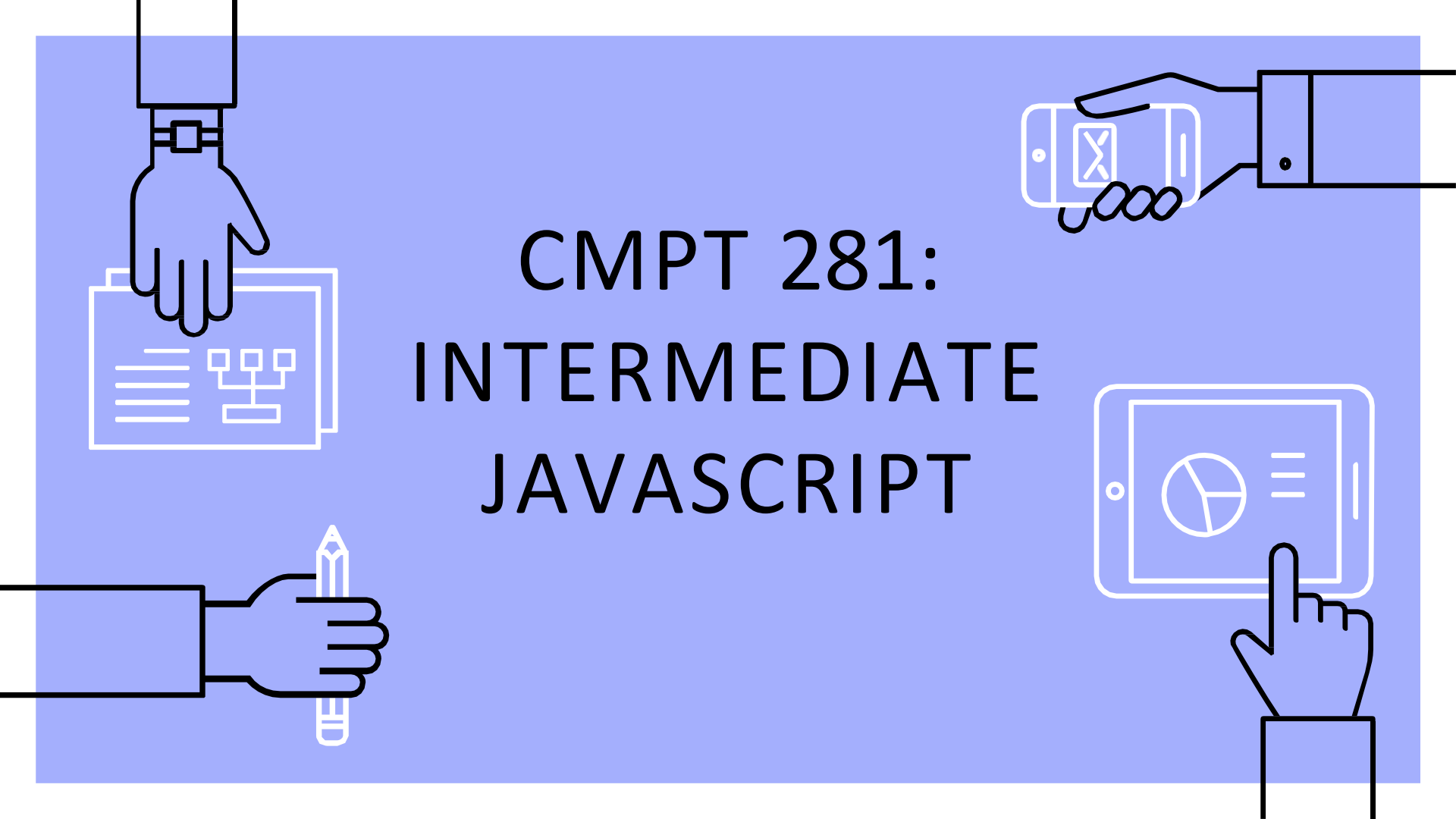
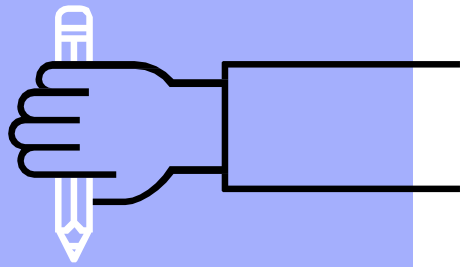
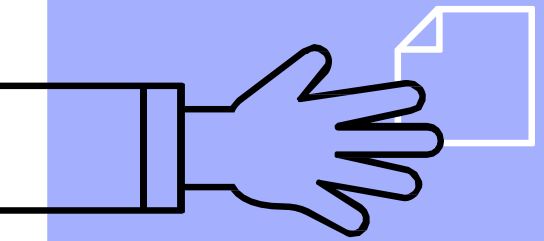

CMPT 281:
INTERMEDIATE
JAVASCRIPT

TODAY'S TOPICS

- Using JavaScript to access and modify HTML elements
- Common JavaScript use cases
- Operators, logical operators, and comparators
- Switches, nested functions, and objects

USING JS TO ACCESS AND MODIFY HTML





ACCESSING HTML ELEMENTS

So far, we have demonstrated ID. Sometimes, we may want to look at a **set** of elements rather than a single element.

Function	Description
document.getElementById(String ID)	Finds an element by ID
document.getElementsByTagName(String name)	Finds elements by tag name
document.getElementsByClassName(String class)	Finds elements by class name

ACCESSING HTML ELEMENTS

Other ways to access HTML elements.

Property	Description
document.anchors	Gets anchor elements in the document
document.body	Gets the body element of the document
document.img	Returns all the img elements of the document
document.scripts	Returns all <script> elements
document.title	Returns the title of the document

CHANGING HTML ELEMENTS

Once we have an HTML element, we can edit its property through an assignment statement.

Property	Description
element.innerHTML	Used to change the inner HTML of an element
element.attribute	Used to change the attribute value of an HTML element
element.style.property	Change the style of an HTML element

USING JAVASCRIPT TO CREATE HTML ELEMENTS

Sometimes you may not want to amend an existing element, but rather, add or remove a HTML element.

Function	Description
document.createElement(element)	Create an HTML element
document.removeChild(element)	Remove an HTML element's child
document.appendChild(element)	Add an HTML element as a child
document.replaceChild(new, old)	Replace an HTML element with another
document.write(text)	Write into the HTML content

HTML EVENTS THAT CAN TRIGGER JAVASCRIPT

Event Name	Event Trigger
onchange	An HTML element has been changed
onclick	A user clicks on an HTML element
onmouseover	The mouse cursor moves over an HTML element
onmouseout	The mouse cursor exits an HTML element
onkeydown	The user inputs a keypress
onload	The webpage has finished loading

TIMED EVENTS

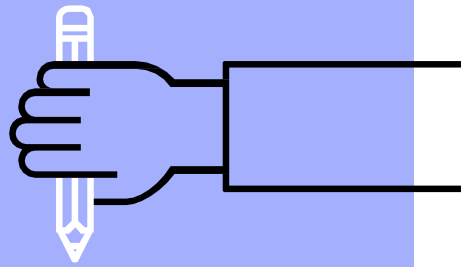
Two key methods for calling things in timed intervals.

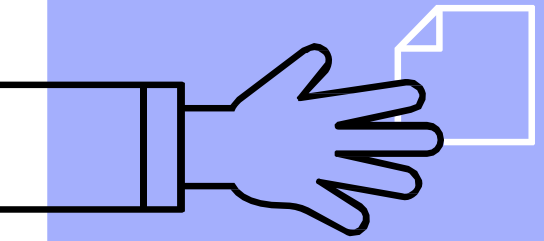
Function	Description
setTimeOut(function, milliseconds)	Executes a function after a delay in milliseconds
setInterval (function, milliseconds)	Executes a function after a delay in milliseconds, and repeats
clearTimeout()	Used to stop a var returned from setTimeOut()
clearInterval()	Used to stop a var returned from setInterval

EDITING BROWSER WINDOWS

Functions	Description
window.open()	Opens a new window
window.close()	Closes the current window
window.moveTo()	Moves the current window
window.resizeTo()	Resizes the current window
window.history.back()	Loads the previous URL in the history list
window.history.forward()	The same thing as clicking forward in a browser

COMMON JAVASCRIPT USE CASES

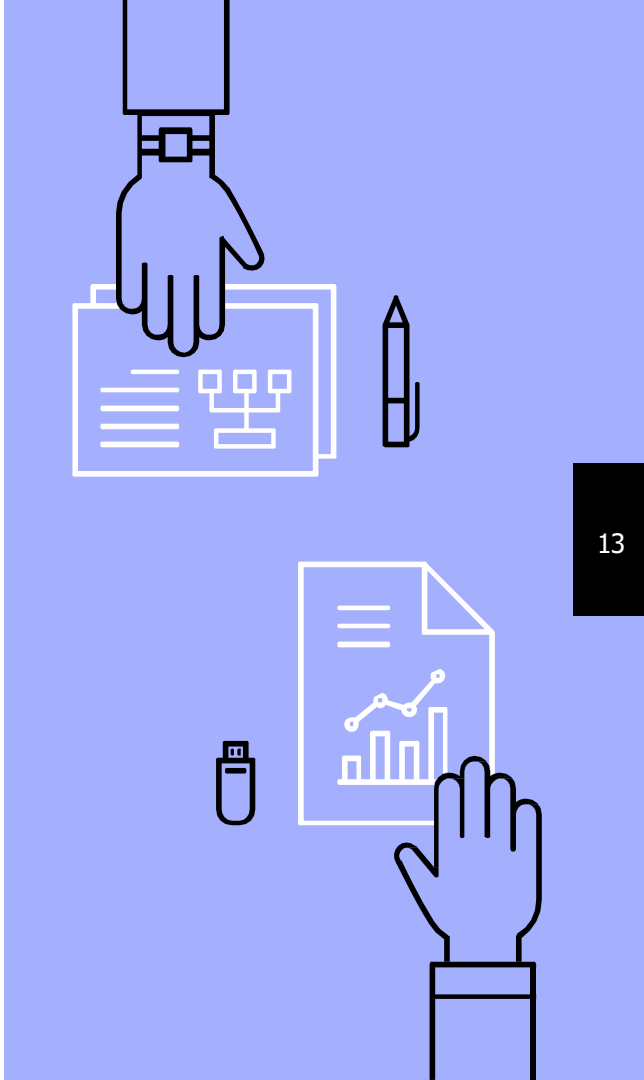




JAVASCRIPT BROWSER FUNCTIONS

- alert() displays an alert box with an OK button
- Very frustrating to encounter as a user
- Unideal for accessibility, so use it sparingly

```
alert("Annoying Alert");
```



COMMON USE CASE: FORMS

HTML element input creates a form:

```
<input id="ageField">
<button type="button" onclick="TestInputValidity()">Submit</button>
```

JavaScript can read the input the .value property:

```
function TestInputValidity() {
  let x = document.getElementById("ageField").value;

  let text;
  if (isNaN(x) || x < 1 || x > 130) {
    text = "Input not valid";
  } else {
    text = "Input OK";
  }
  return text;
}
```

COMMON USE CASE: FORMS

CHANGING STYLES WITH JAVASCRIPT

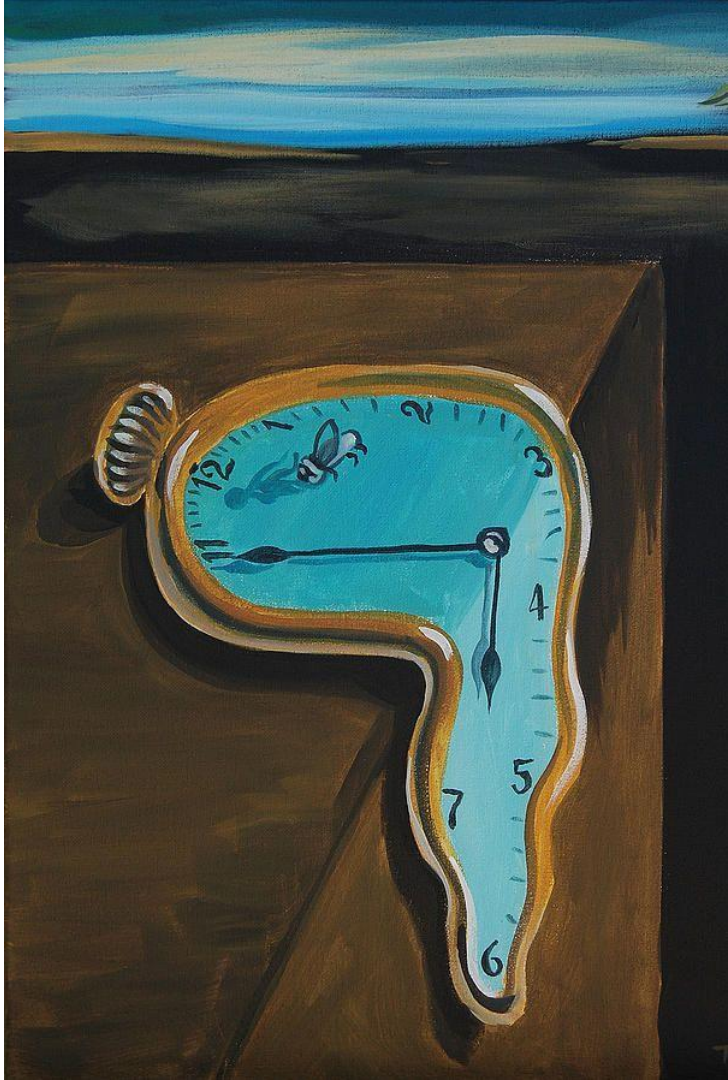
Styles can be changed with JavaScript, but it's generally bad practice.

```
document.getElementById("nav").style.color = "blue";
```

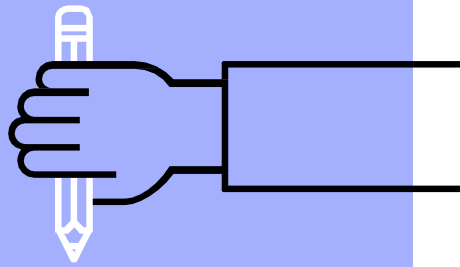
DATE

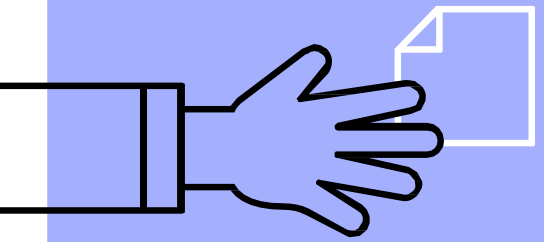
January 1, 1970 00:00:00 UTC – the 'origin' epoch for Date.

- Date() returns the current date and time
- Date.now() returns number of milliseconds since the origin epoch
- Date.parse() parses a string representation of date and returns the number of milliseconds since the origin epoch
- Date.UTC() is similar to .parse, but uses UTC (Coordinated Universal Time)



OPERATORS, LOGICAL OPERATORS, COMPARATORS





OPERATORS

Operator	Description
+	Addition
-	Subtraction
/	Division
*	Multiplication
**	Exponentiation
%	Modulus
++	Increment by
--	Decrement by

COMPARISON OPERATORS

Operator	Description
==	Equal to
===	Equal to and equal value
!=	Not equal
!==	Not equal type and equal value

COMPARISON OPERATOR EXAMPLES

Example of ==vs. ===

```
if (42 == "42") {
    //do something
}
```

```
if (42 === "42") {
    //do something
}
```

LOGICAL OPERATORS

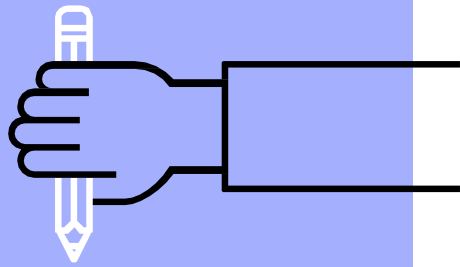
Logical Operators	Description
&&	And
||	Or
!	Not

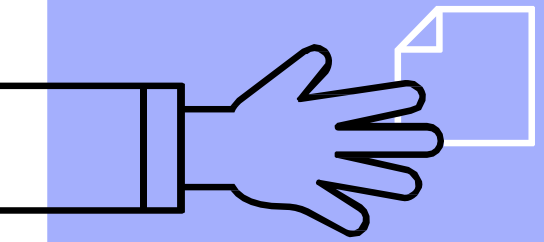
LOGICAL OPERATOR EXAMPLE

Lots of "or" conditionals – this is one approach, but looks funky:

```
if (jobTitle == "Professor" ||
    jobTitle == "Asst. Professor" ||
    jobTitle == "Assistant Professor" ||
    jobTitle == "Associate Professor"||
    jobTitle == "Assoc. Professor") {
        honorific = "Professor";
}
```

SWITCHES, NESTED FUNCTIONS, OBJECTS





SWITCHES

Switches allow us to present more human readable lists of laying out conditions.

Switches can use string values for cases as well.

```
switch (new Date().getDay())
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
     day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
}
```

SWITCHES (DEFAULT VALUES)

Often useful to set a default case that handles most expected outputs.

```
const xmas = new Date('December 25, 1995 23:15:00');
const year = xmas.getYear();
switch (year % 4) {
    case 0:
        days = 29;
    break;
    default:
        days = 28;
}
```

NESTED FUNCTIONS

JavaScript allows you to nest functions within functions, and those functions only exist within the scope of the function that called it.

Used in situations where it’s useful to encapsulate part of your algorithm, and that part of the algorithm will only ever be called by the main (rather than nested) function.

NESTED FUNCTION EXAMPLE

calculateTotal function has calculateTax function nested within it

```
function calculateTotal(price, quantity) {

    var subtotal = price*quantity;
    return subtotal + calculateTax(subtotal);

    function calculateTax(subtotal) {
        var taxRate = 0.05;
        var tax = subtotal * taxRate;
        return tax;
    }
}
```

ASSIGNING OBJECTS

Using objects is useful!

- Allows us to store related values in a single entity
- Very similar to variable assignment

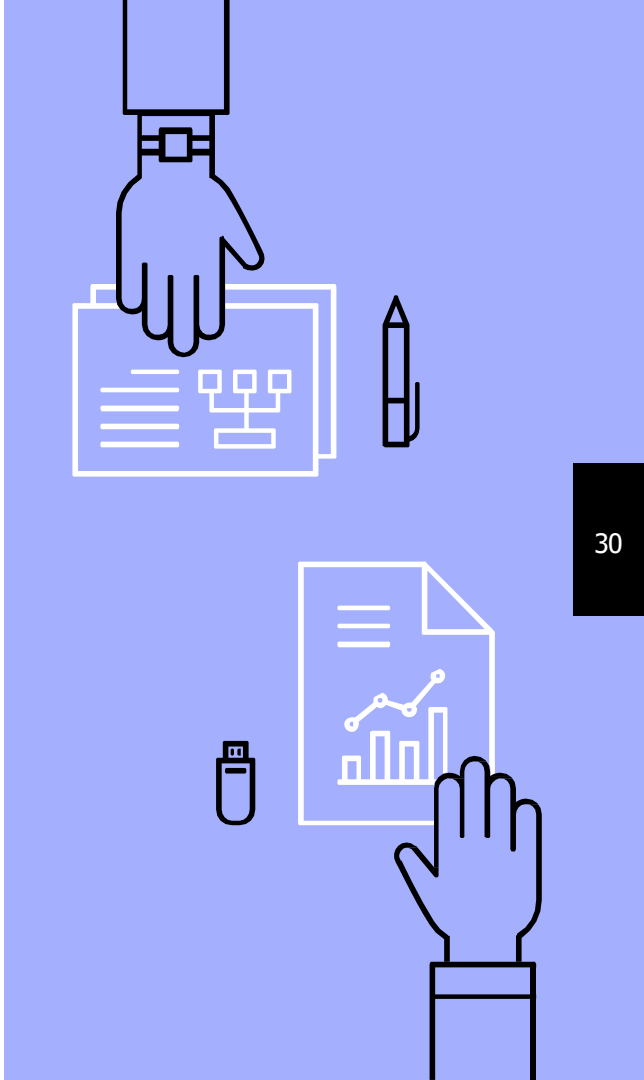
```
const student = {firstName:"John", lastName:"Doe", age:20, NSID:"jbd234"};
```

ACCESSING OBJECTS

Can be done in two ways:

objectName.propertyName
objectName[“propertyName”]

```
student.NSID;
student[“NSID”];
```



THE 'THIS' KEYWORD

'this' can be used to refer to an instance of an object – e.g.:

```
const student = {
  firstName: "Frank",
  lastName: "Walker",
  NSID: "frw245",
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```