

# Theme: Next Generation Web

## Subject: Innovative Browser Core

### Introduction

Today's web browser is built upon DOM + JavaScript paradigm. DOM represents a view of an HTML page, and JavaScript is used to dynamically update DOM to provide interactive web pages. Efficient DOM + JavaScript implementation affects the overall performance of a web browser. The purpose of this project is to investigate techniques to improve a web browser technology, in particular, tightly-coupled DOM + JavaScript architecture, to the next level. By analyzing today's web browsers, we address technical challenges as follows:

(1) Memory-inefficient DOM: While tree-based DOM in today's browser provides clean and logical architecture, the size of DOM is often huge in practice, and it often prevents from implementing a memory-efficient browser for resource-constraint devices such as wearable devices.

(2) Parallelism: current DOM data structure is not taking the advantage of multi-cores due to serial tree-based DOM read/manipulation algorithms.

(3) Context switch between JS and native DOM: While JavaScript modifies DOM via JS Binding interface, it makes frequent memory context switches due to different language implementation. One research idea is to implement DOM purely in JavaScript and run DOM as JavaScript objects to reduce the context switch overhead occurred in today's web browser implementation.

The aim of this project is to research, develop and optimize web browser techniques, in particular, by addressing the above challenges.

### Scope

Technical challenges include:

- Research and develop memory-efficient DOM model.
- Research and develop efficient parallelism-aware DOM model.
- Implement a web browser in JavaScript with efficient DOM model addressing the above challenges.

### Research questions

We are interested in the following research questions. These questions are not exhaustive but different research questions are open to discuss with research partners.

- What would be the most memory-effective way of representing DOM data structure? If the current tree-based data structure is not efficient, is there a new way of representing DOM?
- What would be the optimal way of reading/updating DOM? Is parallelism help? If so, what would be efficient parallelism-aware DOM reading/updating algorithms?
- What would be an efficient technique to improve the DOM and JavaScript interaction performance? How do we reduce the overhead resulted from different language interaction?

### Expected Deliverables

The following is open to discussion:

- Design of new memory-efficient and/or parallelism-aware data structure and algorithms
- Detailed progress reports every 3 months summarizing accomplishments.
- Prototype implementation
- Patents with Samsung Electronics (if agreed)
- Research papers (if agreed)