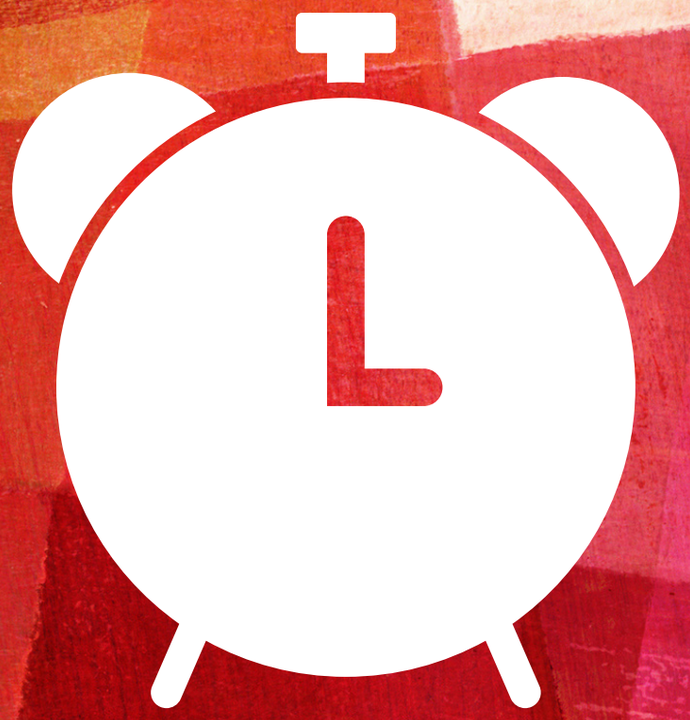
MODULE 3
QUIZZES & EXPERIMENTS

Web Front End Certification Course - SkewCode



Quiz 1

Variables and Primitive Data Types

JS FUNDAMENTALS – VARIABLES AND PRIMITIVE DATA TYPES

- What are the 3 primitive types in JS?
- What gets printed to the console in each case?

a)
```
var x;
console.log( 'x = ', x );
```

b)
```
var x = 1.5;
console.log( typeof x );
```

c)
```
var x = 1.5;
console.log( parseInt( x ) );
```

d)
```
var x = "Hello";
console.log( parseInt( x ) );
```

e)
```
var x = "1.5abc";
console.log( parseInt( x ),
parseFloat( x ) );
```

f)
```
var greeting = 'Hello,
O\'Donnell';
console.log( greeting );
```

g)
```
var greeting = 'Hello, O\'Donnell';
console.log( greeting.length );
```

h)
```
var greeting = null;
console.log( typeof greeting );
```

Experiment - 1

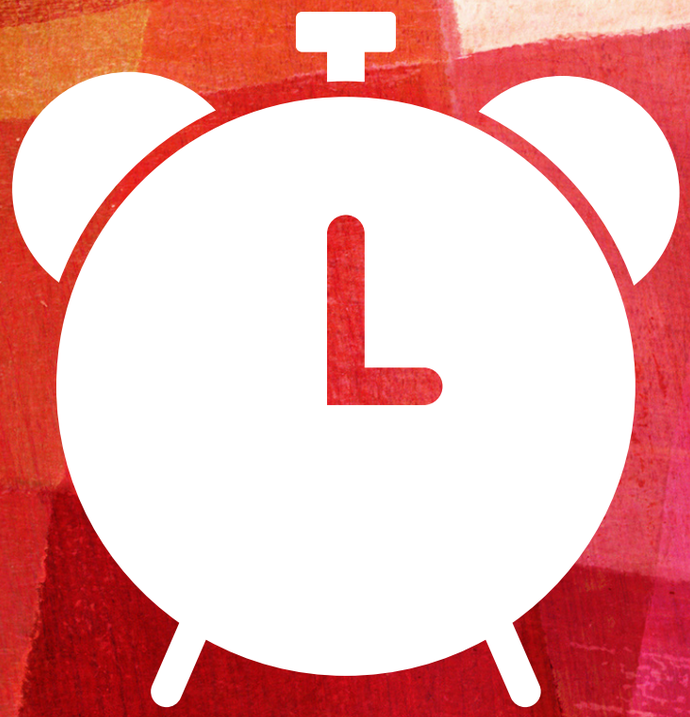
Variables and Primitive Data Types

JS FUNDAMENTALS – VARIABLES AND PRIMITIVE DATA TYPES

- Declare one each of number, string, boolean, null, undefined and one without any explicit initial value. Log each onto the console and check the results

- Check the results of these statements
  a) console.log(1 + 2 + 'hello');
  console.log('hello' + 1 + 2);
  b) console.log(x); // Note: do not define any variable x
  c) console.log('before x is defined and initialized : ', x);
  var x = 20;
  console.log('after x is defined and initialized : ', x);

- What do you infer from the results of b) and c) in the previous question? Read up on 'hoisting' in JS.
  https://developer.mozilla.org/en-US/docs/Glossary/Hoisting



Quiz 2

Variable Scopes, Scope chain

JS FUNDAMENTALS – VARIABLE SCOPES, SCOPE CHAIN

- What happens in each of these cases?

a)
```
var x = 1;
function foo() {
    console.log( 'inside foo : ', x );
}
foo();
console.log( 'outside foo : ', x );
```

b)
```
var x = 1;
function foo() {
    var x = 2;
    console.log( 'inside foo : ', x );
}
foo();
console.log( 'outside foo : ', x );
```

c)
```
function foo() {
    x = 1;
    console.log( 'inside foo : ', x );
}
foo();
console.log( 'outside foo : ', x );
```

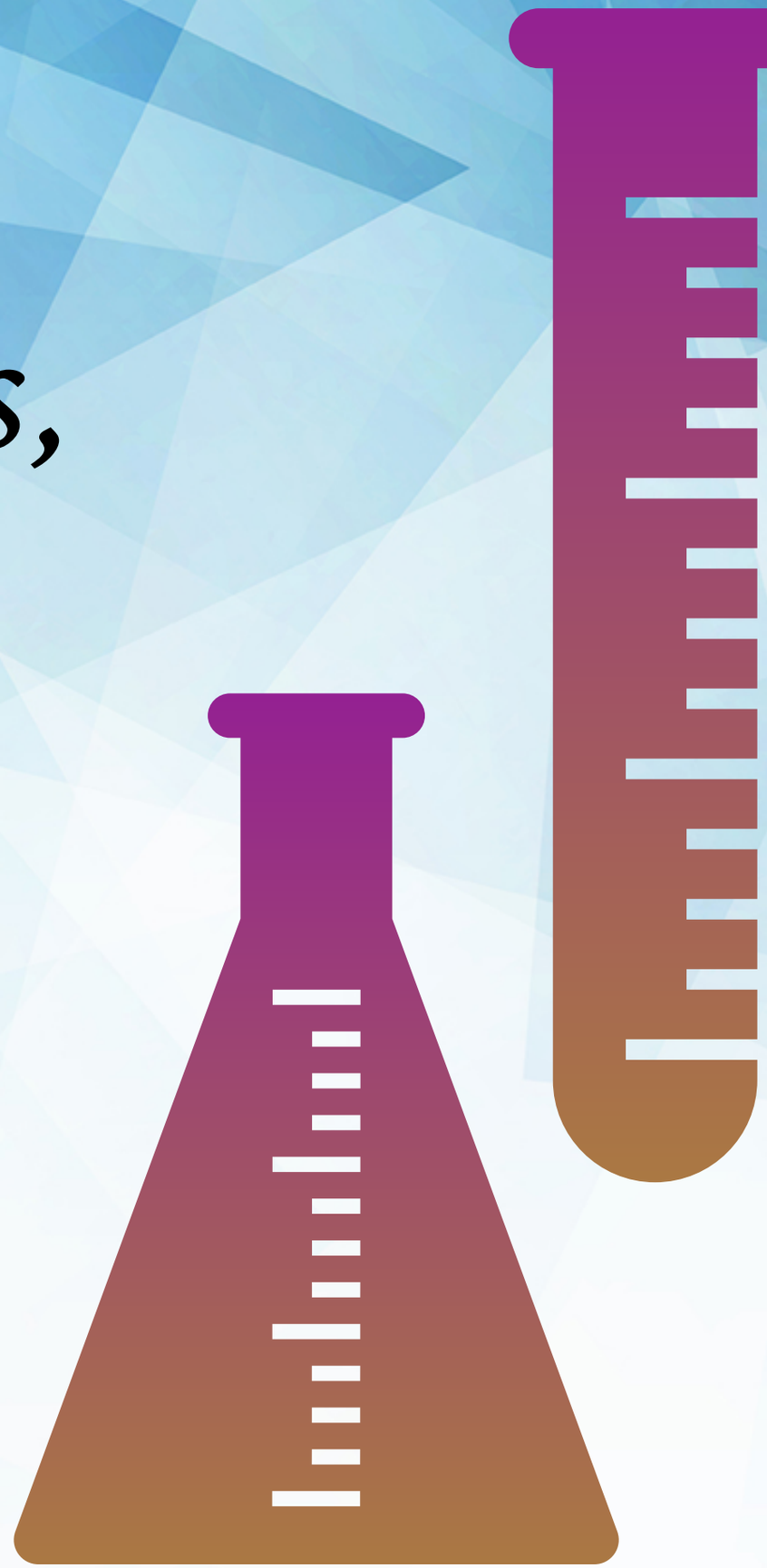
d)
```
function foo() {
    x = 1;
    console.log( 'inside foo : ', x );
}
console.log( 'outside foo : ', x );
foo();
```

e)
```
function foo() {
    var x = 1;
    console.log( 'inside foo : ', x );
}
foo();
console.log( 'outside foo : ', x );
```

f)
```
function foo() {
    var x = 1;
    for( var x = 0; x < 10; x++ ) {
        console.log( 'inside foo for loop : ', x );
    }
    console.log( 'inside foo : ', x );
}
foo();
```

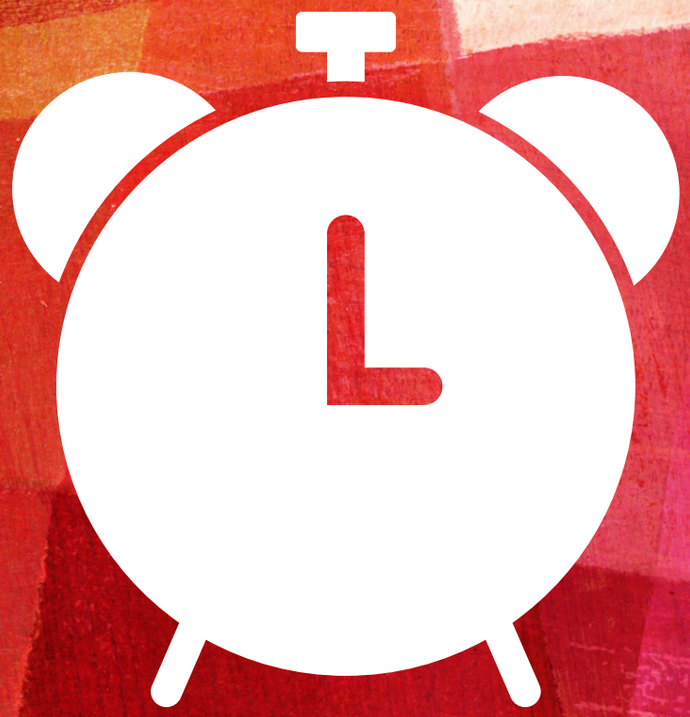
Experiment - 2

Variable Scopes, Scope Chain



JS FUNDAMENTALS – VARIABLE SCOPES, SCOPE CHAIN

- In the same file, execute the following steps and check the results. Make sure they are as per your predictions.
  - Declare variables x, y, z in global scope and set the values 1, 2, 3 respectively. Log x, y, z.
  - Declare a function foo with a new variables y and z set to 4, 5 respectively. Log x, y, z within foo.
  - Declare a function bar within foo and set a new variable z within it to 6. Log x, y, z within bar.
  - Call bar( ) within foo.
  - Call foo( ) globally.



Quiz 3

Using Arrays

JS FUNDAMENTALS – USING ARRAYS

- Can an array have items of different data types in JS?
- What is the output in each case?

a)
```
var array = [1, 2, , , 3];
console.log( array, array.length );
```

b)
```
var matrix = [
    [ 1, 2, 3 ],
    [ ],
    [ 4, 5, 6, 7, , ],
    ,
    [ 8, 9, 10, 11, 12 ],
    [ 13, 14, 15, 16, 17, 18 ]
];
console.log( matrix.length );
console.log( matrix[2] );
console.log( matrix[3] );
console.log( matrix[1].length );
console.log( matrix[2].length );
console.log( matrix[2][3] );
console.log( matrix[2][4] );
console.log( matrix[3].length );
```

Experiment - 3

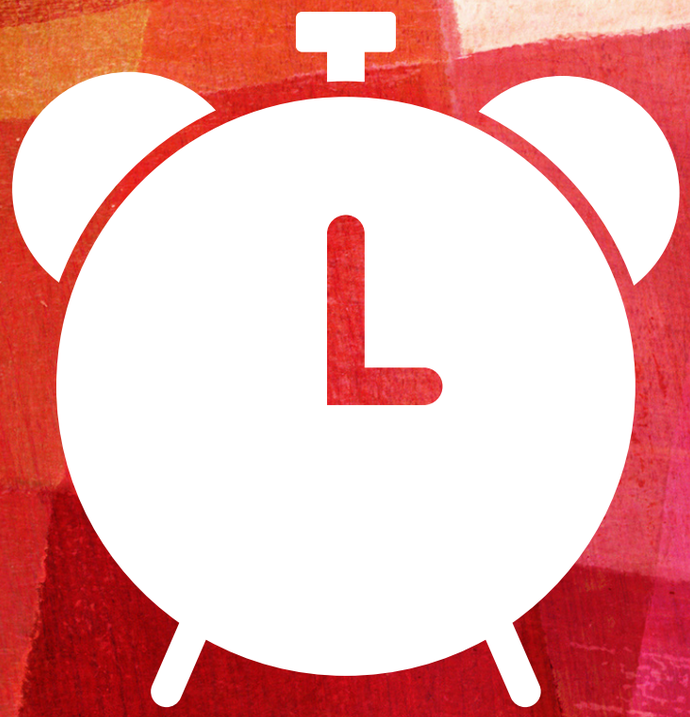
Using Arrays

JS FUNDAMENTALS – USING ARRAYS

➤ Write a function that creates a 2 dimensional matrix as below. Your function should use two for loops.

```
[
    [ 1, 2, 3, 4 ],
    [ 5, 6, 7, 8 ],
    [9, 10, 11, 12 ]
];
```

For every row, calculate and display the sum of row elements.
For every column calculate and display the sum of column elements.



Quiz 4

Expressions & Operators

JS FUNDAMENTALS – EXPRESSIONS & OPERATORS

- Why is it preferable to use == when compared to ===, and !== when compared to != operator?

- What is the output?

a)
```
console.log( [1,2,3].length == “3” );
console.log( [1,2,3].length === 3 );
```

b)
```
var a = 10.5, b = 2;
console.log( a / b );
console.log( a % b );
```

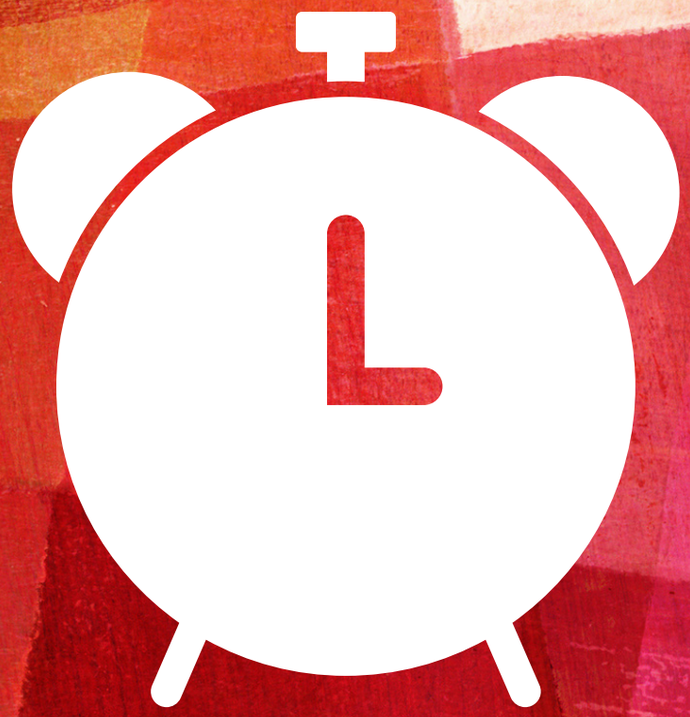
c)
```
var a = 10.5, b = 2;
console.log( a++ / b - 0.5 * 3 + 2 );
```

Experiment - 4

Expressions & Operators

JS FUNDAMENTALS – EXPRESSIONS & OPERATORS

- Execute the following and check the results
  a) console.log(typeof [1, 2, 3]);
  b) console.log([1, 2, 3] instanceof Array);
  c) console.log(typeof null);
  d) console.log(null instanceof Object);



Quiz 5

Control Flow

JS FUNDAMENTALS – CONTROL FLOW

- What data types can switch…case expressions take?
- What is the output?

a)
```
for( var i = 0; i < [1, 2, 3].length; i++ ) {
    console.log( i );
}
```

b)
```
var i = 10;
while( --i ) {
    console.log( i-- );
}
```

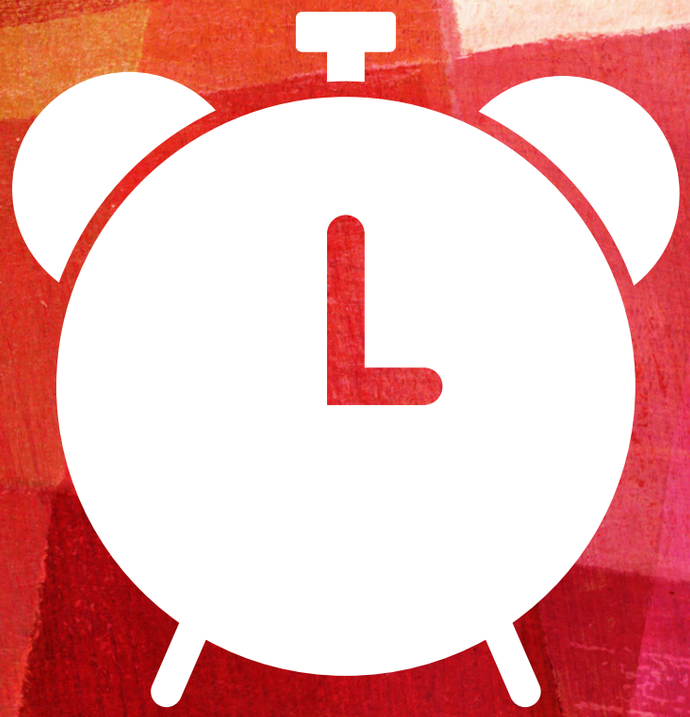
Experiment - 5

Control Flow

JS FUNDAMENTALS – CONTROL FLOW

- Given,
  var d = new Date; // d holds current date as a Date object
  h = d.getHours(); // hour of the day

  Write a snippet of code that logs either ‘Good morning’, ‘Good afternoon’, or ‘Good evening’ depending on the time of the day. Display ‘Good day’ otherwise.
  Consider
  - 5am - 12pm as morning (exclude 12pm)
  - 12pm - 4pm as afternoon (exclude 4pm)
  - 4pm - 8pm as evening (exclude 8pm)



Quiz 6

Function Declaration & Usage

FUNCTION DECLARATION & USAGE

- What are the 2 important ways of declaring a function?
- What is the output in each case?

a)
```
function foo( a, b ) {
    console.log( a, b );
}
var result = foo(1);
console.log( result );
```

b)
```
function sum( a, b ) {
    return a + b;
}
console.log( sum( 1 ) );
console.log( sum( 100, 200, 300 ) );
```

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

d)
```
var x = (function( a, b ) {
    return a + b;
}());
console.log( x );
```

e)
```
var x = (function( a, b ) {
    return a + b;
}(1, 2));
console.log( x );
```

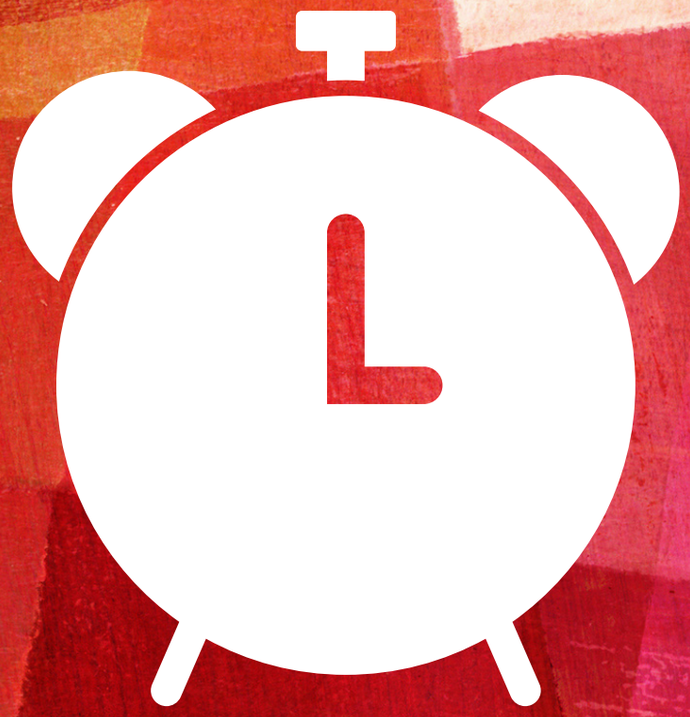
f)
```
function foo( ) {
    var f = 100;
    function sum( a, b ) { return f + a + b; }
    console.log( sum( 200, 300 ) );
}

foo( );
console.log( sum( 200, 300 ) );
```

Experiment - 6

Function Declaration & Usage

FUNCTION DECLARATION & USAGE

- Declare a function sum( ) that accepts an array of numbers and returns the sum.
  - Use the function declaration syntax and execute sum( )
  - Change it to a function expression and call sum( )
  - Change it to an IIFE and pass it an array. Store the result in a variable called sum.



Quiz 7

Handling Variable Number of Arguments

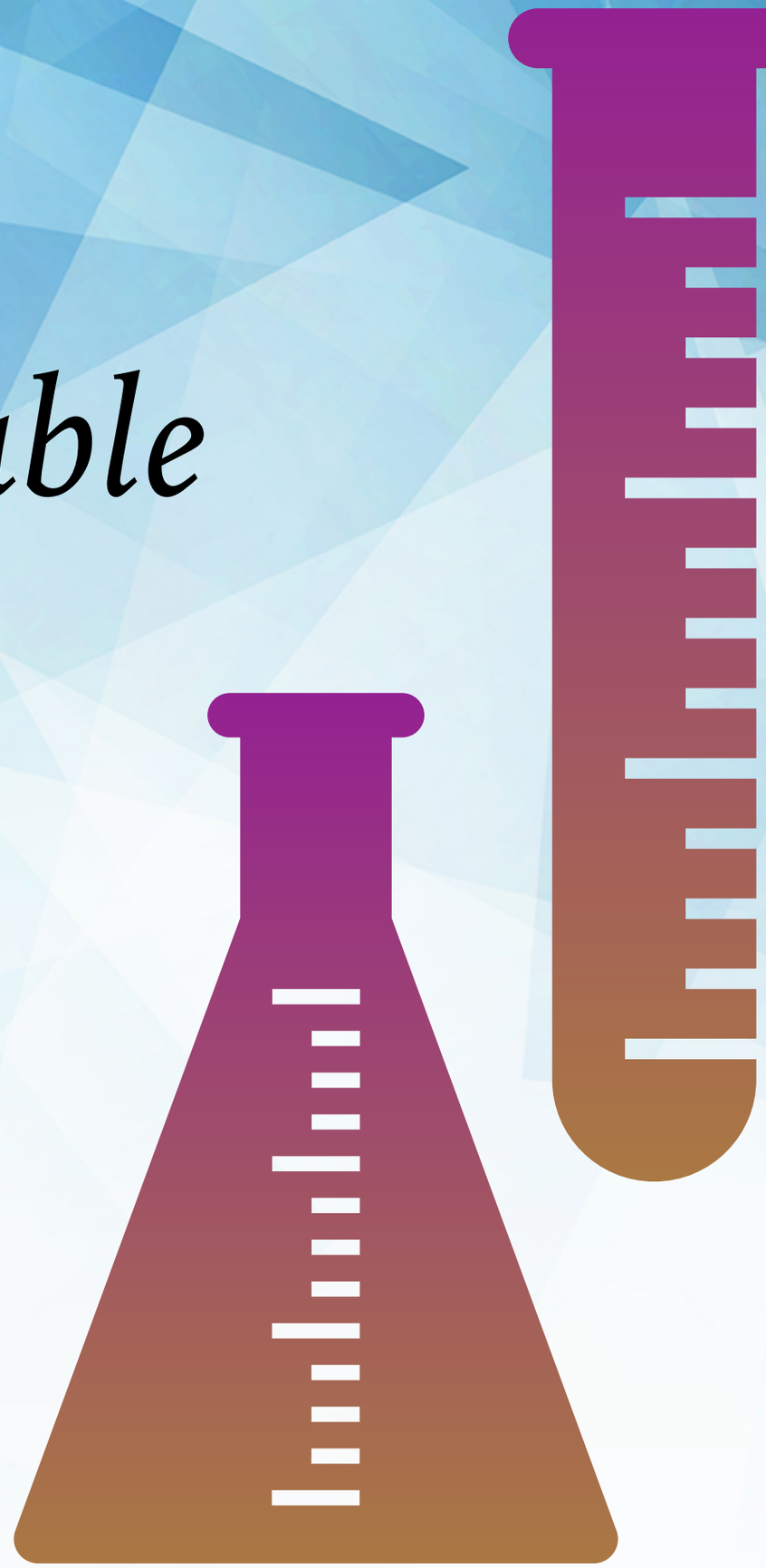
HANDLING VARIABLE NUMBER OF ARGUMENTS

- What is the output of the following?

```
function sum( ) {
    console.log( arguments, arguments.length, arguments instanceof Array );
    for( var i = 0, sum = 0; i < arguments.length; i++ ) {
        sum += arguments[i];
    }
    return sum;
}

console.log( sum( 1, 2, 3, 4 ) );
console.log( sum( 'hello', '', 'how', '', 'are', '', 'you' ) );
console.log( sum( ) );
```

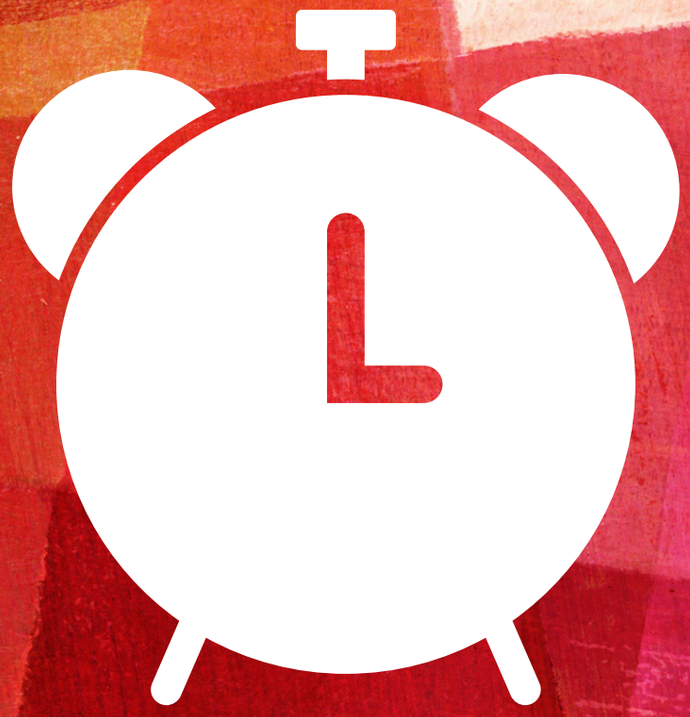
Experiment - 7

Handling Variable Number of Arguments



HANDLING VARIABLE NUMBER OF ARGUMENTS

- Write a function that calculates the maximum value among its arguments (Assume all arguments passed to the function are numbers)



Quiz 8

Callbacks - Passing Functions as Arguments

CALLBACKS – PASSING FUNCTIONS AS ARGUMENTS

- Call the sum function on the indicated line, passing it a function that logs the result to console. In what order do the logs messages appear?

```
function sum( numbersArray, onSumCalculated ) {
    setTimeout(
        function( ) {
            for( var i = 0, sum = 0; i < numbersArray.length; i++ ) {
                sum += numbersArray[i];
            }
            onSumCalculated( sum );
        },
        1000
    );
}

var numbersArray = [1, 2, 3, 4];
sum( _____, _____ );
console.log( 'After call to sum( )' );
```

- Find out what happens if you change the 2nd argument of setTimeout() from 1000 to 0.

Experiment - 8

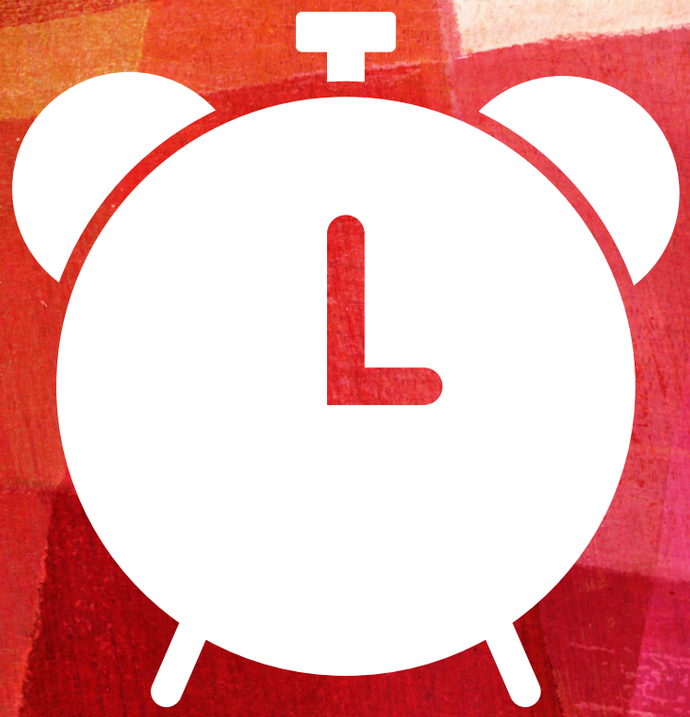
Callbacks - Passing Functions as Arguments

CALLBACKS – PASSING FUNCTIONS AS ARGUMENTS

- Write a function that calculates the sum of its arguments.
- Modify it so that the function now, optionally, accepts another function as its last argument. The passed function accepts a number and returns a number. In this case, the sum should be calculated after applying the passed function to each argument.

Example:

```
function sum( 1, 2, 3, transform ) {
    // code for sum goes in here
    // ...
}
sum( 1, 2, 3 ); // 6
sum( 1, 2, 3, function square( n ) { return n * n; } ); // 14
```



Quiz 9

Returning Functions

RETURNING FUNCTIONS

- Complete the blanks so that an appropriate greeting message gets logged.

```
function greetingFactory( time ) {
    if( time < 12 ) {
        return function( ) { console.log( ‘Good morning’ ); };
    } else if( time >= 12 && time < 4 ) {
        return function( ) { console.log( ‘Good afternoon’ ); };
    } else {
        return function( ) { console.log( ‘Good evening’ ); };
    }
}
var time = 8;
var greetingFor8Am = _____;
greetingFor8AM( );
```

- Now change it so that it logs ‘Have a great morning’

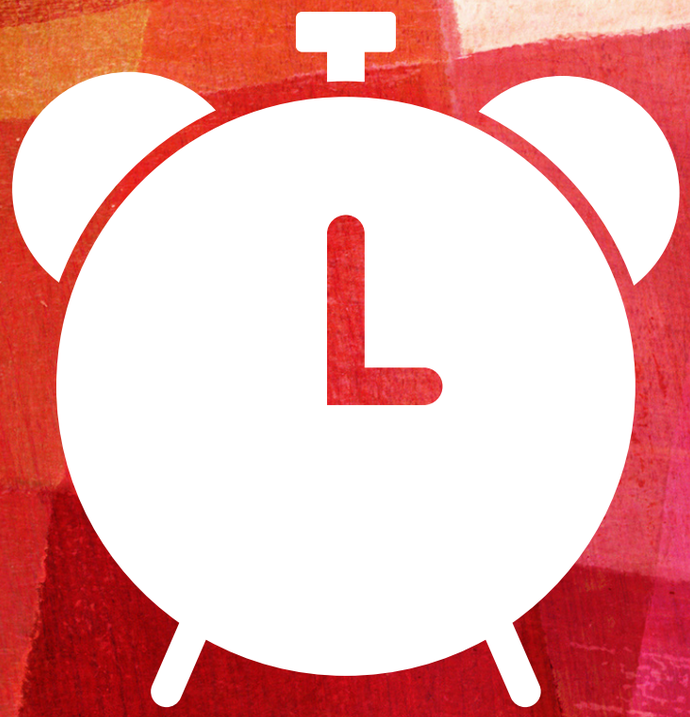
```
function greetingFactory( time ) {
    if( time < 12 ) {
        return function( greeting ) { console.log( greeting + ‘ morning’ ); };
    } else if( time >= 12 && time < 4 ) {
        return function( greeting ) { console.log( ‘greeting’ + ‘ afternoon’ ); };
    } else {
        return function( greeting ) { console.log( greeting + ‘ evening’ ); };
    }
}
var time = 8;
greetingFactory( _____ )( _____ );
```

Experiment - 9

Returning Functions

RETURNING FUNCTIONS

- Write a function that returns
  - a squaring function when 2 is passed
  - a cubing function when 3 is passed



Quiz 10

Closures

CLOSURES

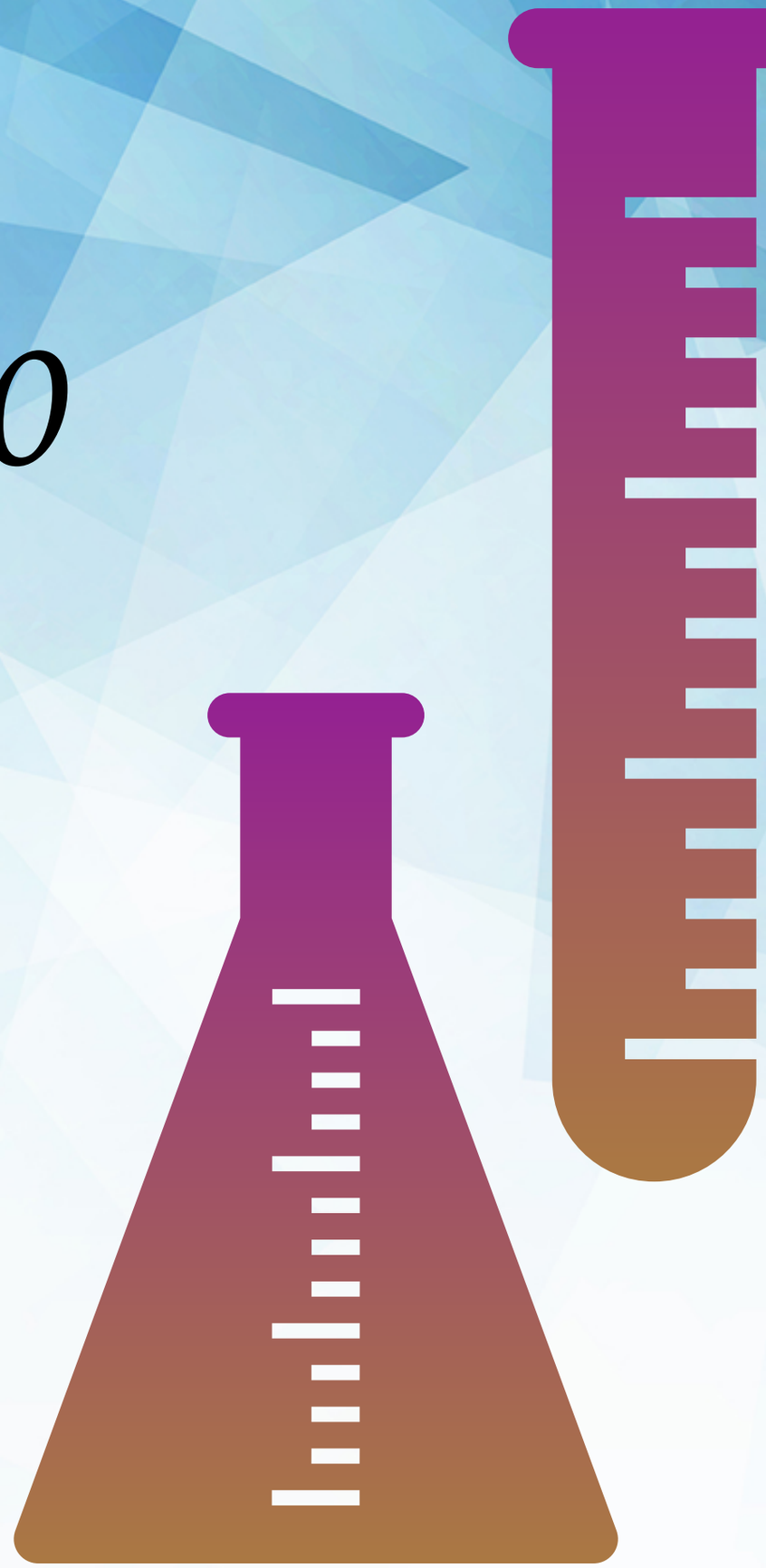
- Create a function that greets ‘Good morning xyz’ by calling the factory function below.

```
function greetingFactory( greeting ) {
    return function( name ) {
        console.log( greeting + ‘ ‘ + name + ‘!’ );
    };
}
var goodMorning = _____;
_____( _____ );
_____( _____ );
```

- In the above snippet, what is the scope of variables greeting and name? What about the lifetime of these variables? Note how the greeting variable is ‘remembered’ by the inner function every time it is called.

Experiment - 10

Closures



CLOSURES

- Write a function sum(a, b) that returns the sum of a and b.
- Define another function called bind(fn, b), that accepts a function fn and a number b. Bind returns a function that accepts an argument **a** and returns the result of calling fn(a, b).
- You should be able to use sum in this fashion

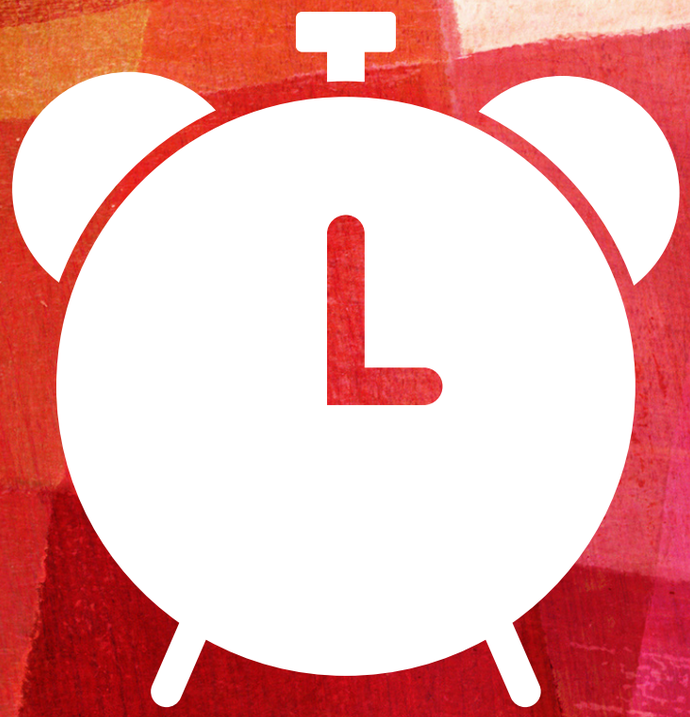
  sum(1, 2); // 3

  var incrFn = bind(sum, 1);

  incrFn(3); // 4

  incrFn(4); // 5

  Essentially, you now have incrFn as a special case of sum( ) that always adds 1 to the passed argument.



Quiz 11

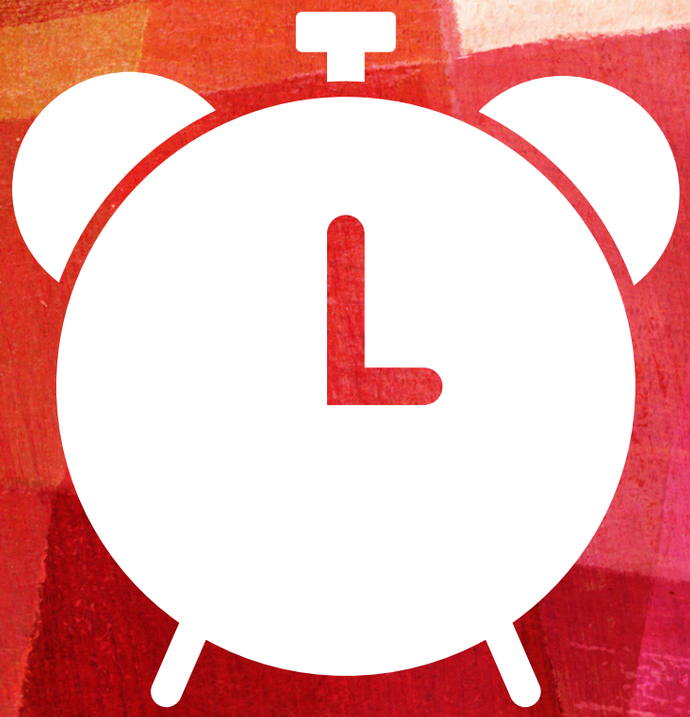
Object Declaration

OBJECT DECLARATION

- Point out the errors in the following object declaration.

```
var cricketer = {
    “name”: ‘Sachin Tendulkar’,
    1dayHighest: 200;
    playsFor: [“Mumbai Indians”, “India”],
    ‘contact’: {
        email: sachin.tendulkar@bcci.com,
        phone: ‘+91-9876543210’
    },
    function getPhone( ) {
        return phone;
    },
    function addTeam( name ) {
        playsFor.push( name );
    },
};
```

- Declare an empty object using the Object( ) constructor. Add a name property to it and set it’s value to ‘Dravid’.



Quiz 12

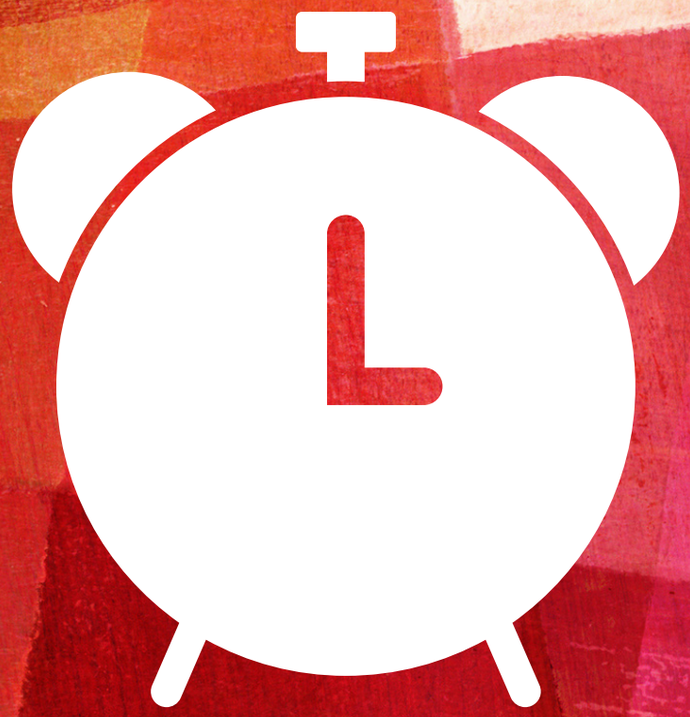
Accessing Properties & Methods

ACCESSING PROPERTIES & METHODS

- Given the following object, perform the following operations.

```
var cricketer = {
    “name”: ‘Sachin Tendulkar’,
    “1dayHighest”: 200;
    playsFor: [“Mumbai Indians”, “India”],
    ‘contact’: {
        email: "sachin.tendulkar@bcci.com",
        phone: ‘+91-9876543210’
    },
    getPhone: function( ) {
        return this.phone;
    },
    addTeam: function( name ) {
        playsFor.push( name );
    }
};
```

  - Log the cricketer’s name
  - Change the one day highest score value to a string ‘200 not out’
  - Log the number of teams Sachin plays for
  - Call getPhone method to obtain Sachin’s phone number and then log it
  - Change Sachin’s phone number to some other value



Quiz 13

Adding & Deleting Properties

ADDING & DELETING PROPERTIES

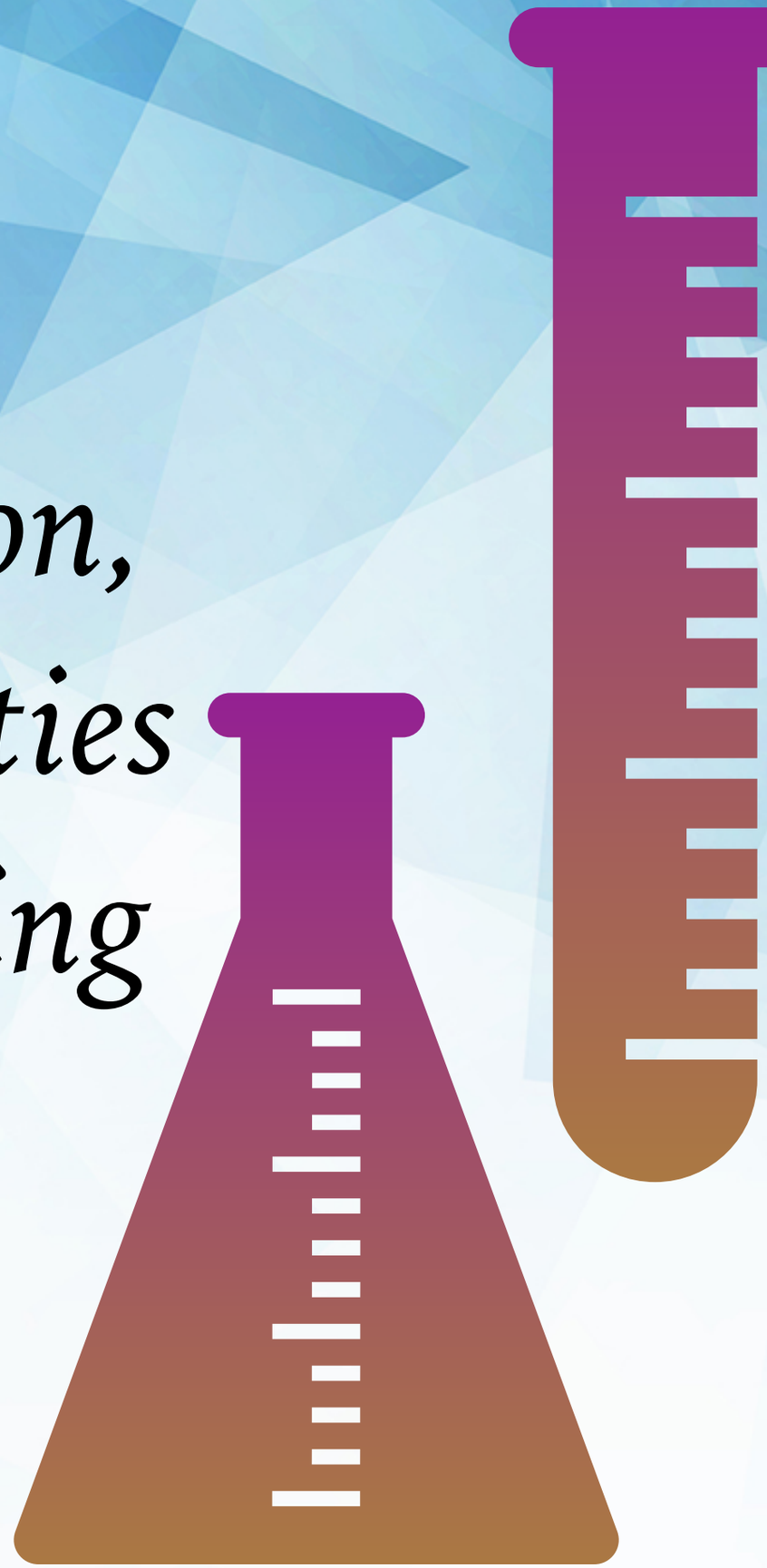
- Given the following object, perform the following operations.

```
var cricketer = {
    “name”: ‘Sachin Tendulkar’,
    “1dayHighest”: 200;
    playsFor: [“Mumbai Indians”, “India”],
    ‘contact’: {
        email: "sachin.tendulkar@bcci.com",
        phone: ‘+91-9876543210’
    },
    getPhone: function( ) {
        return this.phone;
    },
    addTeam: function( name ) {
        playsFor.push( name );
    }
};
```

  - Add a property ‘dob’ and set it to a Date object like so - new Date('24 April 1973')
  - Add a function updateEmail( ) that accepts a new email id and replaces the existing one
  - Remove property 1dayHighest
  - Sachin just messaged us saying he doesn’t like that we give out his number. Help him remove the phone number property.

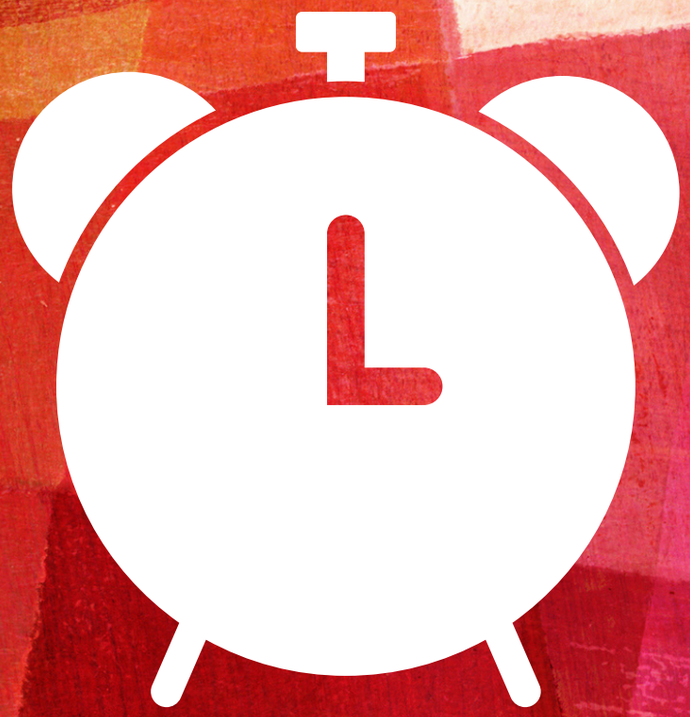
Experiment - 11, 12, 13

Object Declaration, Accessing Properties & Methods, Adding and Deleting Properties



OBJECT DECLARATION

- Declare an object addresses that stores the shipping_address and billing_address as two properties. Each address is in turn an object with properties first-line, second-line (properties need to be quoted), contact, pin-code and city. The contact is an array of email ids stored as strings.
- Declare a function getCity(type) that accepts an string argument. If type is ‘billing’ return the shipping address's city. Else return the shipping address’s city.
- Remove the contact property from each address and instead add an email property.



Quiz 14

Module Pattern

MODULE PATTERN

- What is the module pattern used for in web applications?

- Where are ‘private’ variables and methods declared?

- How are the ‘public’ variables and methods exposed?

- Can the ‘public’ methods access the ‘private’ variables and methods of the module? Why, or why not?

MODULE PATTERN

```
var shoppingCart = (function( ) {
    var items = [ ];

    _________ {
        ___________: function( _____ ) {
            items.push( _____ )
        },
        ___________: function( ) {
            console.log( _____ );
        }
    };
}( ));
```

Fill in the blanks so that the shoppingCart module has

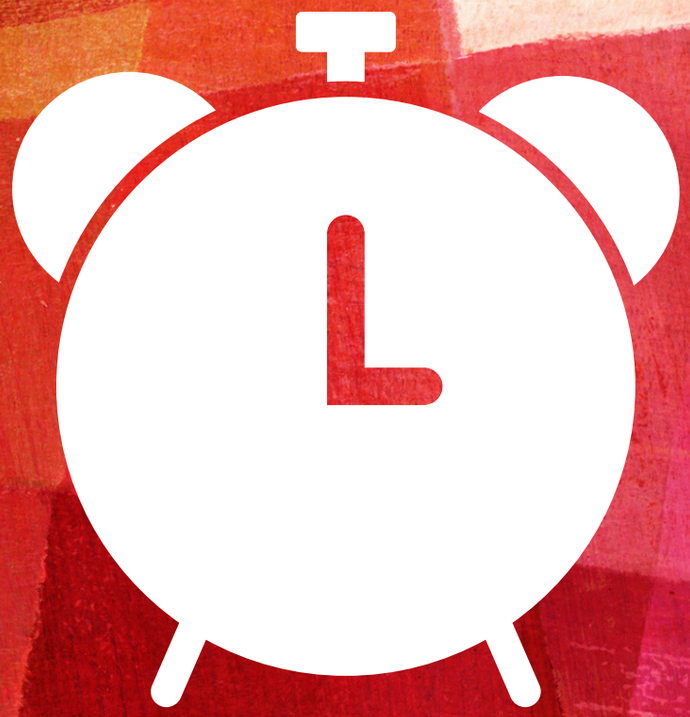
- A method addToCart that accepts an item and adds it to the items array
- A method displayCart that shows the items in the cart

Experiment - 14

Module Pattern

MODULE PATTERN

- Create a sequence id generator module. It has a ‘private’ variable _counter initialised to 0.
- The public interface should have a getNext( ) method that returns a unique number each time it is called (you can generate ids as 1, 2, 3, …
- The public interface also has a method isEmitted(x) that accepts a number and returns true/false depending on whether the number has already been generated or not.



Quiz 15

Function Context and Constructors

FUNCTION CONTEXT & CONSTRUCTORS

- What is the output in each case?

a)
```
function foo( ) {
    console.log( this );
}
foo( );
```

b)
```
obj = {
    foo: function() {
        console.log( this );
    }
};
obj.foo( );
```

c)
```
function Foo( ) {
    console.log( this );
}
console.log( new Foo( ) );
```

d)
```
function Foo( ) {
    var foo = function( ) {
        console.log( this );
    }
    foo( );
}
new Foo( );
```

e)
```
function Foo( ) {
    this.foo = function( ) {
        console.log( this );
    }
    this.foo( );
}
var x = new Foo( );
x.foo( );
```

f)
```
function Foo( ) { }
Foo.prototype.foo = function( ) {
    console.log( this );
}
new Foo( );
this.foo( );
```

FUNCTION CONTEXT & CONSTRUCTORS

- We would like to set up an _id property for the object inside the asynchronous function. Can you fix the bug to get it working?

```
function Foo( ) {
    setTimeout(function( ) {
        this._id = Math.ceil( Math.random( )* 1000000 );
    }, 0);
}
var foo = new Foo( ); // id is not present - why so??
```

FUNCTION CONTEXT & CONSTRUCTORS

- What is the output?

```
function Comedian( name ) {
    this.name = name;
    this.showName1 = function( ) {
        console.log( this.name );
    };
}
Comedian.prototype.showName2 = function( ) {
    console.log( this.name );
};

var laurel = new Foo( ‘Laurel’ );
var hardy = new Foo( ‘Hardy’ );

laurel.showName1( ); laurel.showName2( ); hardy.showName1( ); hardy.showName2( );
console.log( laurel.showName1 === hardy.showName1 );
console.log( laurel.showName2 === hardy.showName2 );
```

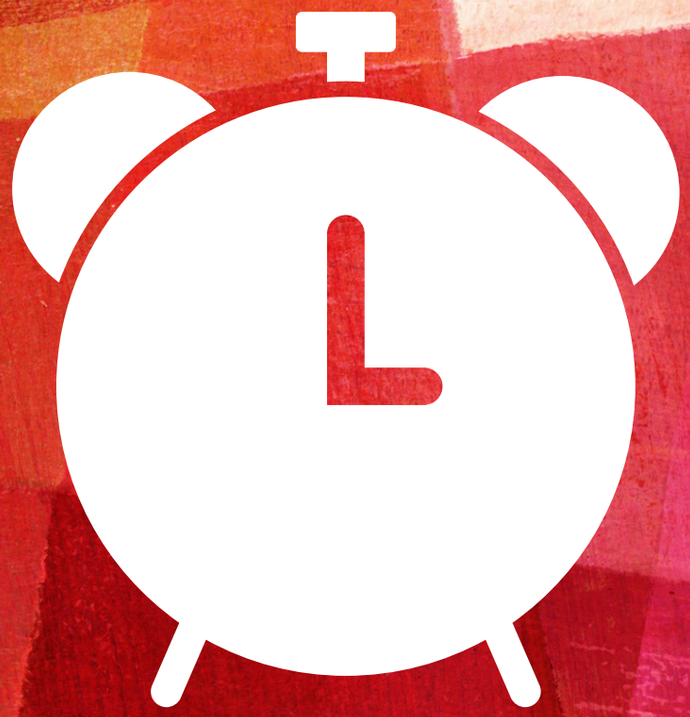
- What is the advantage of adding methods of the object to the Function’s prototype rather than within the constructor function?

Experiment - 15

Function Context and Constructors

FUNCTION CONTEXT & CONSTRUCTORS

- Create a constructor MovieCharacter. An object of this class has the following properties whose initial values are passed to the constructor
  - name: string
  - artist: string
  - punchDialogue: string
  - likes: number
  - dislikes: number
- The following functions are declared on the prototype
  - sayPunchDialogue( ) - logs the punch dialogue to console
  - getPopularity( ) - returns (likes - dislikes)
  - addLike( ), addDislike( ) - increments likes/dislikes
- Creates some MovieCharacter objects and test out their methods.



Quiz 16

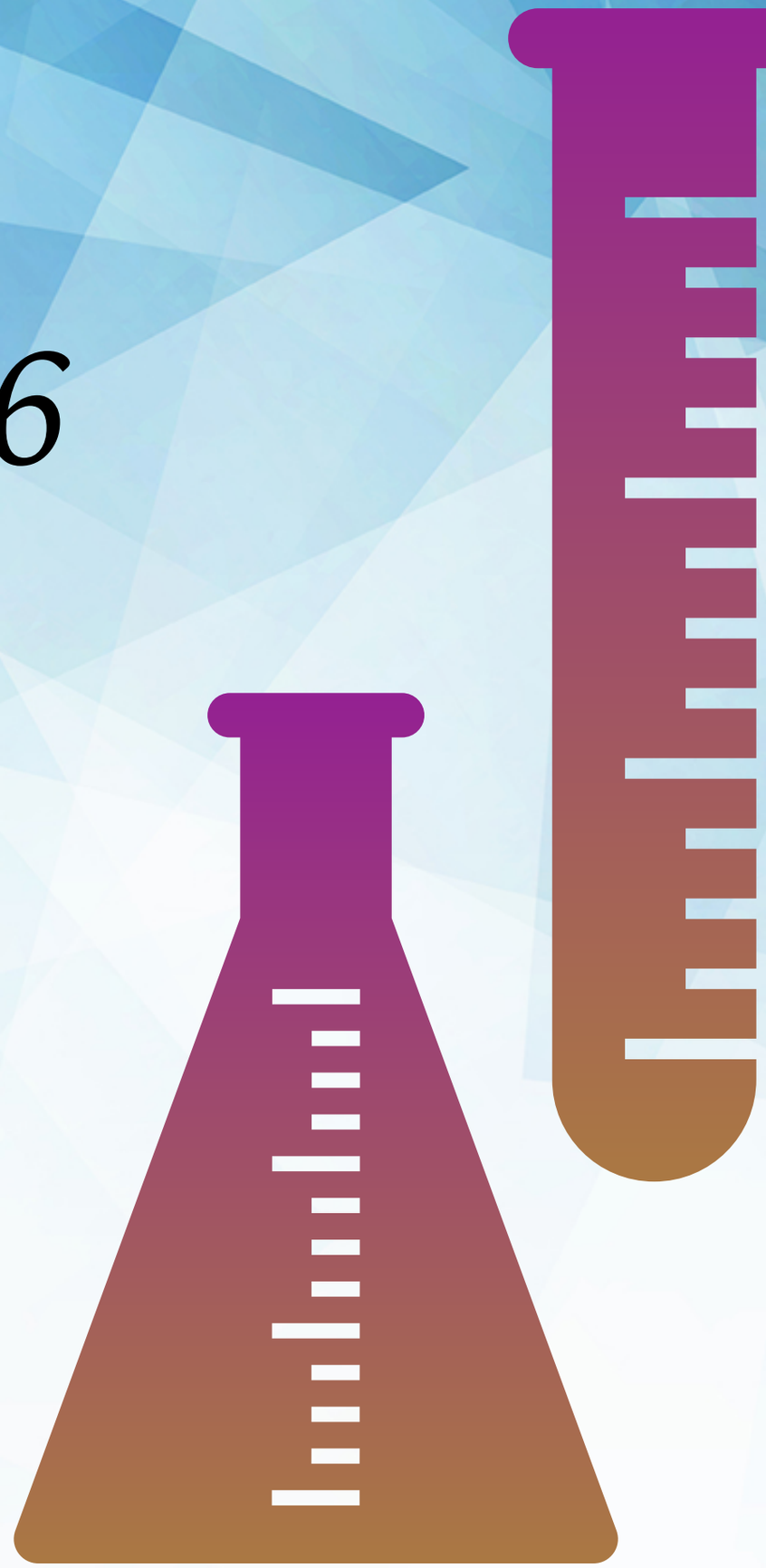
Prototypes

PROTOTYPES

- var person = {

Experiment - 16

Prototypes



PROTOTYPES

ATTRIBUTIONS

- Abstract blue background from Freepik http://www.freepik.com/free-vector/abstract-blue-background_859016.htm