IS216
Web Application Development II

Session 4

JavaScript – Part 1 (Basics)

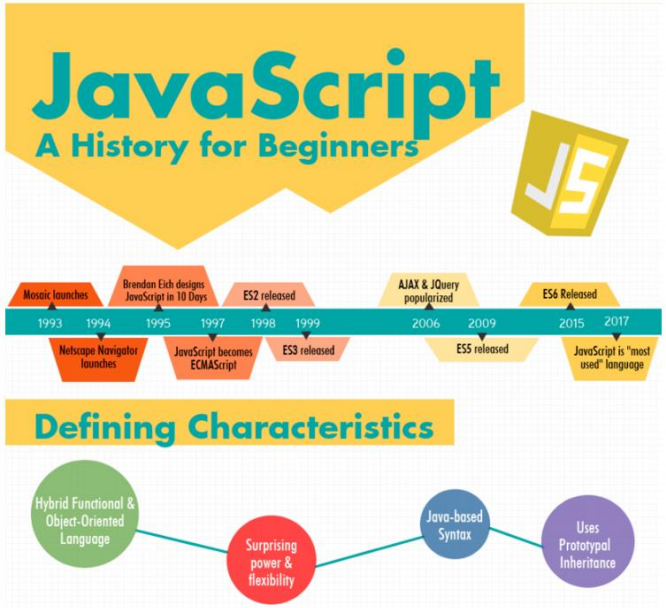
K. J. Shim

Sections: G1 / G2 / G3 / G11

Agenda

CSS Bootstrap
Container
Grid System

JavaScript - Basics
Syntax
Data types
Operators
Conditional statements
Loops
Functions
String operations
Array operations



{ FRONT-END }
HTML
CSS
JavaScript

< BACK-END >
.NET
PHP
Node.js

Sessions 5/6/7

We meet in **SCIS SR 2.4** *(physical/f2f)*

- **Local students**: your default learning mode is physical/f2f
  - *Lessons will be online/Zoom for overseas students (same lesson day/time – a.k.a. hybrid classroom)*
  - If you're unable to come to school, just drop Kyong/TAs a short Slack DM.
    - No need to present MCs or explain your circumstance unless you think Kyong/TAs should know.
    - We want to make sure you're ALIVE and OKAY (and didn't go MIA).

- Choose a **seat** in **Session 5** and ***stick to it*** (you cannot change to another seat) → **SMU Policy**
  - In view of this, you will do **pair programming** with your **neighbor** (of your choice). So, please sit with someone you will do pair programming with.

- If this plan (of going physical/f2f) changes, Kyong/TAs will inform you as soon as possible via **email** and in **Slack #general**. *Please pay attention to ALL email messages from SMU offices – especially those from the Provost / Associate Provost / Vice Provost / Prof Shim offices.*

Project (25%)

1. **Register Group Members** in **eLearn** → Group Info → **Groups**
   - https://elearn.smu.edu.sg/d2l/home/299951
2. **Project Briefing Document**
   - https://smu.sg/2021-is216-project-briefing
3. Once we have your **Group Number**, we will add your group members to a **Proposal Document** (Google Doc)
   - ***Sample*** **Proposal**: https://smu.sg/2021-supreme-project-sample-proposal
   - **[DEADLINE]** *Initial* proposal (with all fields filled out) due (Week 8) on **8-OCT-2021 (FRI) 12PM SG Time**
   - Teaching team will review and add **comments** inside your **Proposal Document** by (Week 9) **11-Oct-2021 (Monday) 12PM SG Time**. You will need to review our comments and **address** by **replying**.
   - **[DEADLINE]** *Final* proposal (with all fields filled out) due (Week 9) on **17-OCT-2021 (SUN) 12PM SG Time**
   - *You can ask **Kyong** to review your proposal ANYTIME before **8-OCT-2021** – she will provide fast feedback and advice.* **Slack DM** her anytime.
4. **[GitHub Classroom]** Every group must push their project source files into a private repo in **is216-supreme Classroom**.
   - **Kyong** will send an invitation to each group by **Week 6 Monday**.

Source Code Files

eLearn → Content → Session 4 → In Class → **Week4.zip**

- **Unzip** it into your **webroot** (*any meaningful sub-directory), for example:*
  - (WAMP) C:\wamp64**www**\is216\…**Week4**
  - (MAMP) /Applications/MAMP/**htdocs**/is216/…/**Week4**

- You don't have to follow the above path – it's just an example.
  - But we DO strongly encourage you to keep source code files **organized** so that you can easily **search** them during **Lab Tests**

Follow Me (Code Together)

- **CSS Bootstrap**
  1. default.html → container.html
  2. default2.html → grid.html
  3. (Pair Programming) **Activity 1**
  4. Week4 → FollowMe → pizza → home_bootstrap.html *(my solution)*
     - Week4 → FollowMe → pizza → wireframe_images *(2 sets of wireframes – Desktop vs. Mobile)*
     - Or, you can check out my **Figma** wireframes at https://bit.ly/3l3yrpf
- **JavaScript**
  1. demo.html
  2. basics.html
  3. (Pair Programming) **Activity 2**

Bootstrap - Grid System - Breakpoints

***** container (container-sm) *****

xs	< 576px	100%
sm	>= 576px	540px
md	>= 768px	720px
lg	>= 992px	960px
xl	>= 1200px	1140px
xxl	>= 1400px	1320px

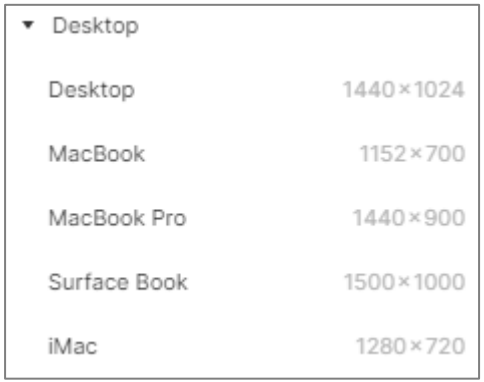
***** container-fluid *****

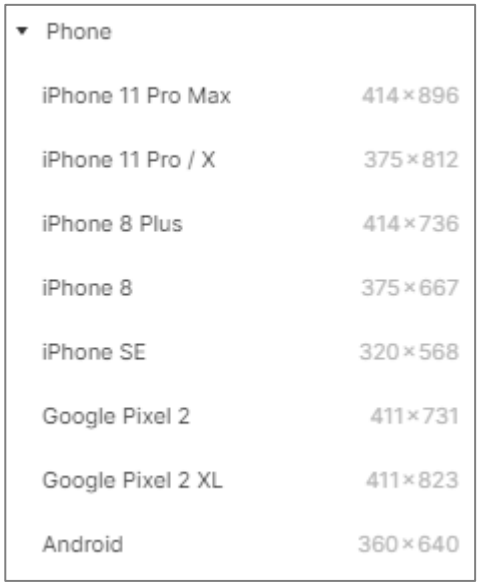
always spans 100% of screen width

Link: https://getbootstrap.com/docs/5.1/layout/grid/

Figma – Screen Dimensions

https://www.figma.com/





Activity 1: Justin Grid (⭐⭐)

1. Go to **FollowMe → grid → exercise1.html**
2. Complete **exercise1.html** and make it **responsive** using **Bootstrap** for the following screen sizes:
   - **Macbook Pro** (**1440**px by 900px)
   - **iPhone 11 Pro Max** (**414**px by 896px)

*See the next 4 slides for **instructions***

Back to Menu

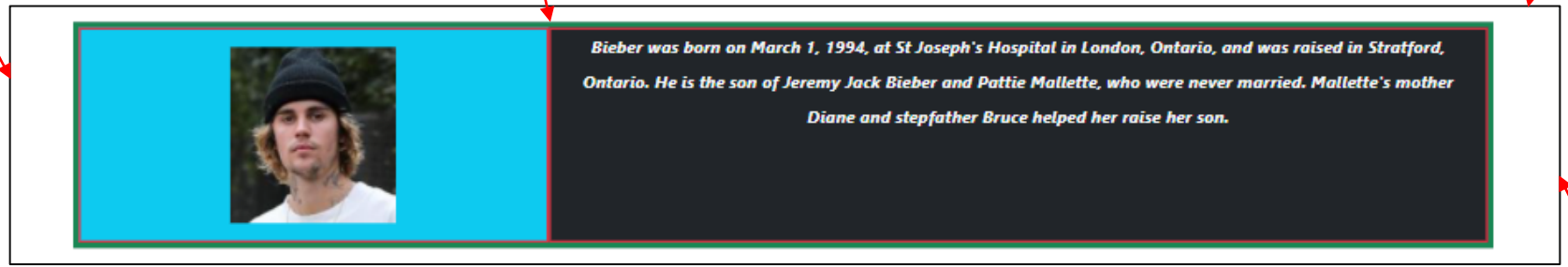
Activity 1: Justin Grid *(continued...)*

exercise1.html when viewed from **Macbook Pro** (**1440**px by 900px)

- This container contains two **boxes** inside.
  → The left box (blue background) contains an image. The right box (black background) contains text.
- This container uses the thickest border available in Bootstrap border class. Border color is green (as shown).
- Use appropriate grid system to create two **boxes** side-by-side.
- This container has **top margin** worth **16px** ($spacer).

Top border of web browser

Left border of web browser

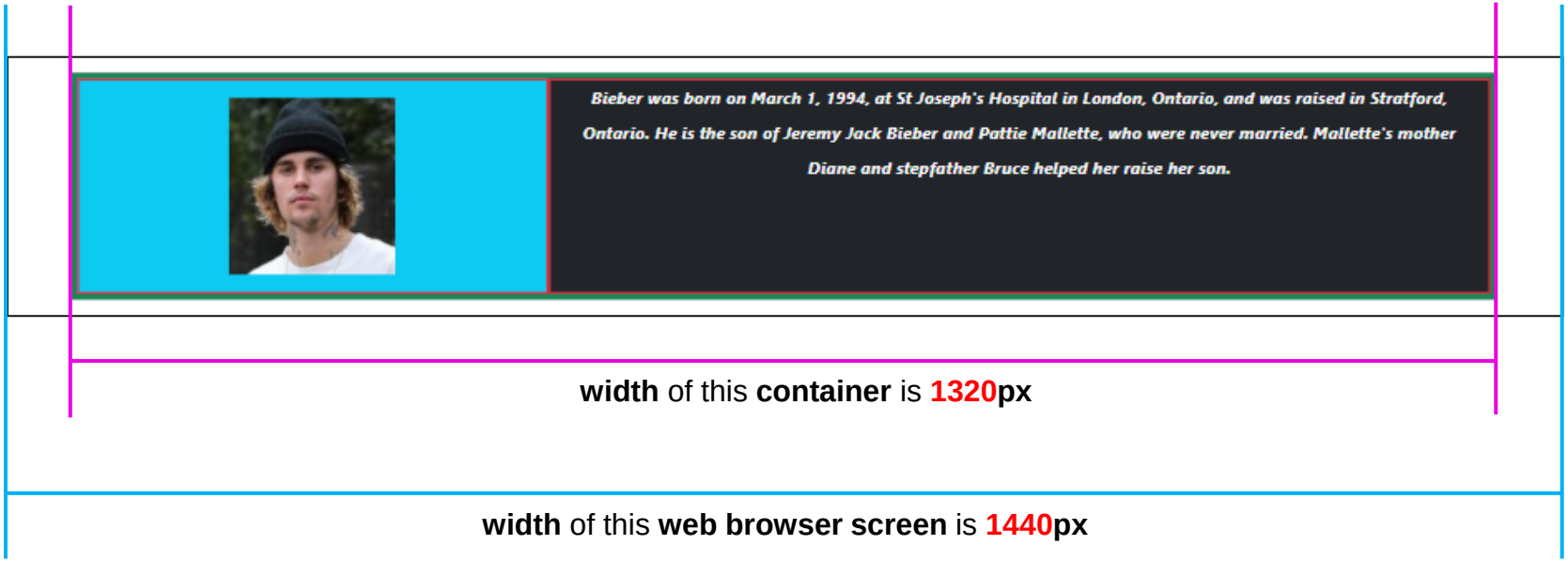


Right border of web browser

Back to Menu

Activity 1: Justin Grid *(continued...)*

exercise1.html when viewed from **Macbook Pro** (**1440**px by 900px)

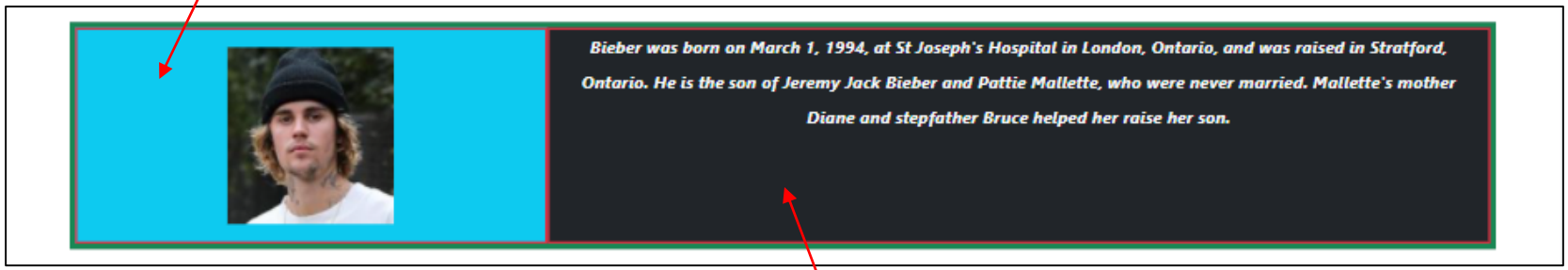


Back to Menu

Activity 1: Justin Grid *(continued…)*

exercise1.html when viewed from **Macbook Pro** (**1440**px by 900px)

- This left **box** takes up about **33.333%** width of its (ancestor) container.
- Uses background color (as shown).
- Uses 2nd smallest border thickness available in Bootstrap border class. Border color is red (as shown).
- This **box** has **top padding** and **bottom padding** – both worth **16px** ($spacer).
- The image is horizontally centered.



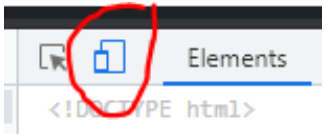
- This right **box** takes up about **66.666666666667%** width of its (ancestor) container.
- Uses background color (as shown).
- Text is in **paragraph** and **centered**. Color is **white**. Text is **bold** and **italic**. Text uses **large (lg) line height**.
- Uses 2nd smallest border thickness available in Bootstrap border class. Border color is red (as shown).
- This **box** does not have any margin or padding set.

Back to Menu

Activity 1: Justin Grid

exercise1.html when viewed from **iPhone 11 Pro Max** (**414**px by 896px)

- When viewed in a **Macbook Pro** screen (1440px width), this **box** is displayed on the **right side** of **the other box** (containing an image).
- However, when viewed in a smaller **iPhone 11 Pro Max** screen, this **box** appears shifted down – it's displayed below the other box.
- Please use **Chrome → Developer Tools → Toggle device toolbar** to test exercise1.html



Back to Menu

Activity 2: JavaScript Marathon

1. Go to **FollowMe → javascript → exercise2.html**
2. **[Task #1⭐]** Given a multi-dimensional array, print the following in **Console**

```
Ambrose with GPA 2.7 is handsome
Darren with GPA 3.8 is not handsome
Stevie with GPA 2.9 is not handsome
Gerry with GPA 3.9 is handsome
Jewel with GPA 2.6 is not handsome
Emily with GPA 3.4 is handsome
Jia Le with GPA 3 is not handsome
```

3. **[Task #2⭐½]** Loop through an **Object** such that your code prints the following in **Console**

```
code: IS216
title: Web Application Development II
core: true
professors: Shar Lwin Khin, Mok Heng Ngee, Sun Jun, and Kyong Shim
```

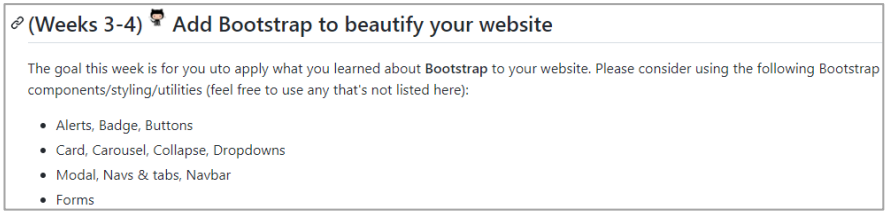
   - Professor names are separated by **comma** (,)
   - The *last professor* in the array is printed after **and**

4. **[Task #3⭐⭐]** Given 2 strings (str1 & str2), return an array of common characters
   - Assume str1 and str2 both do NOT contain any whitespaces.
   - Assume the strings are NOT case-sensitive. E.g. "a" is same as "A"
   - The return array should not contain duplicates. Only lower-case characters are allowed in the array.
   - Test Cases 1/2/3 are provided for you already (do not remove them)

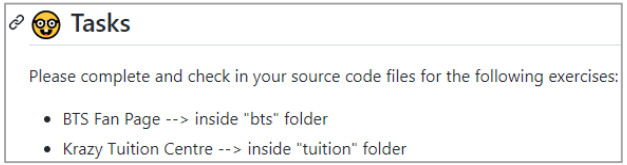
Back to Menu

Things to Finish Before Session 5

1. Complete:
   - **JavaScript – Part 1 (Basics) – Challenges 6, 7, 8, 9**
   - **Resource files** can be downloaded from this link
2. "My Profile" GitHub assignment ***Continue to work on Week 3 tasks, label it as (Weeks 3-4)***



3. **[Individual]** Complete and ***push*** "bts" and "tuition" source code files to **GitHub**
   - Click on this **invite link**: https://classroom.github.com/a/ihw9jmR4



Please try to complete this by Week 6

4. eLearn → Journal → **CSS** *(Due Week 5 Monday 9AM)*
5. eLearn → Content → Session 5 → Before Class
   - Watch the video
   - Complete [Session 5] Pre-Class Quiz (JavaScript – Part 2)

JavaScript: Hello World

Example

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <script>
      document.write("Hello World!")
    </script>
  </body>
</html>
```

Note

Semicolon at the end of a statement is **optional** (but recommended) if there is only one statement at the line.

The “document” object represents the HTML document that is displayed in that window.

You can program with
- VSC+Chrome
- JSFiddle.net
- CodePen.io

JavaScript in ‹head›

Javascript code can be placed in both HTML <head> and <body> sections.

Place Javascript code in <head> section, only if it needs to be executed before the page is rendered.

Otherwise, put it at the bottom of <body>.

Examples

```
<html>
   <head>
          <script>…</script>
   </head>
   <body>
          <script>…</script>
   </body>
</html>
```

JavaScript : Syntax

Declaration and assignment

```
// How to declare variables
var x, y, z;
const PI = 3.14159265359;


// How to assign values
x = 1;
y = 2;

// How to compute values
z = x + y;
```

Commenting

```
//this is a single line comment

/*

The following function has the following specification.

Precondition: ...

Postcondition: ...

*/
```

JavaScript : Keywords

var	Declares a variable
const	Declares a variable (cannot be reassigned)
if … else	Branches depending on some condition
for	Loop for some number of times
do … while	Loops while some condition is satisfied
continue	Jumps out of a loop and starts at the top
break	Terminates the containing loop
function	Declares a function
return	Exits a function
try … catch ...	Tries something and catch/handle exceptions

Case sensitive

Naming Convention

Underscore: e.g.

first_name, last_name, master_card, inter_city

Upper Camel Case (Pascal Case): e.g.

FirstName, LastName, MasterCard, InterCity

Lower Camel Case: e.g.

firstName, lastName, masterCard, interCity

Data Types

JavaScript is dynamically typed.

Types of variables are generally not known at compile time

Type checks are performed mostly at run time.

Examples

```
var boo = 1 > 0;  // Boolean, boo == true

var length = 16;  // Number

var firstName = "Johnson"; // String

var schools = ["sis", "soa", "sol"];  // Array

var x = {name: "John", age:21};    // Object

var y;  // typeof y == 'undefined'
```

Numbers

JavaScript does not define different types of numbers.

JavaScript numbers are always stored as double precision floating point numbers.

Examples

```
// Written with decimals
var x1 = 34.00;

// Written without decimals
var x2 = 34;

//Scientific notations
var y = 123e5;   // 12300000
var z = 123e-5;  // 0.00123
```

Strings

String literals are written in quotes -- single or double quotes

There are many functions on strings.

Examples

```
//concatenation
"txt1" + 'txt2' //result: "txt1txt2"

//string length
"Txt1".length   //result: 4

//create the string: Love all; “trust” a few
'Love all; "trust" a few'
//or through escaping
"Love all; \"trust\" a few"
```

Arrays

JavaScript arrays are written with square brackets [ ].

Array items are separated by commas.

Indexes start from zero, i.e., the first item is [0], second is [1], etc.

Example

```
var schools = ["SIS", "SOA", "SOL"];

schools[1]; //result: SOA

var cars = new Array("Saab", "Volvo", "BMW");

var x = cars.length;  // The length

var y = cars.sort();  // cars is sorted as well
```

Arrays are kinda objects.

Objects

JavaScript objects are written with curly brackets { }.

Object properties are written as name:value pairs, separated by commas.

Example

```
var person = {
     firstName: "Yournal",
     lastName: "Drunk",
     age: 50
};
```

Objects are references.

Exercise 1

Given the following program,

```
var cars = [ "Saab", "Volvo", "BMW"];
var newcars = cars;
var person = { firstName: "Lance", lastName: "Foo", age: 30};
var newperson = person;
person.firstName = "Prince";
cars[0] = "Honda"
```

Take ex1.html, and complete the TODO and answer the following:

- What is the value of newperson.firstName?
- What is the value of newcars[0]?

Dynamic Typing

JavaScript is dynamically typed.

That is, the same variable can be of different data types at different times.

The type depends on the context.

Example:

```
var x;          // x is undefined
x = 2;          // Now x is a Number
x = "Britney"; // Now x is a String
```

Be mindful of the types.

Dynamic Typing

JavaScript converts the type of variables implicitly.

The idea is to infer automatically what is the expected type and convert the type implicitly.

Example:

```
var xx = "100";
var yy = "10";
var zz = xx / yy;    // zz == 10
var z = xx - yy;     // z == 90, likewise for multiply *
var zzz = xx + yy;   // zzz will be "10010"
```

Type Conversion

It is your responsibility to make sure the types are correct, with facilities such as

- **Operator: typeof**
- Function: isNaN
- Functions for type conversion

Example

```
typeof 3.14       // Returns "number"
typeof (3)        // Returns "number"
typeof (3 + 4)    // Returns "number”
typeof ""         // Returns "string"
typeof "John"     // Returns "string”
typeof false      // Returns "boolean"

var x;
typeof x          // Returns "undefined"
```

Type Conversion

It is your responsibility to make sure the types are correct, with facilities such as

- Operator: typeOf
- **Function: isNaN**
- Functions for type conversion

NaN is a reserved word that represents a number that is not a legal number.

Example

```
var xx = 100 / "y";  // x will be NaN
isNaN(xx);  // returns true

//how about 50 / "10"?
//how about typeOf NaN?
```

Type Conversion

It is your responsibility to make sure the types are correct, with facilities such as

- Operator: typeOf
- Function: isNaN
- **Functions for type conversion**

Example

```
var x = 123;
x.toString();
(100+23).toString(); // returns “123”

var x = 9.656;
x.toFixed(2);  // returns 9.66

Number("10.33");      // returns 10.33
Number("10,33");      // returns NaN

parseInt("10.33");      // returns 10
parseInt("10 years");   // returns 10
parseInt("years 10");   // returns NaN
parseFloat("10.33");    // returns 10.33
parseFloat("10 20 30"); // returns 10
```

Exercise 2

Make use of ex2.html to evaluate the following. Check whether the results are expected.

```
5 + "34"
5 - "4"
10 % 5
5 % 10
"Java" + "Script"
" " + " "
" " + 0
true + true
true + false
false + true
false - true
3 - 4
"Bob" - "bill"
```

// Evaluate the below comparisons:

```
5 >= 1
4 <= 1
1 != 1
"A" > "B"
"B" < "C"
"a" > "A"
"b" < "A"
true != true
```

Operators

`+, -, *, **, /, %`	Addition, Subtraction, Multiplication, Exponentiation, Dvision, Modulus,
`++, --`	Increment and Decrement (by 1)
`+=, -=, *=, /=, %=, **=`	Example: `x += y  // x = x + y;` Example: `x **= y  // x = x**y`
`==, !=`	Check equality after type conversion
`===, !==`	Check if the type is the same and then if the value is the same
`<, >, >=, <=, !=`	Usual meanings
`? … : ...`	Example: `x > y ? x:y`
`&&, ||, !`	Logical and, or, and negate

= VS. == VS. ===

Examples

```
0 == false    // Result??

0 === false   // Result??

2 == "2"      // Result??

2 === "2"     // Result??

10 = 10       // Result??
```

Exercise 3

Given ex3.html, implement a simple calculator based on the requirements.

Conditional Branching

Syntax

```
if (condition) {

//...

} else { //optional

//...

} else { //optional

//...

}
```

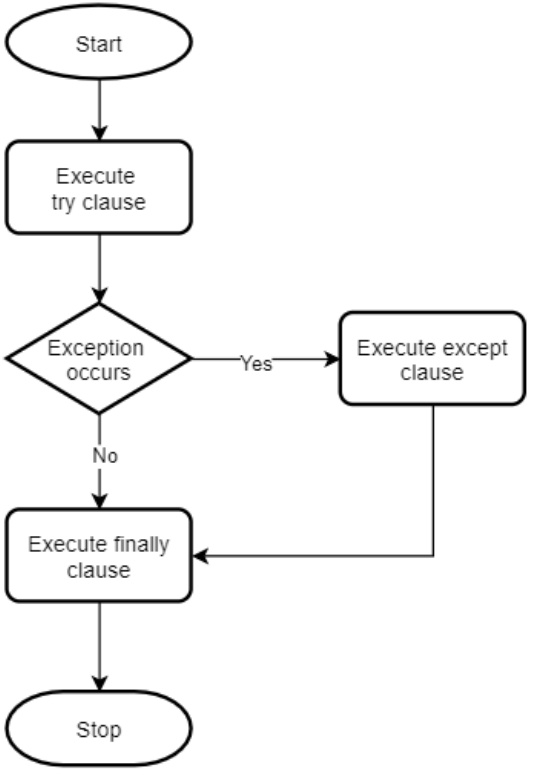
Examples

```
var age = prompt("What is your age?");
if( Number(age) < 18 ) {
   alert("Sorry, you are too young to drive this car");
}
else if( Number(age) > 18 ) {
   alert("Enjoy the ride!");
}
else if( Number(age) === 18 ) {
   alert("Congratulations on your first year of driving.");
}
```

Exception Handling

Syntax

```
try {
  // Block of code to try
}
catch(err) {
  // Block of code to handle errors
}
finally {
  /* Block of code to execute
     regardless of try/catch result */
}
```



Exception Handling

Syntax

```
try {
  ...
}
catch(err) {
  ...
}
finally {
  ...
}
```

Example

```
var json = openAndReadFile();

try {
  var user = JSON.parse(json); // error occurs
  alert( user.name ); // doesn't work
}
catch (e) {
  // ...the execution jumps here
  alert( "Our apologies. Jnvalid JSON." );
}
finally {
  closeFile();
}
```

Loops

for-loops, loops through a block of code for a number of times

Example:

```
for( i=0; I < 10; i++ )
{
   console.log(i);
}
```

for ... in loops, loops through the properties of an object

Example:

```
var person = {
     firstName: "Lance",
     lastName: "Kaypoh",
     school: "SCIS"
     age: 25
};

var prop;

for (prop in person) {
  console.log(person[prop]);
}
//result: Lance, Kaypoh, SCIS, 25
```

Loops

for … of loops, loop through the values of an ***iterable*** variable

Example:

```
var sharks = [ "great white", "tiger"];
for( shark of sharks ) {
   console.log(shark);
}

for( c of "sfdagdsg@da") {
   if (c == "@") {
      found = true;
   }
}
```

while–loops through a block of code until a specified condition becomes false

Example:

```
while (i < 10)
{
   text += "<br>The number is " + i;
   i++;
}
```

for … of … doesn't work for objects.
for … in … works for arrays.

Exercise 4

Complete ex4.html.

Given an array, loop through the array and then find the person called “The Chosen One”. Print his name and age in the console.

String Operations

Operation	Remark	Example
length	Returns the length of the string	`var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";` `txt.length //result: 26`

indexOf	Returns the index of the given string; -1 if it is non-existent	`//string indexing`
		`"Please locate where 'locate'`
		`occurs!".indexOf("locate"); //result: 7`

lastIndexOf	Returns the index of the last occurrence of a specified text in a string	`txt.lastIndexOf("EFG") //4`
		`txt.lastIndexOf("EFG", 5)`
		`//-1 (not exist); searching backwards from index 5`

slice	Extracts a part of a string and returns the extracted part in a new string	`var str = "Apple, Banana, Kiwi";`
		`var res = str.slice(7, 13); //result: Banana`
		`str.slice(1) // "pple, Banana, Kiwi"`

String Operations

Operation	Remark	Example
toLowerCase	...	`var txt2 = txt.toLowerCase();`
toUpperCase	...	`txt = txt2.toUpperCase();`
trim	Removes whitespace from both sides of a string	`var str = "  Hello World!  ";` `alert(str.trim());  // "Hello World!"`

substring	Extracts a part of a string and returns the extracted part in a new string	`var schools = "SOA, SCIS, SOL";`
		`var sis = schools.substr(5, 3); // SCIS`
		`schools.substr(5); // SCIS, SOL`

String Operations

Operation	Remark	Example
replace	Searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced; By default, the replace() method replaces only the first match	`var str1 = "IS216 is an awesome IS course!";` `var str2 = str1.replace("IS216", "CS216");`

split	Convert a string into an array	`var str = "a,b,c,d,e";  // String`
		`var arr = str.split(",");  // Split on commas`
		`console.log(arr);`
		`var str = "Hello";  // String`
		`var arr2 = str.split("");  // Split in characters`

Array Operations

Operation	**Remark**	**Example**
push	Adds a new element at the end	`var schools = ["SCIS", "SOA", "SOL"];` `schools.push("SOSS");`
pop	Removes the last element	`schools.pop(); // Removes the last element "SOSS"`
shift	Pops the first element	`var schools = ["SCIS", "SOA", "SOL"];` `var scis = schools.shift(); // scis = "SCIS"`
unshift	Adds a new element at index 0.	`// "SOSS" is added at idx 0. returns 3` `schools.unshift("SOSS");`

slice	Creates a slice of an array	`var schools = ["SOSS", "SCIS", "SOE", "SOL"];`
		`var newSchools = schools.slice(1,3); // -> SCIS,SOE`
		`var newSchools = schools.slice(2);  // -> SOE,SOL`

Looping Over Array Elements

for-loops can be used to loop through an array:

Try using for ... of loop

Examples

```
var schools, schoolList, len, i;
schools = ["SOSS", "SCIS", "SOE", "SOL"];
len = schools.length;

schoolList = "<ul>";
for( i = 0; i < len; i++ )
{
   schoolList += "<li>" + schools[i] + "</li>";
}
schoolList += "</ul>";
```

Exercise 5

Given ex5.html, complete the following task: check whether the the user-provided string is a palindrome or not.

***A palindrome is a string which reads the same backward as forward. For example, “Race Car” and “Anna” are palindromes. “Apple Pie” and “John” are not. Ignore spaces in deciding a palindrome.

Functions

A block of code designed to perform a particular task.

A function is executed when "something" invokes it.

A function has its own scope.

Example

```
function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}

var temp = "The temperature is " + toCelsius(75) + " Celsius";

function myFunction() {
  var school = "SCIS";
  // code here CAN use var school
}

// code here can NOT use var school
```

Functions

Javascript is “functional”.

Functions are objects.

Examples

```
function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
alert(typeof toCelsius); // function

alert(toCelsius instanceof Object); //true

var body = "return Math.PI * radius * radius";
var circle = new Function("radius", body);
alert(circle(5));  // 78.5398..
```

Functions

Javascript is “functional”.

Functions are objects.

Function objects can be passed around just like any object.

Examples:

```
// the following is a function which negates
// any given function

function negate (f) {
   return function (i) {
      return !f(i);
   };
}

var isNumber = negate(isNaN);
alert(isNumber(5));            // => true
alert(isNumber("A"));          // => false
alert(isNumber(NaN));          // => false
```

Functions

Having functions as first-class citizen has a lot of convenience, e.g., callback functions.

Examples

```
const showMessage = function() {
   console.log("This message is shown after 3 seconds");
}

setTimeout(showMessage, 3000);
// setTimeout is a function which calls a function after a
// given period of time (in milliseconds).

setTimeout(function() {
   console.log("This message is shown after 3 seconds");
}, 3000);
```

Functions and this

“this” keyword refers to the object it belongs to.

In a method, this refers to the owner object.

Alone, this refers to the global object.

In an event, this refers to the element that received the event.

Examples: *wk4example2.html*

```
function Book (type, author) {
   this.type = type;
   this.author = author;
   this.getDetails = function () {
      return this.type + " written by " + this.author;
   }
}

var book = new Book("Fiction", "Peter King");
alert(book.getDetails());
// result: "Fiction written by Peter King"

alert(book.type) //what is the result
```

“this”

“this” refers to the current containing object.

Example

```
function bike() {
   console.log(this.name);
}

var name = "Ninja";
var obj1 = { name: "Pulsar", bike: bike };
var obj2 = { name: "Gixxer", bike: bike };

bike();         // what is logged?
obj1.bike();    // what is logged?
obj2.bike();    // what is logged?
```

Prototype-based Object-Orientation

Objects in Javascript are a collection of properties/methods.

Properties are written as name:value pairs.

Methods are actions that can be performed on objects.

Example: *wk4example3.html*

```
var user = {
   name: "Mary",
   age: 26,
   hobby: ["swimming", "badminton"],
   isMarried: false,

   addHobby: function(new_hobby) {
      this.hobby.push(new_hobby);
   }
};
```

Be mindful of “this” in this example. Without it, there is an error.

Accessing Object Properties/Methods

You can access object properties and methods in two ways:

```
objectName.propertyName
objectName.method(parameters)
```

Or

```
objectName["propertyName"]
objectName["methodName"](parameters)
```

Example: *wk4example3.html*

```
console.log(user.name);

// access obj property like an array
console.log(user["name"]);
```

An array can be viewed as a special object of the following form

```
var array = {
  0: "first",
  1: "second",
  2: "third"
};
```

Modifying Object Properties/Methods

You can modify object properties or methods in two ways:

```
objectName.propertyName = new_value;
```

Or,

```
objectName["propertyName"]  = new_value;
```

You can add/delete object properties or methods.

Example

wk4example3.html

Exercise 6

forEach

forEach() method executes a provided function once for each array element.

Example

```
var array1 = [1, 2, 3];
array1.forEach(
   element => console.log(element+1)
);

// expected output: 2,3,4
```

Exercise

Given ex6.html, define a function which works in a way similar to forEach according to the provided requirements.

Take Away Message

Javascript is

- Dynamic typing
- Prototype-based object-oriented
- Functional