WELCOME

CPNT-262: JAVASCRIPT

COURSE INFORMATION

BEGINNING JAVASCRIPT BY JEREMY MCPEAK, 5TH EDITION

GRADING

Activity	Points
Daily JavaScript Exercises	25%
Final Project	60%
Participation/Attendance	15%

FINAL PROJECT

- Day 9 is an all-day workshop
- Final Project is due at the end of Day 9
- You will be creating a website for a travel agency.

RESOURCES

Slides, problem sets/assignments, and the Final Project can be found on the Brightspace website.

ACTIVITY: GITHUB

- For this class, you will use Github and Github pages to host your files and submit your assignments.
- If you have not already done so, sign up for a Github account and install Github Desktop.

ACTIVITY: CREATE YOUR FIRST REPOSITORY

1. In the upper right corner, next to your avatar or identicon, click and then select New repository.
2. Name your repository sait-js.
3. Write a short description - "Repository for JS exercises."
4. Select Initialize this repository with a README.
5. Click Create repository.

ACTIVITY: GITHUB DESKTOP

1. Open Github Desktop.
2. Add your Github account under **File > Options**.
3. Go to **File > Clone Repository**.
4. Select your new sait-js repository.
5. Create a local path where you want to store your repository (C:\Desktop\Repositories\sait-js).
6. Click Clone.
7. Find your new folder on the Desktop and add 9 folders to it (day1-8 and final-project)

ACTIVITY: UPDATING YOUR REPO

1. Once you've created new files or folders in your local repository, you can upload these changes to Github.
2. In Github Desktop, see the changes you've made on the left.
3. Add a summary of the changes you've made.
4. Click Commit to master.
5. Push your changes.

You can work on these files at home by cloning it to your home computer and Pulling and Pushing changes.

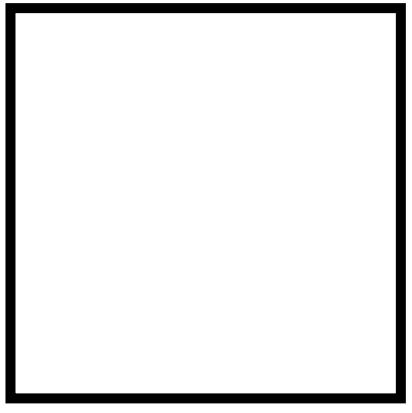
You will submit your github repository URLs with completed work in it.

WHAT IS JAVASCRIPT?

```
    /**
     * Expects an argument that is either a youtube URL or a ID,
     * and returns back the ID.
     */
    getIdFromUrl: function(videoIdOrUrl) {
        if (videoIdOrUrl.indexOf('http') === 0) {
            return videoIdOrUrl.split('v=')[1];
        } else {
            return videoIdOrUrl;
        }
    },
```

EXAMPLE: DRAW A SQUARE

1. Find a whiteboard and a dry erase marker.
2. Uncap the dry erase maker.
3. Hold the marker in your hand.
4. Place the marker against the whiteboard.
5. Move your hand 1 foot to the right.
6. Stop.
7. Move your hand 1 foot down...



INTERPRETED VS COMPILED LANGUAGES

Interpreted

- JavaScript, Ruby, PHP

Compiled

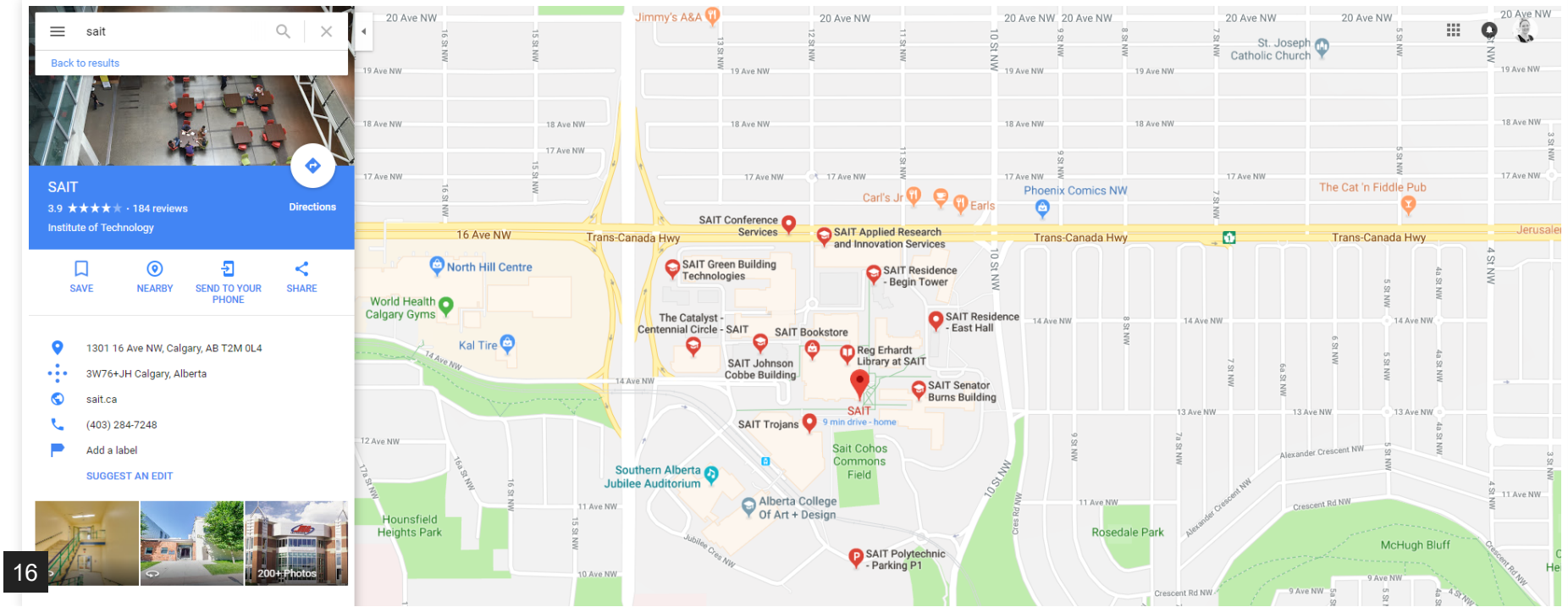
- C#, Java

THE HISTORY OF JAVASCRIPT

- 1995: At Netscape, Brendan Eich created "JavaScript".
- 1996: Microsoft releases "JScript", a version of JavaScript that was made for Internet Explorer 3.
- 1997: JavaScript was standardized in the "ECMAScript" spec.
- 2005: "AJAX" was coined and the web 2.0 age began.
- 2006: jQuery 1.0 was released.
- 2010: Node.js was released.
- 2015: ECMAScript 6 was released.

WHAT CAN YOU DO WITH JAVASCRIPT?

- validate form input
- create image carousels and lightboxes
- keep track of users with cookies
- create interactive elements like tabs, sliders, and accordions
- create full-featured apps like maps, calendars and productivity software



WHAT TOOLS DO YOU NEED?

To work with JavaScript, you need:

- a text editor
- a web browser

ADD JS TO A WEB PAGE

```
<script>
    // JavaScript goes here
</script>
```

```
<!-- Put scripts directly before the closing body tag. -->

    <script>
        // JavaScript goes here
    </script>
</body>
```

LINK TO AN EXTERNAL FILE

```
<script src="example.js"></script>
```

ACTIVITY - CREATE YOUR FIRST SCRIPT

1. Make a folder on your desktop called `sait-js`
2. Inside, make a new page called `index.html`.
3. Write this code in your new page.

```
<!DOCTYPE html>
<html>
<head>
    <title>Test Page</title>
</head>
<body>
    <p>This is my awesome JavaScript code.</p>
    <script>
    alert('Hello World!');
    console.log('Secret message');
    </script>
<script src="//0.0.0.0:35729/livereload.js?snipver=1" async="" defer=""></script>
</body>
</html>
```

ACTIVITY - EXPLORE THE CONSOLE

1. Open your practice page (`index.html`).
2. Open the console.
3. In Chrome, use the keyboard shortcut:
    - Mac: Command + Option + J
    - Windows: Control + Shift + J
4. Do you see anything in the console?
5. Try typing in 2 + 2 and hitting enter.

DATA TYPES AND VARIABLES

STATEMENTS

Each instruction in JS is a "statement".

```
console.log('Hello World!');
```

COMMENTS

You can leave comments in your code. People can read these but computers will ignore them.

```
/*
I can make long comments
with multiple lines here
*/

console.log('Hello World!'); // Or make short comments here
```

GETTING RESULTS ONTO YOUR SCREEN

Open a popup box.

```
alert('Hello World!');
```

Display a message in your console.

```
console.log('Hello World!');
```

Add something to the page.

```
document.write('Hello World!');
```

ACTIVITY - COMMENTS/EXTERNAL FILES

1. Open index.html.
2. Add a comment to the code.
3. Try different ways of printing a message.
4. Create a new file called mycode.js.
5. Move your code to this file and link it to your page.

VARIABLES

- Declare, then initialize in 2 statements:

```
var x;
x = 5;
console.log(x);
```

- Or declare and initialize in one statement:

```
var y = 2;
console.log(y);
```

- Reassign the value later:

```
var x = 5;
x = 1;
```

RULES FOR VARIABLE NAMES

- begin with letters, $ or _
- only contain letters, numbers, $ and _
- case sensitive
- avoid reserved words
- choose clarity and meaning
- prefer camelCase for multipleWords (instead of under_score)
- pick a naming convention and stick with it

EXAMPLES OF VARIABLE NAMES

OK:

```
var numPeople, $mainHeader, _num, _Num;
```

Not OK:

```
var 2coolForSchool, soHappy!
```

ACTIVITY: CREATE A VARIABLE

- In your .js file, create a variable and give it a valid name and value.
- Then, display the value in the console.

PRIMITIVE DATA TYPES

- string: an immutable string of characters

```
var greeting = 'Hello Kitty';
var restaurant = "Pamela's Place";
```

- number: whole (6, -102) or floating point (5.8737)

```
var myAge = 34;
var pi = 3.14;
```

- boolean: represents logical values - true or false

```
var catsAreBest = true;
var dogsRule = false;
```

PRIMITIVE DATA TYPES

- undefined: represents a value that hasn't been defined

```
var notDefinedYet;
```

- null: represents an empty value

```
// foo is known to exist but it has no type or value:
var foo = null;
```

NUMBERS

Variables can be numbers (integers or floating point).

```
var numberOfKittens = 5;
var cutenessRating = 9.6;
```

ARITHMETIC OPERATORS

Once you have numbers, you can do math!

```
var numberOfKittens = 5;
var numberOfPuppies = 4;
var numberOfAnimals = numberOfKittens + numberOfPuppies;
```

ARITHMETIC OPERATORS

Example	Name	Result
`-a`	Negation	Opposite of `a`
`a + b`	Addition	Sum of `a` and `b`
`a - b`	Subtraction	Difference of `a` and `b`
`a * b`	Multiplication	Product of `a` and `b`
`a / b`	Division	Quotient of `a` and `b`
`a % b`	Modulus	Remainder of `a` and `b`

ACTIVITY: ARITHMETIC

- Create 2 variables and try some arithmetic operators.
- Don't forget to display your results in the console.

STRINGS

- a string holds an ordered list of characters

```
var alphabet = "abcdefghijklmnopqrstuvwxyz";
```

- the length property reports the size of the string

```
console.log(alphabet.length); // 26
```

STRINGS

You can use double quotes or single quotes for string data.

```
var kittenName = "Fluffy";
var puppyName = 'Spike';
```

If you want to use a quote in your string, you'll need to escape it or vary your quotation marks.

```
console.log('I\'d like to use an apostrophe.');
console.log("I'd like to use an apostrophe.");
console.log('Now I want to use "double quotes".');
```

STRING OPERATORS

You can use a `+` to concatenate strings.

```
var kittenName = "Fluffy ";
var fullName = kittenName + "Fluffykins";
console.log(fullName); // Outputs "Fluffy Fluffykins"
```

STRING OPERATORS

You can also use `+=` to add things to the end of a string.

```
var kittenName = "Admiral ";
kittenName += "Snuggles";
console.log(kittenName); // outputs "Admiral Snuggles";
```

ACTIVITY: WHAT'S YOUR NAME?

- Create 2 variables
  - First Name
  - Last Name
- Put them together to make a full name.
- Display your results in the console.

EXPRESSIONS

Variables can store the result of any "expression".

```
var x = 2 + 2;
var y = x * 3;

var name = "Geneviève";
var greeting = "Hello" + name;
var title = "Duchess";
var formalGreeting = greeting + ", " + title;
```

EXPRESSIONS

Variables can store input from users using the **prompt** function.

```
var name = prompt("What's your name?");
console.log("Hello " + name);
```

LOOSE TYPING

JS figures out the type based on value, and the type can change:

```
var x;
console.log(typeof x) // undefined

x = 2;
console.log(typeof x) // number

x = "Hi";
console.log(typeof x) //string
```

A variable can only be one type.

```
var y = 2 + ' cats';
console.log(typeof y);
```

EXERCISES

1. The Fortune Teller
2. The Age Calculator
3. The Lifetime Supply Calculator
4. The Geometrizer
5. The Temperature Converter

EXERCISE: THE FORTUNE TELLER

Why pay a fortune teller when you can just program your fortune yourself?

Store the following into variables:

- number of children
- partner's name
- geographic location
- job title

Output your fortune to the console like so: "You will be a X in Y, and married to Z with N kids."

EXERCISE: THE AGE CALCULATOR

Want to find out how old you'll be? Calculate it!

- Store your birth year in a variable.
- Store a future year in a variable.
- Calculate your 2 possible ages for that year based on the stored values. (For example, if you were born in 1988, then in 2026 you'll be either 37 or 38, depending on what month it is in 2026.)
- Output them to the console like so: "I will be either NN or NN in YYYY", substituting the values.

EXERCISE: THE LIFETIME SUPPLY CALCULATOR

Ever wonder how much a "lifetime supply" of your favorite snack is? Wonder no more!

- Store your current age into a variable.
- Store a maximum age into a variable.
- Store an estimated amount per day (as a number).
- Calculate how many you would eat total for the rest of your life.
- Output the result to the screen like so: "You will need NN to last you until the ripe old age of X".

EXERCISE: THE GEOMETRIZER

Calculate properties of a circle, using these definitions.

- Store a radius into a variable.
- Calculate the circumference based on the radius, and output "The circumference is NN" into the console.
- Calculate the area based on the radius, and output "The area is NN" into the console.

EXERCISE: THE TEMPERATURE CONVERTER

It's cold out! Let's make a converter based on the steps here.

- Store a celsius temperature into a variable.
- Convert it to fahrenheit and output "NN°C is NN°F".
- Now store a fahrenheit temperature into a variable.
- Convert it to celsius and output "NN°F is NN°C."

FUNCTIONS

DECLARING FUNCTIONS

To declare (create) a function, you can give it a name, then include all the code for the function inside curly brackets { }

```
function parrotFacts() {
    console.log('Some parrot species can live for over 80 years.');
    console.log('Kakapos are a critically endangered flightless parrot.');
}
```

USING FUNCTIONS

To invoke (use) a function, type its name, followed by a parenthesis `( )`

```
parrotFacts();
```

ACTIVITY: WRITE A FUNCTION

- Write a function that outputs a sentence into your console.
- Call that function in your code.

ARGUMENTS

Functions can accept input values, called `arguments`.

```
function callKitten(kittenName){
    console.log('Come here, ' + kittenName + '!');
}

callKitten('Fluffy'); // outputs 'Come here, Fluffy!'

function addNumbers(a, b){
    console.log(a + b);
}

addNumbers(5, 7);  // outputs 12
addNumbers(9, 12); // outputs 21
```

ARGUMENTS

You can pass `variables` into functions.

```
function addOne(num){
    var newNumber = num + 1;
    console.log('You now have ' + newNumber);
}

// Declare variables
var numberOfKittens = 5;
var numberOfPuppies = 4;

// Use them in functions
addOne(numberOfKittens);
addOne(numberOfPuppies);
```

ACTIVITY: WRITE A SIMPLE PROGRAM

- Write a simple program to combine a first name and a last name inside a function.
- Update the function to accept a first and last name as arguments.

RETURNING VALUES

You can have a function give you back a value to use later.

```
function square(num) {
    return num * num;
}

console.log(square(4));        // outputs '16'

var squareOfFive = square(5); // squareOfFive equals '25'
```

`return` will immediately end a function.

ACTIVITY: RETURN STATEMENTS

- Add a return statement to your "name" function.
- Use that function to set the value of a variable.

VARIABLE SCOPE

GLOBAL SCOPE

- A variable declared outside of a function has a `global scope`.
- It can be accessed anywhere, even inside of function.

```
var awesomeGroup = 'Girl Develop It'; // Global scope

function whatIsAwesome() {
    console.log(awesomeGroup + ' is pretty awesome.'); // Will work
}

whatIsAwesome();
```

LOCAL SCOPE

- A variable declared inside a function has `local scope`.
- It can only be accessed within that function.

```
function whatIsAwesome() {
        var awesomeGroup = 'This class'; // Local scope
        console.log(awesomeGroup + ' is pretty awesome.'); // Will work
}

whatIsAwesome();

console.log(awesomeGroup + ' is pretty awesome.'); // Won't work
```

BOOLEAN VARIABLES

BOOLEAN VARIABLES

- Represent logical values: `true` and `false`.

```
var catsAreBest = true;
var dogsRule = false;
```

BOOLEAN VARIABLES

- Some values are considered `falsy`.
- They evaluate to `false` in a Boolean context.

```
// the following variables will evaluate as false

var myName = '';
var numOfKids = 0;
var isMarried;     // remember a variable with no value is undefined
```

- `null` and `NaN` will also evaluate as `false`.
- Everything else evaluates as `true`.

CONTROL FLOW

THE IF STATEMENT

Use `if` statements to decide which lines of code to execute, based on a condition.

```
if (condition) {
        // statements to execute
}
var age = 30;

if (age > 18) {
        console.log('You are an adult');
}
```

COMPARISON OPERATORS

Symbol	Purpose
`==`	Tests if LHS is Equal to RHS
`===`	Tests if LHS is Identical to RHS
`!=`	Tests if LHS is Not equal to RHS
`!==`	Tests if LHS is Not identical to RHS
`<`	Tests if LHS is Less than to RHS
`>`	Tests if LHS is Greater than to RHS
`<=`	Tests if LHS is Less than or equal to to RHS
`>=`	Tests if LHS is Greater than or equal to to RHS

WARNING: `==` VS `===`

2 equal signs means testing for same values.

```
77 == '77'
// true
```

3 equal signs means testing for same value AND same type.

```
77 === '77'
// false (Number v. String)
```

Generally default to `===`, especially as you're learning.

ACTIVITY: TEMPERATURE

- Make a variable called "temperature".
- Write some code that tells you to put on a coat if it is below 10 degrees.

THE IF/ELSE STATEMENT

Use `else` to provide an alternate set of instructions.

```
var age = 30;

if (age >= 16) {
        console.log('Yay, you can drive!');
} else {
        console.log('Sorry, you have ' + (16 - age) +
' years until you can drive.');
}
```

THE IF/ELSE STATEMENT

If you have multiple conditions, you can use `else if`.

```
var age = 30;

if (age >= 65) {
        console.log('I am at least 65 years old.');
} else if (age >= 30) {
        console.log('I am between 30 and 64 years old.');
} else if (age >= 18) {
        console.log('I am between 18 and 29 years old.');
} else {
        console.log('I am younger than 18 years old.');
}
```

ACTIVITY: MODIFY TEMPERATURE CODE

Modify your "wear a coat" code for these conditions:

- If it's less than 10 degrees, wear a coat.
- If it's less than 0 degrees, wear a coat and a hat.
- If it's less than -20 degrees, stay inside.
- Otherwise, wear whatever you want.

LOGICAL OPERATORS

`&&` - **AND** (True if both LHS AND RHS are true)

`||` - **OR** (True if LHS OR RHS is True)

`!a` - **NOT** (True if `a` is NOT true)

USING LOGICAL OPERATORS

```
var likesDogs = true;
var likesCats = true;

if (likesDogs && likesCats) {
        console.log('I like dogs and cats.');
} else if (likesDogs || likesCats) {
        console.log('I like cats or dogs, but not both.');
} else {
        console.log("I don't like dogs or cats.");
}
```

ACTIVITY: LOGICAL OPERATORS

- Add a logical operator to your "what to wear" program.