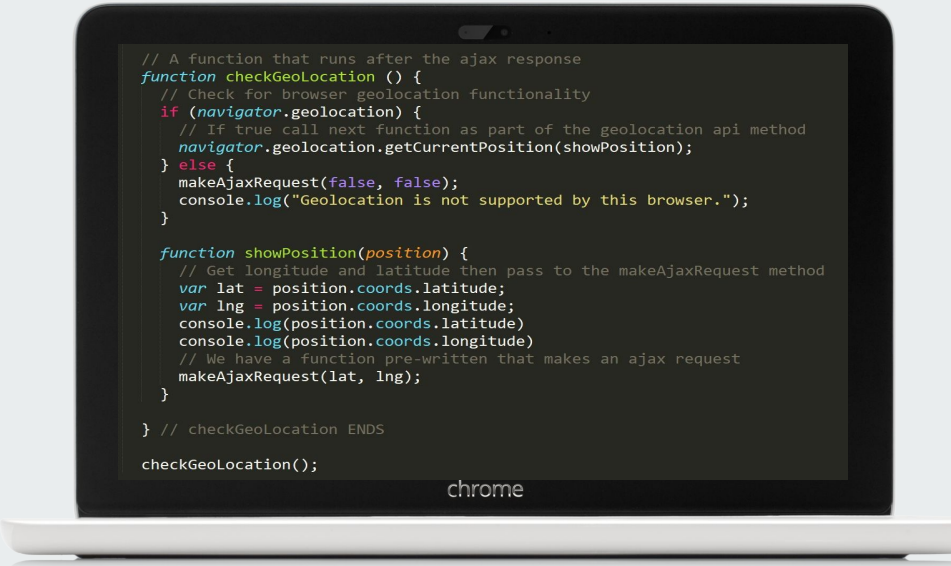
Code Cleanliness

Some tips for maintaining clean code and Examples of when and how to re-factor code.



Use Comments

Comments ***are a map*** for yourself and others. Comments help you to keep track of what your code is for and how it works.

```
//•A function that runs after the ajax response
function checkGeoLocation () {
  //•Check for browser geolocation functionality
  if (navigator.geolocation) {
    //•If true call next function as part of the geolocation api method
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    makeAjaxRequest(false, false);
    console.log("Geolocation is not supported by this browser.");
  }

  function showPosition(position) {
    //•Get longitude and latitude then pass to the makeAjaxRequest method
    var lat = position.coords.latitude;
    var lng = position.coords.longitude;
    console.log(position.coords.latitude)
    console.log(position.coords.longitude)
    //•We have a function pre-written that makes an ajax request
    makeAjaxRequest(lat, lng);
  }

} //•checkGeoLocation ENDS

checkGeoLocation();
```

The comments in the code above follow the same pattern,
Are indented and are descriptive.

Use indentation

Indentation is a must for clean and readable code. Items within functions or other blocks should be indented.

```
// A function that runs after the ajax response
function checkGeoLocation () {
  // Check for browser geolocation functionality
  if (navigator.geolocation) {
    // If true call next function as part of the geolocation api method
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    makeAjaxRequest(false, false);
    console.log("Geolocation is not supported by this browser.");
  }

  function showPosition(position) {
    // Get longitude and latitude then pass to the makeAjaxRequest method
    var lat = position.coords.latitude;
    var lng = position.coords.longitude;
    console.log(position.coords.latitude)
    console.log(position.coords.longitude)
    // We have a function pre-written that makes an ajax request
    makeAjaxRequest(lat, lng);
  }

} // checkGeoLocation ENDS

checkGeoLocation();
```

Indentation will make you code more readable and easier to scan.

Use descriptive names

Use logical and descriptive names so your code immediately tells viewers what it is for and what it does.

```
1 // A function that runs after the ajax response
function checkGeoLocation () {
  // Check for browser geolocation functionality
  if (navigator.geolocation) {
    // If true call next function as part of the geolocation api method
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    makeAjaxRequest(false, false);
    console.log("Geolocation is not supported by this browser.");
  }

2 function sp(pos) {
    // Get longitude and latitude then pass to the makeAjaxRequest method
    var lt = pos.coords.lat;
    var lg = pos.coords.lon;
    // We have a function pre-written that makes an ajax request
    ajax(lt, lg);
  }

} // checkGeoLocation ENDS

checkGeoLocation();
```

1. *Has good descriptive names which allows us to read what the code is doing*
2. *Is abstract and 'code looking' not ideal for human or tired eyes.*

Follow a pattern or general structure

If you have an idea of an overall pattern or structure for your code, it will make life easier - less bugs, more readable

```javascript
(function(){

  // Using an object in a higher scope to pass and access variables from within other functions
  var cloudObject = {};

  function init () {
    // Get Elements
    var getHeaderOne = document.querySelector('#theHeader');
    var getButton = document.querySelector('button');

    // ** Assigning the variables to the cloud object
    cloudObject.header = getHeaderOne;
    cloudObject.button = getButton

    // Set up event listeners
    getButton.addEventListener('click', buttonClick, false);

    console.log(getButton);
  }

  function buttonClick () {
    // Change the color
    $('h1').css('color', 'red');
    changeHeaderContent();
  }

  function changeHeaderContent () {
    // ** Accessing and using the cloud object
    cloudObject.header.textContent = 'Something New';
  }

  init();

}());
```

The code above uses function declarations with a cloud object to make variables easily accessible. This is a great pattern for beginners who are still getting comfortable with javascript.

Follow a pattern or general structure B

If you have an idea of an overall pattern or structure for your code, it will make life easier - less bugs, more readable

```javascript
// The object programming pattern
var app = {
  // This is a property
  cloudObject: {},
  // This is a method
  init: function () {
    console.log('are you alive?');
    // Get Elements
    var getHeaderOne = document.querySelector('#theHeader');
    var getButton = document.querySelector('button');

    console.log(app.cloudObject);
    app.cloudObject.header = getHeaderOne;
    app.cloudObject.button = getButton;
    // Set up event listeners
    getButton.addEventListener('click', app.buttonClick, false);
    console.log(getButton);
  },
  // This is a method
  buttonClick: function () {
    // Change the color
    $('h1').css('color', 'red');
    app.changeHeaderContent();
  },
  // This is a method
  changeHeaderContent: function  () {
    app.cloudObject.header.textContent = 'Something New';
  }
};

// Calling the init function from the object
app.init();
```

The code above uses a standard object with methods/functions also with a cloud object to make variables easily accessible. This is a great pattern for beginners who are still getting comfortable with javascript.

Reasons to refactor code

Sometimes after a long time working and expanding a program, your code will become messy, convoluted and scattered.

Reasons to refactor code

Sometimes after a long time working and expanding a program, your code will become messy, convoluted and scattered.

- Code has minimal to no comments
- Bugs are Starship Troopers level
- Not sure exactly how things are working
- Code lacking indentation
- Code is hard to read
- A large amount of repetitive code
- Functions are very long
- God Line - An excessively long line of code
- Code looks messy and is generally hard to read

When developers find a smelly code, the next step they do is refactoring. Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the software yet improves its internal structure. It may be the single most important technical factor in achieving agility.

If we have code smell or any of the issues on the right.....

It is time to **1. Do a code review**, and **2. Do a refactor**

Code has minimal to no comments

Bugs are Starship Troopers level

Not sure exactly how things are working

Code lacking indentation

Code is hard to read

A large amount of repetitive code

Functions are very long

God Line - An excessively long line of code

Code looks messy and is generally hard to read

If you have any of the above it is time to review and refactor your code.