CISC 1600 Lecture 3.5
Introduction to JavaScript

Topics:

JavaScript overview

JavaScript syntax

P5.js

JavaScript on the web

What is JavaScript?

- JavaScript is not Java
- JavaScript was designed to add interactivity to HTML pages
  - It has grown beyond that to run outside the browser
- JavaScript is usually interpreted
  - but Google created an open source compiler for it (V8)
- JavaScript is a web standard

What can JavaScript do?

- Insert dynamic text into an HTML page
- Read and write HTML elements
- React to events: mouse, keyboard, network, etc.
- Asynchronously load or update parts of pages
- Validate form data before it is submitted
- Examples for class here

Popup boxes

JavaScript allows interactivity through the use of pop-up boxes

There are 3 general kinds:

```
alert("sometext");
confirm("sometext");
// returns "true" or "false")
prompt("sometext","defaultvalue");
//returns what user enters as answer
```

See 6_PopUps

JavaScript general syntax

- Syntax is much like Processing (or C or Java)
- Statements “should” end with a semicolon ;
- White space does not matter (except for line breaks sometimes)

```
/*****************************
This is JavaScript (JS), the programming language that powers
the web (and this is a comment, which you can delete).
*****************************/
function greetMe(name) {
  var today = new Date();
  alert("Hello " + name + ", today is " + today.toDateString());
}
```

Variables hold values

- Variables are named locations for storing data
- Assign a value to a variable to refer to it later
  - Values can be simple: number, string, boolean
  - Or complex: DOM node, document object, array of other values
- Create a variable: `var count;`
- Assign a value to it: `count = 0;`
- Use it: `newH1 = "...count = " + count + "</h1>";`
- JavaScript in the browser has several pre-defined variables, e.g.,
  - window: the browser window/frame around the page
  - document: the DOM and other page characteristics

Some JavaScript syntax for math, logic, etc.

- Mathematical:
  `+, -, *, /, %, ++, --`
- Logical:
  `&&, ||, !, <, >, etc...`
- Variable declaration and assignment:

```
var x=10;
var y=2;
var z=x/y;
```

- Strings can be concatenated with the '+' sign.

```
var txt1 = "What a very";
var txt2 = "nice day";
var txt3 = txt1 + " " + txt2;
```

- Note: If you add a number and a string, the result will be a string!

JavaScript is object-oriented

- Like Processing, you can define new data types in JavaScript as objects
- An object is a collection of facts and functions
  - Facts: data, the state of the object
  - Functions: the available operations to perform on it
- In JavaScript, both are accessed using a dot '.'
  - Fact: document.children
  - Function: document.getElementById()
- An object can contain simple values or other objects

Selection: The ability to make a choice

- Execute code only if certain conditions are true
- This is mainly the “if” statement in languages

```
if ((count % 3) == 0) {
    target.innerHTML = target.innerHTML +
newH1;
} else {
    target.innerHTML = newH1;
}
```

Repetition: The ability to repeat an action

- Repeatedly execute code, modifying state
- In JavaScript: mainly “for” and “while” loops

```
var stars = "";
for (var i=0; i < count; i++) {
  stars = stars + "*";
}
// OR
while (stars.length < count) {
  stars = stars + "*";
}
```

Functions: reuse operations

- JavaScript implements the procedural paradigm
  - Because you can write and use procedures (functions)
- Functions take input arguments & return outputs

```
function replaceInnerHtml(nodeId, html) {
  var target = document.getElementById(nodeId);
  target.innerHTML = html;
  return target;
}
```

- Functions allow code to be more modular and reusable
- Debug a function once, use it many times

Arrays are data structures that store multiple values

- Arrays in JavaScript are like arrays in Processing

```
var langs = ['c', 'c++', 'java'];
for (var i=0; i < 3; i++) {
    console.log(langs[i]);
}
```

- Unlike Processing you can put things with different types into the same array in JavaScript

```
var misc = [1, 'a', true];
for (var i=0; i < 3; i++) {
    console.log(misc[i]);
}
```

P5.js is Processing in JavaScript

- Processing syntax is based on Java
- Originally, Processing code was translated to Java
- Processing.js converts Processing Java code into JavaScript to run in browsers
- P5.js implements Processing directly in JavaScript
  - Syntax based on JavaScript instead of Java
- OpenProcessing.org supports both

Processing vs P5 (notes)

- Processing

```
void setup() {
  // setup stuff
}

void draw() {
  // draw stuff
}
```

- P5

```
function setup() {
  // setup stuff
}

function draw() {
  // draw stuff
}
```

Processing vs P5 (notes)

- Processing

```
float max_distance;

void setup() {
  size(640, 360);
  noStroke();
  max_distance = dist(0, 0, width, height);
}

void draw() {
  background(0);

  for(int i = 0; i <= width; i += 20) {
    for(int j = 0; j <= height; j += 20) {
      float size = dist(mouseX, mouseY, i,
j);
      size = size/max_distance * 66;
      ellipse(i, j, size, size);
    }
  }
}
```

- P5

```
var max_distance;

function setup() {
  createCanvas(640, 360);
  noStroke();
  max_distance = dist(0, 0, width, height);
}

function draw() {
  background(0);

  for(var i = 0; i <= width; i += 20) {
    for(var j = 0; j <= height; j += 20) {
      var size = dist(mouseX, mouseY, i, j);
      size = size/max_distance * 66;
      ellipse(i, j, size, size);
    }
  }
}
```

Converting Processing sketch to P5

- (In OpenProcessing.org, switch to P5 mode)
- Change `void` to `function`
- Replace types (`int`, `float`, `boolean`) with `var`
- Remove types from function parameters
- Replace `size()` with `createCanvas()`
- Convert classes to JavaScript syntax
- Changes names of certain variables and functions

Examples

- Examples of converting Processing sketches to P5
- See Collection 05 on
  https://www.openprocessing.org/class/57767/

JavaScript can be defined and used in the body or the head of a web page

```
<html>

    <head>
        <script type="text/javascript">
            function message()
            {
                alert("This alert box called by
onload event");
            }
        </script>
    </head>

    <body onload="message()">
        <script type="text/javascript">
            document.write("Message written by
JavaScript");
        </script>
    </body>

</html>

<!-- See 2_EventListeners-->
```

Interactivity: events and listeners

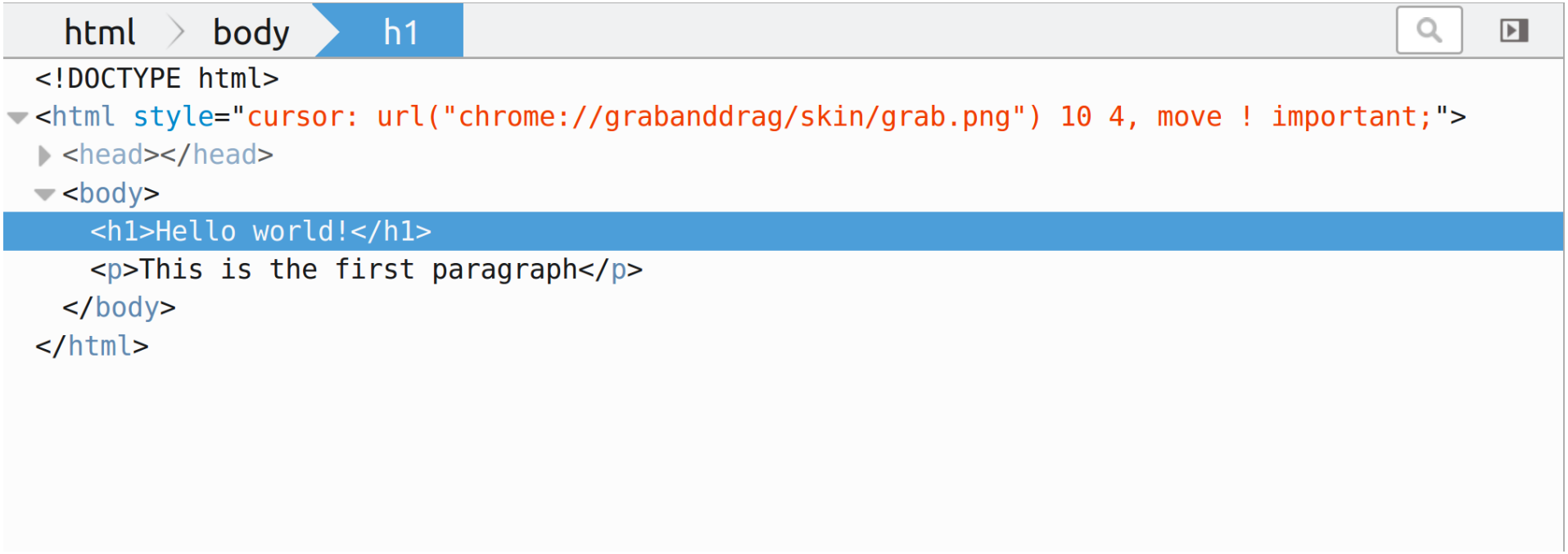
- JavaScript code can respond to “events”
- Events happen in response to
  - User-interface interactions (mouse, keyboard)
  - Network operations (page or resource loading)
- Can register “event listener” functions
  - Called when event happens with information on event

```
<button type="button" onclick="go()">Go!</button>
```

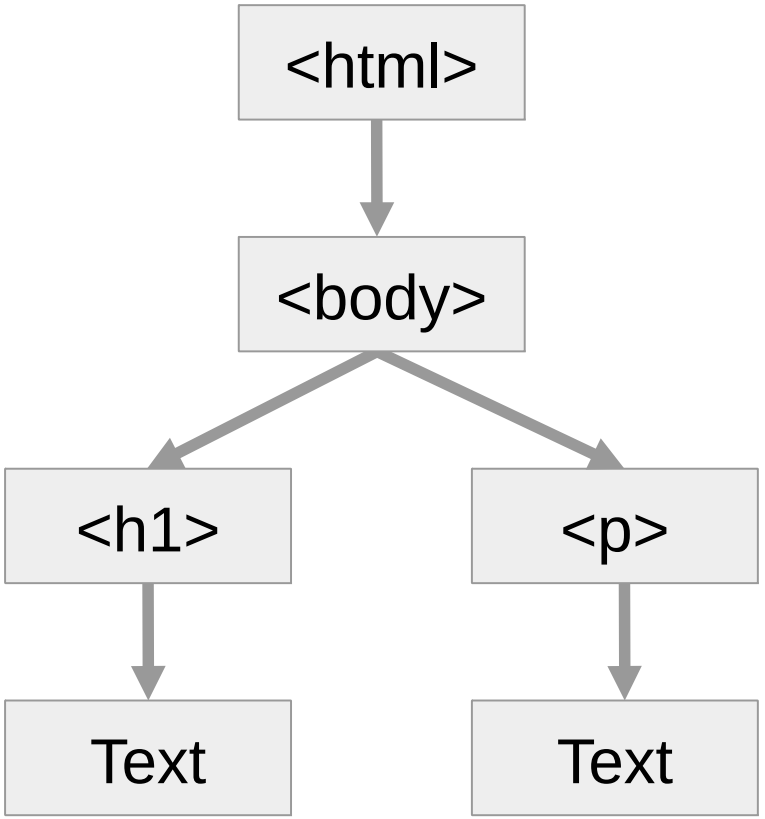
Recall: Document Object Model (DOM)

- A web browser interprets your HTML
  - And builds a model of the page, the DOM
- The DOM is what is rendered to the screen
- The DOM can be manipulated by CSS and JavaScript after it is built
- When building a page, consider its structure first, i.e., the DOM

Example DOM


html
body
h1
<!DOCTYPE html>
<html style="cursor: url("chrome://grabanddrag/skin/grab.png") 10 4, move ! important;">
<head></head>
<body>
<h1>Hello world!</h1>
This is the first paragraph
</body>
</html>

Example DOM



JavaScript can manipulate the DOM

- Several ways to select DOM elements
  - `document.getElementById("idOfElement");`
  - `document.getElementsByTagName("div");`
  - `parent.children[index];`
  - Etc.
- Several ways to modify DOM elements
  - `element.innerHTML = "Some HTML";`
  - `img.src = "newUrl";`
  - `element.style.border = "1pt solid";`

Simple DOM manipulation

```
<html>
<head>
</head>
<body>
    <script type="text/javascript">
            document.write("<h1>Hello World!
</h1>");
    </script>
</body>
</html>

<!-- Note use of document object model -->
<!-- Example Here:  1_FirstJavaScript.html
-->
```