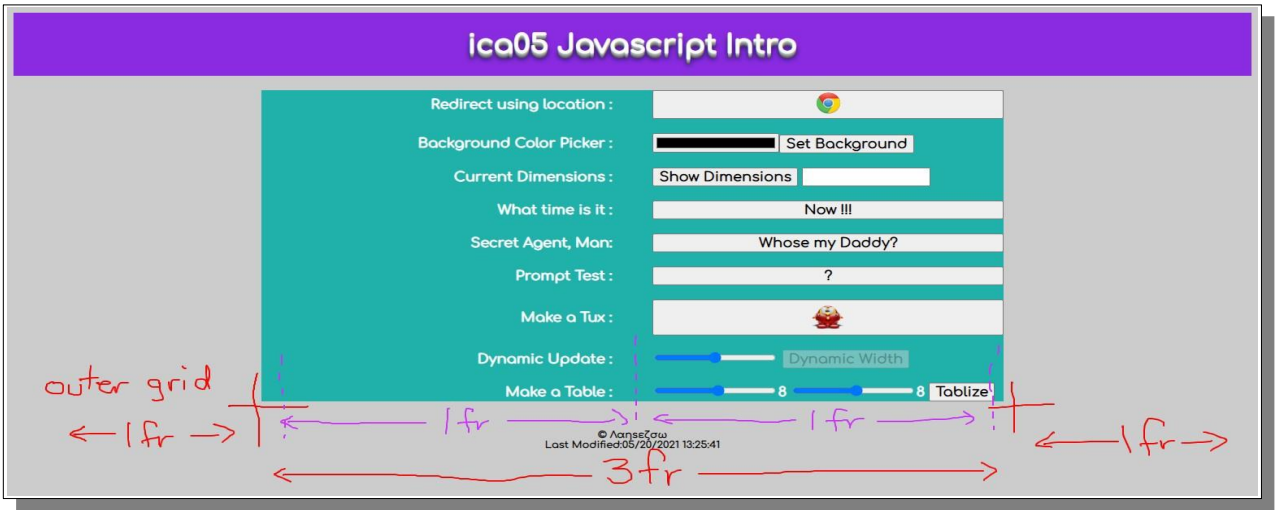
School of Applied Sciences And Technology
CMPE2000 – ICA06 – Javascript Intro

In this exercise you will implement basic use some javascript events and core DOM methods.

As usual, create a new folder called ica06, create a html and css file named and linked appropriately. Same standing orders for header and footer. We will continue to use grid for layout, using an external style sheet.



Create your layout using simple nested grids. The header, followed by a container of 3 columns, with the center section being another 2 column grid to the label/control layout . Below your main section is our full width output div, followed by the footer.
Our mobile-first design will collapse like you expect, in order with captions preceding their buttons or input elements.

In this exercise, we will use the new tag <button></button> to implement our buttons. The biggest benefit of the new type is to allow content (like images) to be put inside the button. Ensure you specify a type="button", although it seems redundant, different browsers treat the default differently between submit, reset, and button.

***No forms were harmed (or used) in this ica.**

Remember all event callback methods, if properly bound, get an automatic local variable called "this" (kinda like c#) which is a reference to the object which invoked the event, we can use this info to access the invoking object, whether input textbox or a button. **Use of "this"** rather than explicit querySelector() calls **is preferred**.

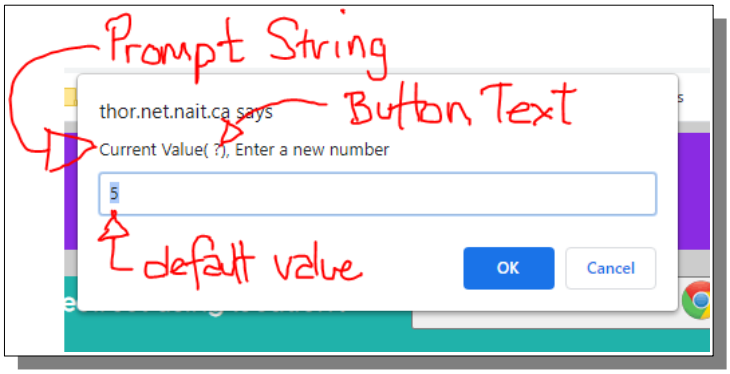
The following rubric defines checkoff requirements and associated marks. Non compliant submissions will not be marked unless otherwise indicated.

JS Standing orders must be evident for a submission to be evaluated.

Base 70% - All requirements met, no part marks

- Mobile / Desktop layout works, grid layout complete, labels and controls as shown.
- Redirect using Location
  - button with browser image directs to google.com using location property
- Background Color Picker
  - Color control correctly shown, selected color is extracted and used as body background color when [Set Background] button is clicked.
- [Show Dimensions] : using the window object to place the innerWidth and innerHeight dimensions into the adjacent textbox in the form `[ Width, Height ]` - with brackets

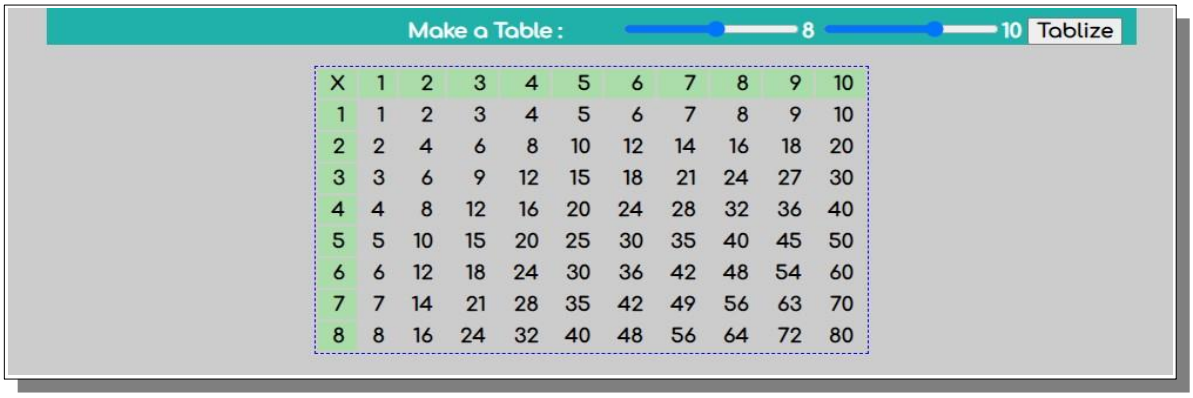
- What time is it : [Now] - clicking the [Now !] button will use the Date() object and use the toTimeString() function to populate an alert.
- [Whose my Daddy ?] - clicking this button will use the navigator object to retrieve the userAgent and populate the button contents (replacing the existing button text). Use of "this" mandatory, non-compliant if any object retrieval document methods used.
- Prompt Test / [?]
  - using window object's prompt method, build a prompt string as shown from the current button text (shown is starting "?", but it will vary. Use "this" to retrieve the current button text.
  - Use 5 as the default input value
  - If the user chose [Cancel] or the user input is not a number, output an error message to the console in the form : `[ RESP ] is Null or NaN` where RESP is the user response
  - Otherwise, if the user input was OK, replace the current button text with the user's response. i.e. If [Ok] was selected below, 5 would be the response and become the button text



Enhanced elements 30%

- Make a Tux / image button as shown.
  - Define a global numeric variable to hold tux height, init to 40
  - When the button is clicked, use strings to create a new HTML image element, setting the height property to your global variable. Append this HTML encoded image string to the end of the DIV holding the label text.
  - Increment the global height by 20, the next use will create a larger image
  - The tuxes will keep growing, and eventually wrap within the DIV..

- Dynamic Update : A range control beside a [Dynamic Width] button
  - set the min and max of the range to 50 and 300
  - using a mousemove event on the range control, set the style width to the value of the range control
  - use of "this" is require – which object will "this" refer to in your mouse event handler ?
  - Depending on page size, the button may wrap within the container DIV (yes, the range and button live in DIV together, taking that column spot)
- Make a Table [Tablize] :
  - Write a javascript function that will accept row and column numbers.
    - It will take the 2 parameters representing row and column and return a string (which may be a table construct, or an error message).
    - Validate that values exist and are positive numbers, if not return an appropriate* error message.
    - Otherwise, construct a Multiplication table using a string to incrementally build it up using nested loops. Return this string.
  - In your button event callback, extract the row and column values from the range controls and invoke your function with these values.
    - assign the return object to the innerHTML of the output DIV sitting under the main input form



X	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80

For extra fun, but no marks, you see label elements beside the ranges, Add onchange events to each bound to a single function which will retrieve the appropriate values and update the labels to show the value when the user releases the slider. Use innerHTML or textContent properties of the label.