COMPSCI 326 - Web Programming
More CSS and DOM surgery

join on the Slack #q-and-a channel as well as Zoom
remember, you can ask questions of your teammates on your group Slack!
please **turn on your webcam** if you can
mute at all times when you aren't asking a question
(https://docs.google.com/document/d/1ttfMyd_I5wORQusDeshgD8Hw6br-a9t85rRzAFD1MQ0/edit?usp=sharing)

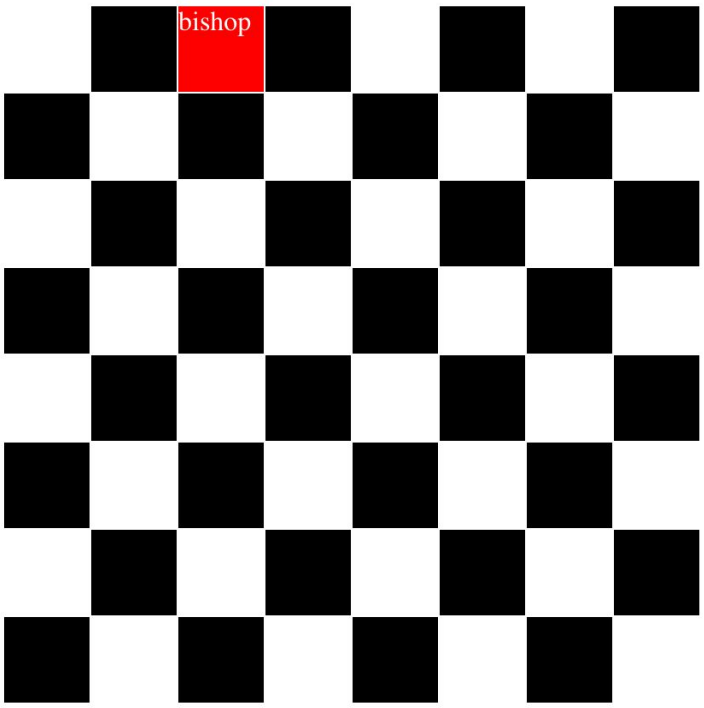
Today: More CSS and DOM surgery

resources:

- CSS (with classes and ids) - CSS: Cascading Style Sheets | MDN
- CSS pseudo-classes (e.g., hover) - Pseudo-classes - CSS: Cascading Style Sheets | MDN
- CSS grids - CSS Grid Layout - CSS: Cascading Style Sheets
- Intro to the DOM - Introduction to the DOM - Web APIs | MDN
- DOM objects - HTML DOM Document Objects
- DOM node tutorial - DOM tree
- onLoad() - onload Event, GlobalEventHandlers.onload - Web APIs | MDN

Exercise today: heading towards a chessboard

- today, programmatically generate the board
    - no "grid items" will be in the HTML
    - you will create them in JavaScript



Last time: CSS:

- ```
  <STYLE>
  p {
    color: green;
  ```
  ```

```
}
.someClassName {
  color : black;
}
#someId {
  color : yellow;
}
.someClassName:hover {
  color: red;
}
</STYLE>
```

- can be used with any HTML element but especially useful with:
  <DIV> = a container, breaks a line (nothing visual, just lets you set style for stuff inside)
  <SPAN> = a container, doesn't break the line (as above)
- grids:
  - put inside a grid container
    - `<div class="grid-container" id="thegrid">`
      `<!-- grid items go in here -->`
      `</div>`
  - each entry is a grid item
    - `<div class="blackSquare grid-item">...</div>`
    - `<div class="whiteSquare grid-item">...</div>`

Then you specify the characteristics of the grid in the STYLE section:

```
.grid-container {
    display: grid;
    grid-template-columns: repeat(2, auto);
    grid-gap: 1px;
    width: 500px;
    height: 500px;
}
.grid-item {
    height: 50px;
    width: 50px;
}
```

So how do we create elements in JavaScript? **DOM SURGERY**

Recall that the DOM is a tree.

- You can **find** individual nodes, **create** nodes, etc.:
  - `const newDiv = document.createElement("div");`
  - `const newTextNode = document.createTextNode("hello");`
- You can read and **update** properties for nodes:
  - `newTextNode.innerText = "howdy"; // also innerHTML to inject HTML -- newTextNode.innerHTML = "<B>HOWDY</B>";`
- All nodes have **classLists** you can modify:
  - `newDiv.classList.add("grid-item"); // now newDiv has the class "grid-item"`
  - `newDiv.classList.add("blackSquare"); // now it has two classes: blackSquare and grid-item`
- Then you can **add** them ("append") to other nodes:
  - `const parent = document.getElementById("thegrid"); // find the grid`
  - `parent.appendChild(newDiv); // add the child`

For example:

- to get the effect of starting with the following HTML:
  `<div id="stuffgoeshere"></div>`

  to

```
<div id="stuffgoeshere">
<p class="somestyle">Hello, <b>world</b></p>
</div>
```

1. find the div element:
   `const theDiv = document.getElementById("stuffgoeshere");`
2. create the element to insert (here, a <P>):
   `const theP = document.createElement("p");`
3. add a class to the element:
   `theP.classList.add("somestyle");`
4. insert HTML into the element:
   `theP.innerHTML = "Hello, <b>world</b>";`
5. finally, add it as a child of the DIV
   `theDiv.appendChild(theP);`

all together:

```
// find the div element:
const theDiv = document.getElementById("stuffgoeshere");

// create the element to insert (here, a <P>):
const theP = document.createElement("p");

// add a class to the element:
theP.classList.add("somestyle");

// insert HTML into the element:
// could also have done this as above, but this is generally
frowned-upon
//   theP.innerHTML = "Hello, <b>world</b>";
// here's how you do this by creating children:
// make two text nodes, Hello, and world
const helloNode = document.createTextNode("Hello, ");
const worldNode = document.createTextNode("world");
// make a boldface container
const bNode = document.createElement("b");
// add in "world", so you get <B>world</B>
bNode.appendChild(worldNode);
```

```
// now add the two children
theP.appendChild(helloNode);
theP.appendChild(bNode);
// finally, add the P as a child of the DIV
theDiv.appendChild(theP);
```

Exercise!

https://docs.google.com/document/d/1WoWb9ARaLlKDqRnsBgzaLo4PYeYlg7uQUDNRdNiCNTg/edit?usp=sharing