols" name="cols"></
      <tr><td colspan="2"><center><button id="mbutton">Make table</button></center>
    </table></form></body></html>
```

```
<?php
function filltable() {
    $rows = $_REQUEST["rows"];
    $cols = $_REQUEST["cols"];
    for ($r=0; $r<=$rows; $r++) {
        echo("<tr>");
        if ($r==0) {
            echo("<th></th>");
        } else {
            echo("<th>$r</th>");
        }
        for ($c=1; $c<=$cols; $c++) {
            if ($r==0) {
                echo("<th>$c</th>");
            } else {
                echo("<td>". ($r*$c) . "</td>");
            }}
        echo("</tr>");
    }}
?>
<html><head><title>Multiplication Table</title></head>
<body>
<table>
<?php filltable(); ?>
</table></body></html>
```

Yes, this this a cheat as we are talking about client-side programming. BUT, it is always correct to ask the question “where should this program run”. There is no correct answer.

Pros / Cons?

Checking User Input

- Best done on client side as it saves a lot of hassle and can be very interactive
- Each input has two fields
  - the form element containing the user input
  - an area for feedback to the user
- UI is checked on every keystroke and on button click
- Note that all this is not wrapped in a form.

```
<body>
    <table id="table1">
      <tr>
        <td>First Name:</td>
        <td><input type="text" id="first" onkeyup="validate();" /></
        <td><div id="errFirst"></div></td>
      </tr>
      <tr>
        <td>Last Name:</td>
        <td><input type="text" id="last" onkeyup="validate();"/></td
        <td><div id="errLast"></div></td>
      </tr>
      <!– not showing email, userid, password and confirm password –
     <tr>
        <td><input type="button" id="create"  value="Create"
             onclick="validate();finalValidate();"/></td>
        <td><div id="errFinal"></div></td>
      </tr>
    </table>
</body>
</html>
```

checkform.html

UI Checking
Javascript

```
var divs = ["errFirst", "errLast", "errEmail", "errUid", "errPassword", "errConfirm"];
var names = ['first', 'last', 'email', 'uid', 'password', 'confirm'];
   function validate() {
     var inputs = new Array();
     inputs[0] = document.getElementById(names[0]).value;
     // …
     var errors = new Array();
     errors[0] = "<span style='color:red'>Please enter your first name!</span>";
     // …
     for (i in inputs) {
       var errMessage = errors[i];
       var div = divs[i];
       if (inputs[i] == "")
           document.getElementById(div).innerHTML = errMessage;
       else if (i==2) {
         var atpos=inputs[i].indexOf("@");
         var dotpos=inputs[i].lastIndexOf(".");
         if (atpos<1 || dotpos<atpos+2 || dotpos+2>=inputs[i].length)
           document.getElementById('errEmail').innerHTML =
                 "<span style='color: red'>Enter a valid email address!</span>";
         else
           document.getElementById(div).innerHTML = "OK!";
       }
       // etc
     }}
```

More UI checking

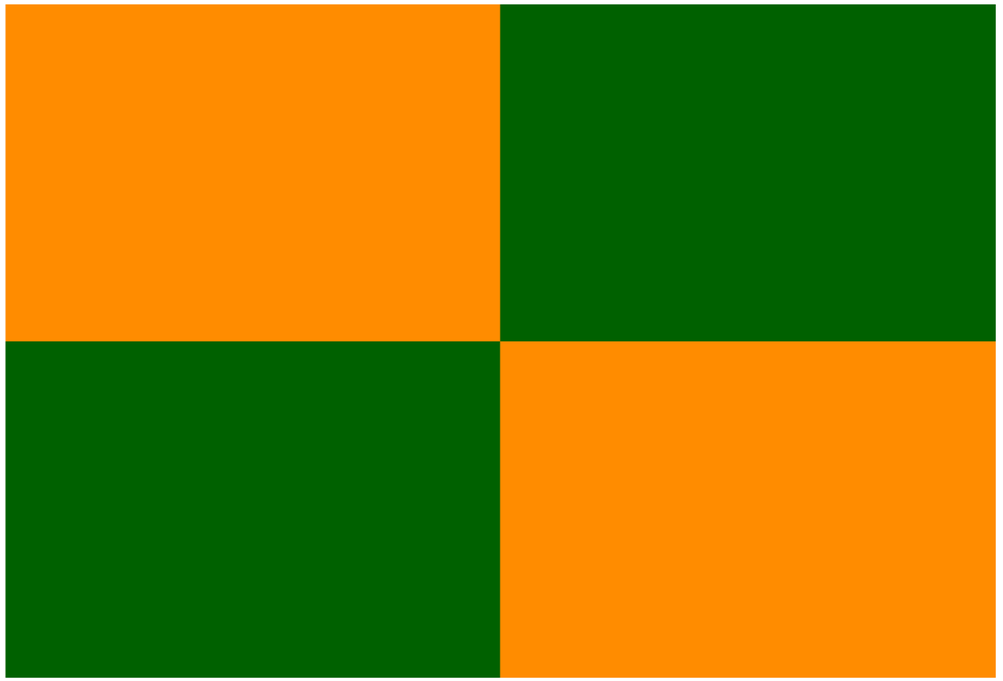
```
function finalValidate()
        {
          var count = 0;
          var params = {};
          for(i=0;i<6;i++)
          {
            var div = divs[i];
            if(document.getElementById(div).innerHTML == "OK!")
                count = count + 1;
            params[names[i]] = document.getElementById(names[i]).value;
          }
          console.log(params);
          if(count == 6) {
                document.getElementById("errFinal").innerHTML =
“All the data you entered is correct!!!";
                post("former.php", params, 'post');
          }
        }
```

```
function post(path, params, method='post') {
            const form = document.createElement('form');
            form.method = method;
            form.action = path;
            for (key in params) {
            if (params.hasOwnProperty(key)) {
                hiddenField = document.createElement('input');
                hiddenField.type = 'hidden';
                hiddenField.name = key;
                hiddenField.value = params[key];
                form.appendChild(hiddenField);
                }
            }
            document.body.appendChild(form);
            form.submit();
            }
```

Checkerboard

from lecture 1

- Remember this?
- Critiques?
  - Only 2x2
  - fixed colors
- Solution
  - forms and javascript!



Recall the HTML

```
<html>
    <style>
        .container { margin-left:5%; width:90%; height:50%; display: flex; }
        .shape {
            margin:0pt; width:50%; height:100%; display: inline-block; padding: 0pt;
        }
        .color1 { background-color:#03335f; }
        .color2 { background-color:#f3b720; }
    </style>
    <body>
        <div class="container">
            <div class="shape color1"></div>
            <div class="shape color2"></div>
        </div>
        <div class="container">
            <div class="shape color2"></div>
            <div class="shape color1"></div>
        </div>
    </body>
</html>
```

Javascript checkerboard

Initial HTML

divs are absolute positions, so they are on top of each other. z-index controls who is on top

```
    <body>
        <div id="thediv" style="background-color: red; margin-top:20%; margin-left:10%; width:80%;
height:80%; display: flex; z-index: -1; position: absolute;">
        </div>
        <div id="thediv" style="margin-top:1%; margin-left:1%; width:98%; height:98%; display:
flex; z-index: 1; position:absolute">
            <table>
                <tr><td>Grid size (1-100)</td><td><input type="text" id="first"
onkeyup="checker();" /></td></tr>
                <tr><td>Color 1</td><td><input type="text" id="color1"/></td></tr>
                <tr><td>Color 2</td><td><input type="text" id="color2"/></td></tr>
                <tr><td colspan=“2” align=“center”><button id="abutton" onclick="setColors();">Set
Colors</button></td></tr>
            </table>
        </div>

    </body>
```

checkerboarding changes on every keystroke

Javascript checkerboard

```
var color1="pink";
var color2="white";
function setColors() {
    if (document.getElementById("color1").value.length > 0)
        color1=document.getElementById("color1").value;
    if (document.getElementById("color2").value.length > 0)
        color2=document.getElementById("color2").value;
    checker();
}
/**
  * takes a number (x) and a precision (digits) and
  * returns a representation with exactly that number of sig digits
  */
function precise(x, digits) {
    return Number.parseFloat(x).toPrecision(digits);
}
```

Javascript checkerboard

```
        function checker() {
             var val = document.getElementById("first").value;
             if (isNaN(val) || (val<0) || (val>100)) return;
             var div = document.getElementById("thediv");
             div.innerHTML="";
             var newHTML = "";
             pct = 100/val;
             for (j=0; j<val; j++) {
                 newHTML=newHTML+"<div style=\"height:100%; width:"+precise(pct,4)+"%;\">";
                 for (k=0; k<val; k++) {
                     color=color1;
                     if ((j+k)%2==0)
                         color=color2;
                     newHTML=newHTML+"<div style=\"background-color:"+color+"; width:100%;
height:"+precise(pct,4)+"%;\"></div>"
                 }
                 newHTML=newHTML+"</div>";
             }
             console.log(newHTML);
             div.innerHTML=newHTML;
         }
```

Idea: build exactly the html that I wrote by hand to make a checkerboard, then put that in the div. Alternately I could have made each element and added the element to the correct spot in the DOM.