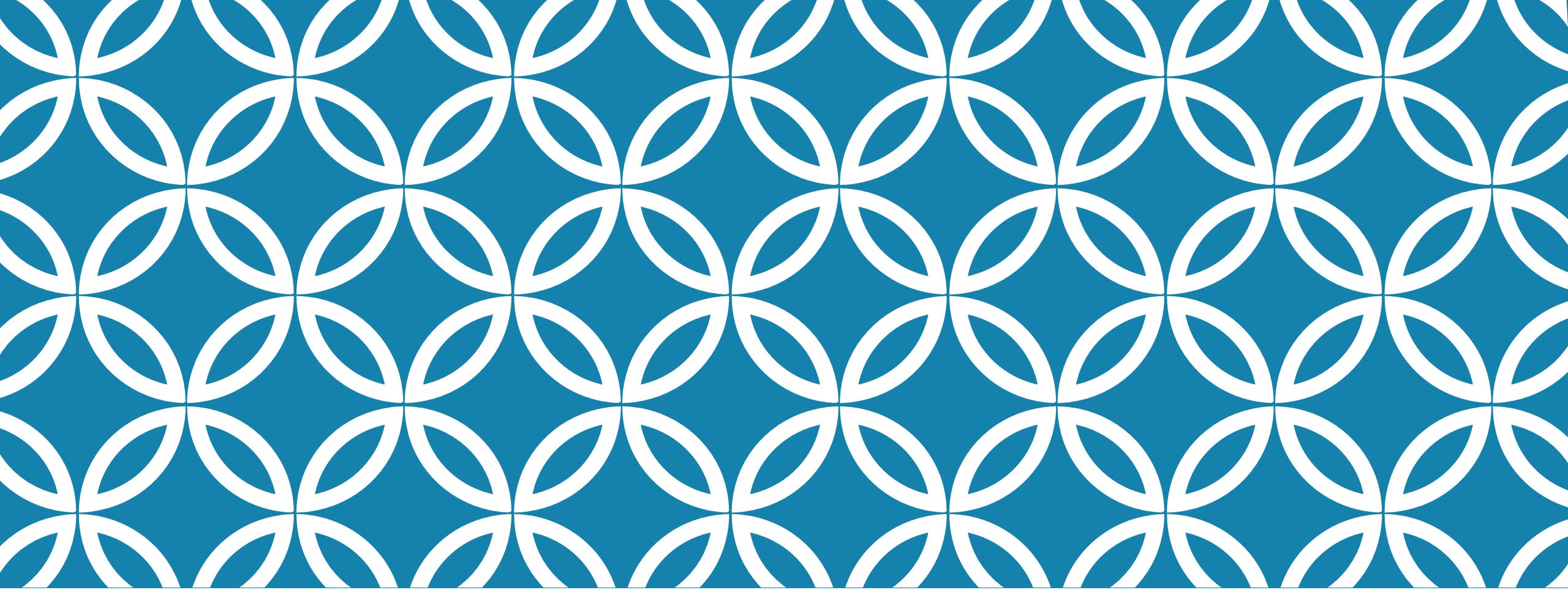

JS ESSENTIALS - WEBINAR 1

By Prashanth Puranik
Freelance Corporate Trainer and Web Developer

Brought to you by
Saturday Live Webinars
https://github.com/saturdaylive/js-essentials

TAKEAWAYS

- Function declaration vs function expressions
- Functions as first-class citizens – What it means
- Function call context (“this” keyword) – How it is assigned

Two ways to define a function*

Function Declaration

&

Function Expression

* There are other ways too – using the Function (constructor) function, arrow functions in ES2015+

The name of the function is optional in case of Function Expressions

Unlike Function Declarations, Function Expressions are NOT hoisted

That's a good thing!

Advantages of Named Function Expressions

1. Enables recursive calls
2. Function name appears in error messages
3. Clarifies intent of function to developer

Prefer Function Expressions.

Name them if it makes sense.

In JS, Functions are ***first-class citizens***

They are on par with any other type

They are (callable) objects

A first-class citizen can

1. Be assigned to a variable
2. Passed to a function
3. Returned from a function

Aside: Function that accept, and/or return other functions are said to be Higher-Order Functions

Some things that its citizenship status enables…

1. Callback pattern to handle results of async tasks
2. Patterns like those in array iterator methods (forEach, map etc.)
3. Function factories

Function call context

The “this” keyword within functions

Provides useful context for a function call

Function call context

It is associated with a **function call.**

NOT the function itself

It can be different for **different calls** of the **same function**

Determining call context (this)*

1. Defaults to global object (window in browser/global in Node.js)
2. object that invokes the method in obj.method()
3. Bound context for call() / apply()
4. Context bound using bind()
5. Newly created object for constructor invocation

* Rules are listed in increasing order of importance.

call() / apply()

One-time function invocation with a changed context

Supports partial application of arguments

bind()*

Multiple function invocations with a fixed context

Also supports partial application of arguments

* bind() returns a new function which then is invoked (likely multiple times), whereas call() and apply() result in the underlying function being called

REFERENCES

- If you are a beginner in JS you can refer https://www.w3schools.com/js/default.asp
- Mozilla Developer Network is a great reference for topics on web development
- For a thorough coverage on topics you can refer https://javascript.info/
- For a deeper understanding of topics covered in this webinar you can refer Kyle Simpson's You Don't Know JS series of books – particularly this one