Introduction to JavaScript

Frameworks and Architectures of the Web

Spring 2018

Today’s Program

Introduction to JavaScript

Syntax and Operators

Control Structures

Arrays, Objects, Booleans

A Pop Quiz

Break

Exercises

● **The Finishing**

Stavroz

The Ginning

Course Outline

Foundation

Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Introduction, HTML Syntax, Structure and Semantics	Interface Principles, Design Patterns and Aesthetics	HTML and CSS Preliminaries	CSS Layout & Positioning	Mobile and Responsive Design, Forms and Data Validation	Introduction to JavaScript	Document Object Model and Events in JavaScript	CSS3, Graphics and Media and Advanced JavaScript

Web Project - Design, Wireframes and Interactive Prototype

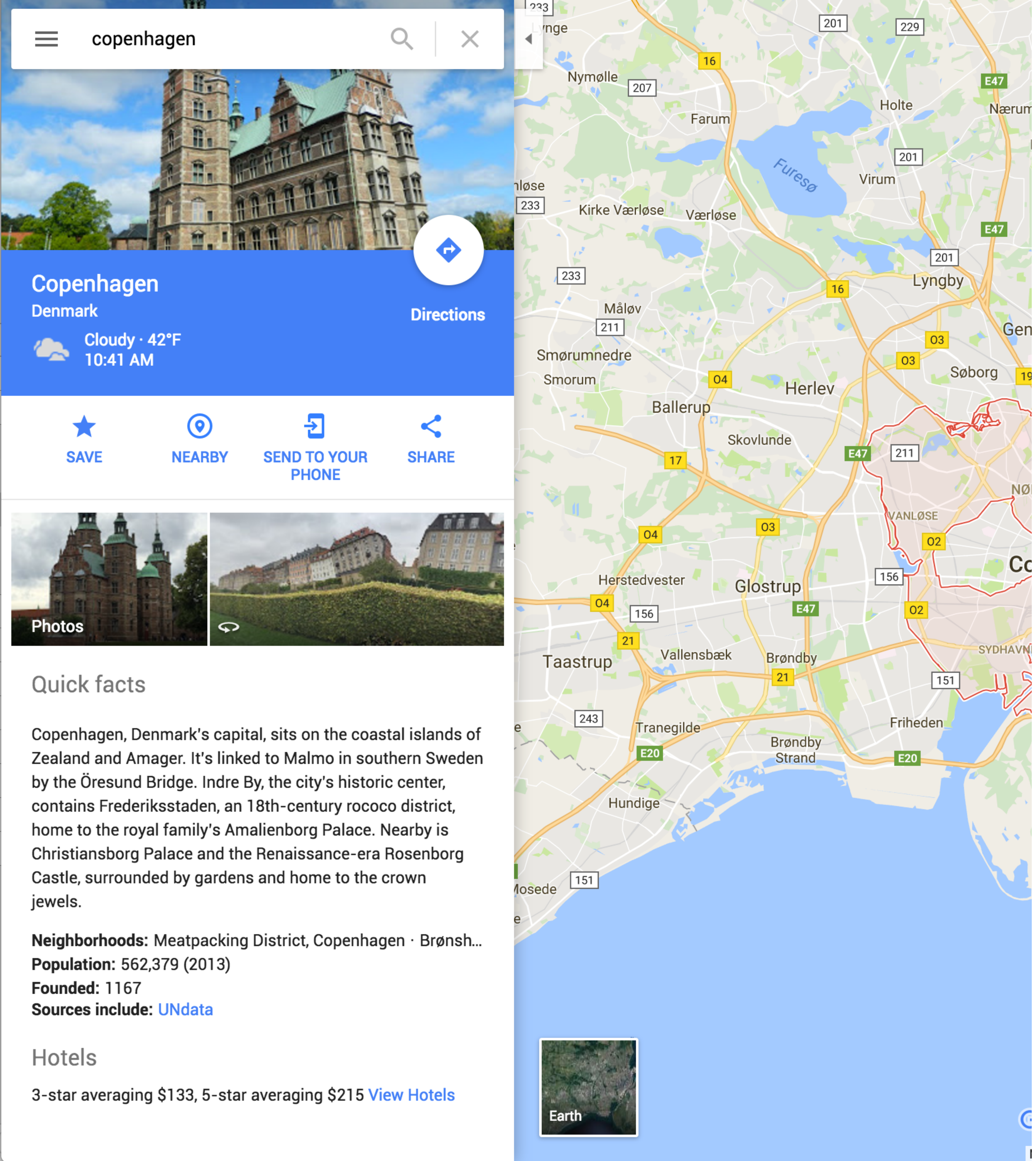
Web Project - Impleme

JavaScript

JavaScript is a programming language that provides interactivity to the Web. You can use it add **events** to your page so that for example, clicking on an element can cause it run a piece of code.

That piece of code can be used to perform some logic and/or **manipulate elements** on a page.

Elements can be manipulated in a number of ways - for example, you can programmatically change an element’s colour or position, or show and hide an element based on certain actions or feedback.


copenhagen

Copenhagen
Denmark
Cloudy · 42°F
10:41 AM

Quick facts

Copenhagen, Denmark's capital, sits on the coastal islands of Zealand and Amager. It's linked to Malmo in southern Sweden by the Öresund Bridge. Indre By, the city's historic center, contains Frederiksstaden, an 18th-century rococo district, home to the royal family's Amalienborg Palace. Nearby is Christiansborg Palace and the Renaissance-era Rosenborg Castle, surrounded by gardens and home to the crown jewels.

Neighborhoods: Meatpacking District, Copenhagen · Brønsh...
Population: 562,379 (2013)
Founded: 1167
Sources include: UNdata

Hotels

3-star averaging $133, 5-star averaging $215 View Hotels

Web Applications

JavaScript allows Web pages to feel more like desktop or mobile applications. Web sites that employ heavy use of JavaScript to provide app-like experiences are called Web Applications. G-mail, Google Docs and Google Maps are all popular examples of Web Applications.

```
<html>
  <head>
    <title>Introduction to JavaScript</title>
  </head>
  <body>
      <h1>My Web Page</h1>
      <p id="demo">A Paragraph</p>
      <input type="button" value="Click Me"
onclick="myFunction()" />
      <script type="text/javascript">

       function myFunction() {
          document.getElementById("demo").innerHTML =
"My First JavaScript Function";
       }

    </script>
  </body>
</html>
```

A Simple JavaScript App

JavaScript can be added into a page by using a `<script>` tag. Any JavaScript code that is placed inside the `<script>` tag runs as the page is loaded.

In the example on the left, the JavaScript code in the `<script>` tag defines a function called `myFunction()`.

The function, once executed, manipulates the contents of the paragraph element with ID `demo` to display “My First JavaScript Function.”

Like your stylesheets, you can place your JavaScript into external files.

The files are referenced and linked within the <head> section of your HTML document. For larger projects, this is considered to be best practice.

index.html

```
<html>
  <head>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
     <h1>My Web Page</h1>
     <p id="demo">A Paragraph</p>
     <input type="button" value="Click Me" onclick="myFunction()" />
  </body>
</html>
```

script.js

```
function myFunction() {
  document.getElementById("demo").innerHTML = "My First JavaScript
Function";
}
```

```
var message = "Hello World!";
alert(message);
```

JavaScript Syntax

JavaScript statements are case sensitive, and should be terminated by a semicolon.

JavaScript Syntax

```
var age = 21;
if (age > 18) {
   alert("Buy me a beer!");
}

function myFunction() {
  var message = "Hello World!";
  alert(message);
}
```

JavaScript blocks are common in functions and control statements. They begin with an opening brace { and end with a closing brace }.

The example on the left shows the code from the precious slide inside a function, which is a piece of code that can be executed later on.

```
/*
The alert will be displayed only if the variable
age is > 18
*/
var age = 21;
if (age > 18) {
   alert("Buy me a beer!");
}


/*
The following function, when called, will display the
message "Hello World!" to the user.
*/
function myFunction() {
  // Assigns the string "Hello World!" to a variable.
  var message = "Hello World!";
  // Displays that variable as a message.
  alert(message);
}
```

Comments

You can add comments to your JavaScript to make it easier for others to read your code.

Comments can be written on a single line using //. You can also create multiline comments using /* and */.

```
var carname = "Volvo"
var yearModel = 1995;
```

Variables

Variables are declared with the keyword `var`. Variables are used to hold bits of information, such a strings, numbers and dates.

Function Scope

Variables declared inside of functions can only be accessed within that function.

```
var carname = "Volvo"
var yearModel = 1995;

function outputCarInfo() {
  var countryOfOrigin = 'Sweden';
  console.log(carname);
  console.log(yearModel);
  console.log(countryOfOrigin);
}

outputCarInfo();

/*
Outputs:
Volvo
1995
Sweden
*/
```

```
var carname = "Volvo"
var yearModel = 1995;

function outputCarInfo() {
  var countryOfOrigin = 'Sweden';
  console.log(carname);
  console.log(yearModel);
  console.log(countryOfOrigin);
}

outputCarInfo();

/*
Outputs:
Volvo
1995
Sweden
*/

console.log(carname);
console.log(yearModel);
console.log(countryOfOrigin);

/*
Outputs:
Volvo
1995
Uncaught ReferenceError: countryOfOrigin is not
defined
*/
```

Function Scope

In the example on the left, the variables `carname` and `yearModel` are declared outside of a function, so they can be accessed everywhere.

However, `countryofOrigin` is declared only inside the `outputCarInfo()` function, so it can only be accessed within that function.

Attempting to access `countryofOrigin` outside of that function results in an error.

https://codepen.io/timwray87/pen/KQjBZo
Modify the code so that the car name, year model and contry of origin are displayed.

Numbers and Strings

```
var y = 5;
var z = 2;
var x = y + z;

console.log(x); // Outputs: 7
```

You can add (+), subtract (-), multiply (*) and divide (/) numbers.

You can also use parentheses to group these arithmetic operations.

If the values of two variables are numbers, they are treated as such in arithmetic operations.

Numbers and Strings

If the values of two variables are strings, they can be joined together using the + operator.

```
var txt1 = "What a very";
var txt2 = "nice day";
var txt3 = txt1 + " " + txt2;
console.log(txt3); // Outputs: "What a very nice day"
```

Numbers and Strings

Adding two numbers will return the sum, but adding a number and a string will return a string.

```
var x = 5 + 5;
var y = "5" + 5;
var z = "Hello" + 5;

console.log(x);
console.log(y);
console.log(z);

/*
Outputs:
10
55
Hello5
*/
```

```
var age, message;
age = 25;
if (age >= 18) {
  message = "I would like a beer";
} else {
  message = "I would like a juice";
}

console.log(message);
// Outputs: I would like a beer
```

`if` statements

`if` statements used to perform actions based on different conditions.

switch statements

switch statements are used to perform actions based on different conditions.

```
var englishWord = "steak";
var translation;
switch (englishWord) {
  case "potato" :
    translation = "kartoffel";
    break;
  case "beef" :
  case "steak" :
    translation = "bøf";
    break;
  default :
    alert("There is no translation available!");
    break;
}

console.log(translation);
// Outputs : bøf
```

```
<button onclick="myFunction('Bob','Builder')">
  Try it
</button>
<button onclick="myFunction('Harry Potter','Wizard')">
  Try it
</button>

<script type="text/javascript">

function myFunction(name, job) {
  alert("Welcome " + name + ", the " + job);
}

</script>
```

Functions

Functions are blocks of code that can be executed multiple times.

You can pass information into a function in the form of **parameters.**

A function's **arguments** are the values that are accessed from its parameters.

for loops

for loops use a counter to execute blocks of code multiple times.

```
var x = "", i;
for (i = 0; i < 5; i++) {
  x = "The number is " + i;
  console.log(x);
}

/*
Outputs :
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
*/
```

while loops

While loops execute a block of code as long as the condition is true.

```
var x = "", i = 0;
while (i < 5) {
  x = "The number is " + i;
  console.log(x);
  i++;
}

/*
Outputs :
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
*/
```

Arrays

Arrays can store multiple values in one variable. You can access an element of the array using the [n] operator, where n represents the index of that element.

```
// METHOD 1 Declare as array, assign values to indices
var myCars = new Array();
myCars[0] = "Saab";
myCars[1] = "Volvo";
myCars[2] = "BMW";

// METHOD 2 Use the Array constructor
var myCars = new Array("Saab","Volvo","BMW");

// METHOD 3 Declare as an array literal
var myCars = ["Saab","Volvo","BMW"];

console.log(myCars[0]);

/*
Outputs :
Saab
*/
```

Object Literals

Objects can be used to define your own custom data structures.

Objects consist of a series of **keys**, which can be strings, each with a **value**, which can be strings, numbers, objects or even functions.

```
// METHOD 1 Create object, assign properties
var personObj = new Object();
personObj.firstName = "John";
personObj.lastName = "Doe";
personObj.age = 25;

// METHOD 2 Create object literal, assign properties
var personObj = {};
personObj["firstName"] = "John";
personObj["lastName"] = "Doe";
personObj["age"] = 25;

// METHOD 3 Object literal with properties and values.
var personObj = {
  "firstName": "John",
  "lastName": "Doe",
  "age": 25
};

console.log(personObj["firstName"]); // Outputs: John
console.log(personObj.lastName);     // Outputs: Doe
```

CodePen Exercise

https://codepen.io/timwray87/pen/ddBjjY

Part 1- Within this CodePen, modify the code so that you create two objects, one representing the first name, last name and age of yourself, and another that represents the first name, last name and age of your classmate.

Part 2 - Create a function called `displayEldestPerson` that accepts a single array of these objects as its argument. The function displays the maximum age of the person within that array. Test the function using the objects created in Part 1.

Dates

The JavaScript `Date` object allows you to easily work with date and time.

You can create, store, output and modify dates using the `Date()` constructor and a number of helper methods, such as `getDate()`, `getMonth()` and `getYear()`.

A full list of these helper methods is available at https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

```
// Create a date based on today's date
var today = new Date();

// Create a custom date
var birthDate = new Date("27 July 1990");


console.log(today);
// Outputs: Tue Mar 14 2017 12:12:19 GMT+0100 (CET)

console.log(birthDate);
// Outputs: Fri Jul 27 1990 00:00:00 GMT+0200 (CEST)

console.log(birthDate.getDate());
// Outputs: 27

console.log(birthDate.getMonth());
// Outputs: 6

console.log(birthDate.getFullYear());
// Outputs: 1990
```

Booleans

Like many other languages, JavaScript also has a boolean data type that represents `true` and `false` values.

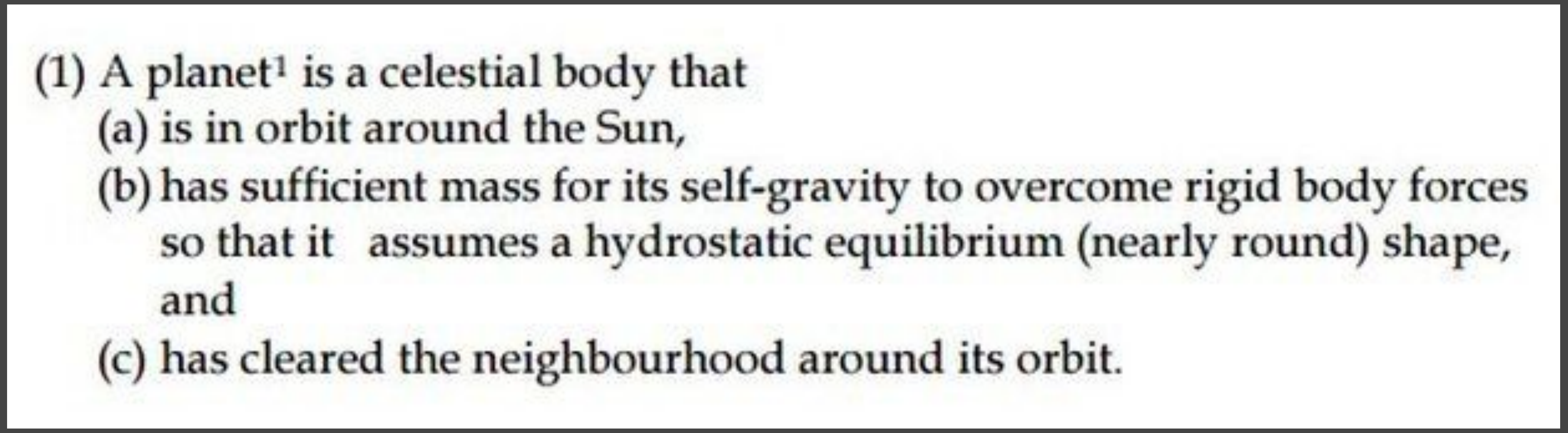
You can use the boolean operators `||` and && to create logical expressions within your `if` statements.

`||` means "or"

&& means "and"

```
var earth = {
  orbitsTheSun: true,
  orbitsAPlanet: false,
  isRound: true
}

var moon = {
  orbitsTheSun: true,
  orbitsAPlanet: true,
  isRound: true
}

var vesta = {
  orbitsTheSun: true,
  orbitsAPlanet: false,
  isRound: false
}

function describeCelestialObject(obj) {
  console.log('---')
  if (obj.orbitsTheSun || obj.orbitsAPlanet) {
    console.log("This object orbits another celestial object.");
  }
  if (obj.orbitsAPlanet) {
    console.log("This object is a moon");
  }
  if (obj.orbitsTheSun && obj.isRound && !obj.orbitsAPlanet) {
    console.log("This object is a planet");
  }
  if (obj.orbitsTheSun && !obj.isRound && !obj.orbitsAPlanet) {
    console.log("This object is an asteroid or a minor
planet.");
  }
}
```

```
console.clear();

describeCelestialObject(earth);
/*
Ouputs:
---
This object orbits another celestial object.
This object is a planet
*/

describeCelestialObject(moon);
/*
Ouputs:
---
This object orbits another celestial object.
This object is a moon
*/

describeCelestialObject(vesta);
/*
Ouputs:
---
This object orbits another celestial object.
This object is an asteroid or a minor planet
*/
```


(1) A planet[1] is a celestial body that
(a) is in orbit around the Sun,
(b) has sufficient mass for its self-gravity to overcome rigid body forces so that it assumes a hydrostatic equilibrium (nearly round) shape, and
(c) has cleared the neighbourhood around its orbit.

CodePen Exercise

https://codepen.io/timwray87/pen/qxzyzO

Create an additional object called `pluto` with the appopriate properties. Create an additional property called `hasClearedItsOwnOrbit`.

Modify the `describeCelestialObject` function so that an object can only be considered a planet if it's cleared its own orbit. The code should be such that pluto is displayed as a 'minor planet'.

Display information about this object by calling the `describeCelestialObject` function, passing in `pluto` as its argument.

FALSY

`false`

`0`

`""`

`null`

`undefined`

`NaN`

TRUTHY

`true`

`"0"`

`"false"`

`[]`

`{}`

everything **not** listed as “falsy”