

JSON and API calls



What is our GOAL for this MODULE?

In this module we learned about the JSON data structure and usage of an API.

What did we ACHIEVE in the class TODAY?

- We learned about the JSON data structure which can be used to store and access data.
- We also learned the concept of an API and how an API call be made to the service and desired data extracted from it.
- We used an API call to change the background of the game depending on the time of the day at a specified place and also built the scoring system in the game.
- We learned to create an asynchronous function in Javascript.

Which CONCEPTS/ CODING BLOCKS did we cover today?

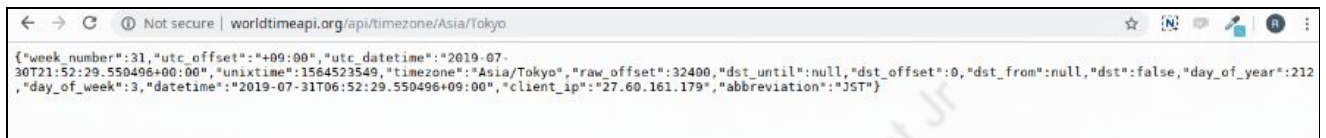
- JSON data structure.
- Concept of API

How did we DO the activities?

API stands for Application Programming Interface. Using API calls we ordered a web server to give us some information. Different Web Servers provided with different types of API calls and "Promise" of information.

We began with using a simple API called worldtimeapi. It gave the time of any specified location on the planet.

With this <http://worldtimeapi.org/api/timezone/Asia/Tokyo> you could change the continent and the city to different values: Asia/Kolkata or Europe/London etc.

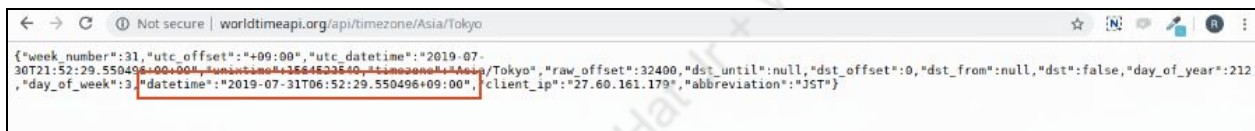


```

{
  "week_number": 31,
  "utc_offset": "+09:00",
  "utc_datetime": "2019-07-30T21:52:29.550496+00:00",
  "unixtime": 1564523549,
  "timezone": "Asia/Tokyo",
  "raw_offset": 32400,
  "dst_until": null,
  "dst_offset": 0,
  "dst_from": null,
  "dst": false,
  "day_of_year": 212,
  "day_of_week": 3,
  "datetime": "2019-07-31T06:52:29.550496+09:00",
  "client_ip": "27.60.161.179",
  "abbreviation": "JST"
}

```

The datetime told us the date and the time with very high precision.



```

{
  "week_number": 31,
  "utc_offset": "+09:00",
  "utc_datetime": "2019-07-30T21:52:29.550496+00:00",
  "unixtime": 1564523549,
  "timezone": "Asia/Tokyo",
  "raw_offset": 32400,
  "dst_until": null,
  "dst_offset": 0,
  "dst_from": null,
  "dst": false,
  "day_of_year": 212,
  "day_of_week": 3,
  "datetime": "2019-07-31T06:52:29.550496+09:00",
  "client_ip": "27.60.161.179",
  "abbreviation": "JST"
}

```

The data was stored in JSON data structure. JSON stands for Javascript Object Notation. A JSON structure can hold multiple values inside {}. Each value was indexed by a name. For example, the date and time value was indexed by the name "datetime".



```

1 class BaseClass {
2   constructor(x, y, width, height, angle) {
3     var options = {
4       'restitution': 0.8,
5       'friction': 1.0,
6       'density': 1.0
7     }
8     this.body = Bodies.rectangle(x, y, width, height, options);
9     this.width = width;
10    this.height = height;
11    this.image = loadImage("sprites/base.png");
12    World.add(world, this.body);
13  }
14  display() {
15    var angle = this.body.angle;
16    push();
17    translate(this.body.position.x, this.body.position.y);
18    angleMode(DEGREES);
19    imageMode(CENTER);
20    image(this.image, 0, 0, this.width, this.height);
21    pop();
22  }
23 }

```

To programmatically call an API. We used the time information we got from this API and made our game a little more dynamic. The angry birds game took place with a day background. If it was daytime, we kept a daytime background. If it was night time, we kept a nighttime background. To do this, we wrote a function called `getTime()` and got the time of the day by making an API call.

```
76   bird.display();
77   platform.display();
78   //log6.display();
79   slingshot.display();
80 }
81
82 function mouseDragged(){
83   if (gameState!=="launched"){
84     Matter.Body.setPosition(bird.body, {x: mouseX , y: mouseY});
85   }
86 }
87
88
89 function mouseReleased(){
90   slingshot.fly();
91   gameState = "launched";
92 }
93
94 function keyPressed(){
95   if(keyCode === 32){
96     // slingshot.attach(bird.body);
97   }
98 }
99
100 function getTime(){
101 }
102 }
```

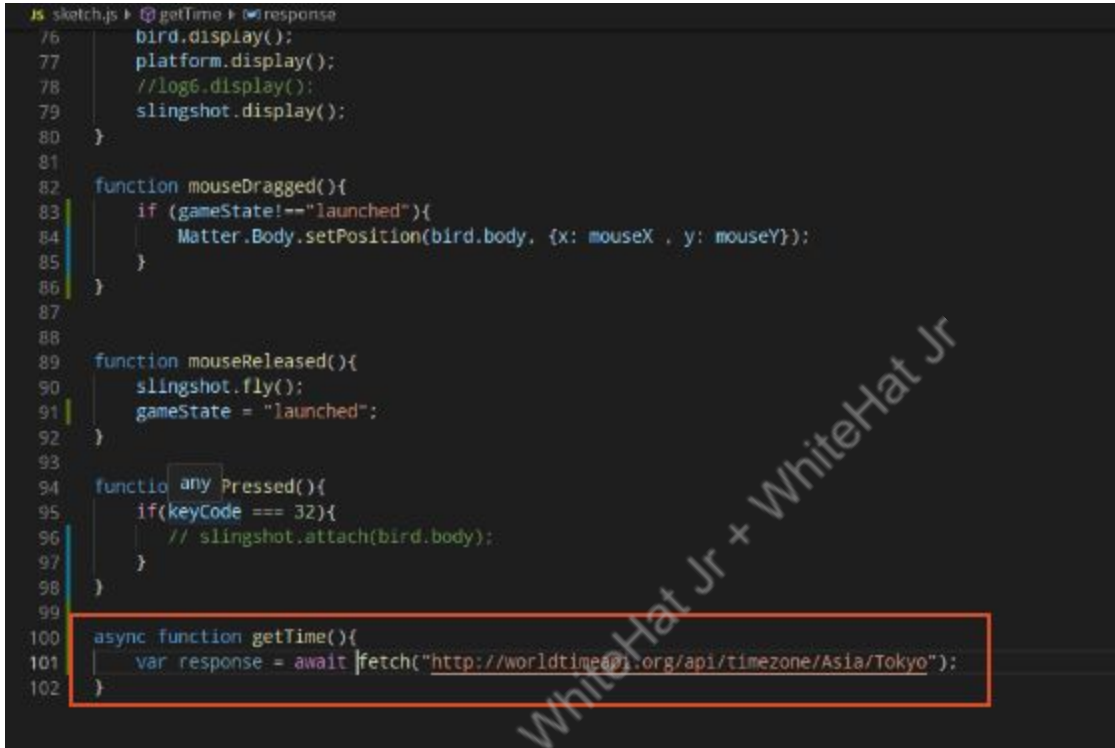
Next, we called an API using `fetch()` and collected the response in a variable.

```
76   bird.display();
77   platform.display();
78   //log6.display();
79   slingshot.display();
80 }
81
82 function mouseDragged(){
83   if (gameState!=="launched"){
84     Matter.Body.setPosition(bird.body, {x: mouseX , y: mouseY});
85   }
86 }
87
88
89 function mouseReleased(){
90   slingshot.fly();
91   gameState = "launched";
92 }
93
94 function keyPressed(){
95   if(keyCode === 32){
96     // slingshot.attach(bird.body);
97   }
98 }
99
100 function getTime(){
101   var response = fetch("http://worldtimeapi.org/api/timezone/Asia/Tokyo");
102 }
```

Note: An API call involved some network requests and took a little time.

However, Javascript executed this synchronously, which meant that it executed one line after the other. It did not wait for the API call to be completed before moving to the next line. However, we wanted it to wait for the API call to be completed. We did this by adding `await` before `fetch()`. It told the computer to wait for the API call to be completed before moving to the next lines.

Thus, we wrote an asynchronous function - a function that waits for some lines to be completed before jumping to the next line. We must instruct the computer that `getTime()` is an async function so that it knows how to function.



```
JS sketch.js 100 getTime 1 response
76   bird.display();
77   platform.display();
78   //log6.display();
79   slingshot.display();
80 }
81
82 function mouseDragged(){
83   if (gameState!=="launched"){
84     Matter.Body.setPosition(bird.body, {x: mouseX , y: mouseY});
85   }
86 }
87
88
89 function mouseReleased(){
90   slingshot.fly();
91   gameState = "launched";
92 }
93
94 function any Pressed(){
95   if(keyCode === 32){
96     // slingshot.attach(bird.body);
97   }
98 }
99
100 async function getTime(){
101   var response = await fetch("http://worldtimeapi.org/api/timezone/Asia/Tokyo");
102 }
```

The response contained a promise of the data. We needed to extract the JSON response out of it. We used the `JSON()` function to do this. We also logged this on our console to see what it contained.

```

js sketch.js ▶ @getTime
76   bird.display();
77   platform.display();
78   //log6.display();
79   slingshot.display();
80 }
81
82 function mouseDragged(){
83   if (gameState!=="launched"){
84     Matter.Body.setPosition(bird.body, {x: mouseX , y: mouseY});
85   }
86 }
87
88
89 function mouseReleased(){
90   slingshot.fly();
91   gameState = "launched";
92 }
93
94 function keyPressed(){
95   if(keyCode === 32){
96     // slingshot.attach(bird.body);
97   }
98 }
99
100 async function getTime(){
101   var response = await fetch("http://worldtimeapi.org/api/timezone/Asia/Tokyo");
102   var responseJSON = await response.json();
103   console.log(responseJSON);
104 }

```

```

datetime: "2019-07-31T08:22:46.245956+09:00"
day_of_week: 3
day_of_year: 212
dst: false
dst_from: null
dst_offset: 0
dst_until: null
raw_offset: 32400
timezone: "Asia/Tokyo"
unixtime: 1564528966
utc_datetime: "2019-07-30T23:22:46.245956+00:00"
utc_offset: "+09:00"
week_number: 31
__proto__: Object

```

Extracting the datetime from the json data we logged responseJSON.datetime.
 This is how values in JSON data structure were accessed using their index.

```

Group similar
⚠ The AudioContext was not allowed to start. It must be resumed (or created) after a user gesture on the page. https://goo.gl/7K7WLu p5.sound.min.js:24
⚠ The AudioContext was not allowed to start. It must be resumed (or created) after a user gesture on the page. https://goo.gl/7K7WLu p5.sound.min.js:25
2019-07-31T05:43:25.490291+09:00 sketch.js:103

```

The datetime contained both the date and time with great precision in string format. However, if one wanted only the hour information, strings can be sliced by the number of characters from the start of the string to the end where we want to slice it. The hour information is present between 11th to 13th character.

```
var datetime = responseJSON.datetime  
var hour = daytime.splice(11,13)
```



To fetch and extract data over API call:

We defined a variable bg to store the background.

Wrote the async function getBackgroundImg() which fetched the datetime information and extracted the hour information. It then chose a background image depending on the hour of the day.

```
async function getBackgroundImg(){  
  var response = await fetch("http://worldtimeapi.org/api/timezone/Asia/Kolkata");  
  var responseJSON = await response.json();  
  
  var datetime = responseJSON.datetime;  
  var hour = datetime.slice(11,13);  
  
  if(hour>=06 && hour<=19){  
    bg = "sprites/bg1.png";  
  }  
  else{  
    bg = "sprites/bg2.jpg";  
  }  
  
  backgroundImg = loadImage(bg);  
  console.log(backgroundImg);  
}
```


We called the function `getBackgroundImg()` function inside `preload()`.

```

1  const Engine = Matter.Engine;
2  const World= Matter.World;
3  const Bodies = Matter.Bodies;
4  const Constraint = Matter.Constraint;
5
6  var engine, world;
7  var box1, pig1,pig3;
8  var backgroundImg,platform;
9  var bird, slingshot;
10
11 var gameState = "onSling";
12 var bg = "sprites/bg1.png";
13 var score = 0;
14
15 function preload() {
16   getBackgroundImg();
17 }
18
19 function setup(){
20   var canvas = createCanvas(1200,400);
21   engine = Engine.create();
22   world = engine.world;
23
24
25   ground = new Ground(600,height,1200,20);
26   platform = new Ground(150, 305, 300, 170);
27
28   box1 = new Box(700,320,70,70);
29   box2 = new Box(920,320,70,70);
30   pig1 = new Pig(810, 350);
  
```

We coded so that the `background()` function was executed only if the background image was defined.

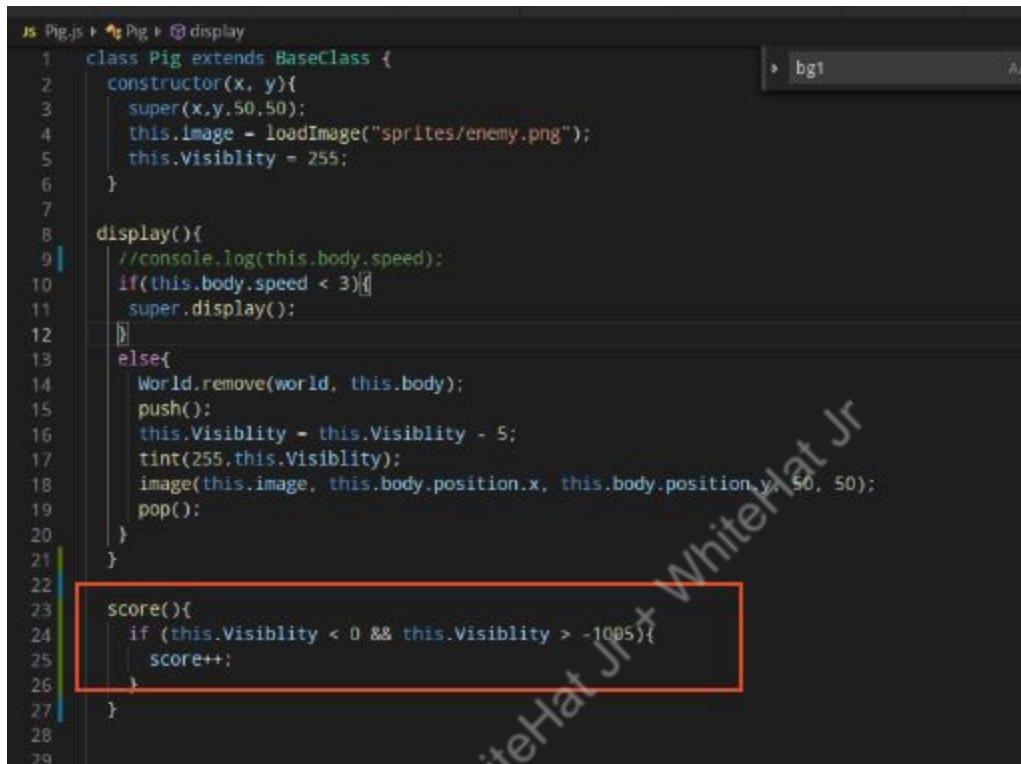
```

41 log5 = new Log(870,120,150, -PI/7);
42
43 bird = new Bird(200,50);
44
45 //log6 = new Log(230,180,80, PI/2);
46 slingshot = new SlingShot(bird.body,{x:200, y:50});
47 }
48
49 function draw(){
50   if(backgroundImg)
51     background(backgroundImg);
52
  
```


We created a score variable and displayed it.

```
js sketch.js ▶ preload
41   log5 = new Log(870,120,150, -PI/7);
42
43   bird = new Bird(200,50);
44
45   //log6 = new Log(230,180,80, PI/2);
46   slingshot = new SlingShot(bird.body,{x:200, y:50});
47 }
48
49 function draw(){
50   if(backgroundImg)
51     background(backgroundImg);
52
53   noStroke();
54   textSize(35)
55   fill("white")
56   text("Score " + score, width-300, 50)
57
58   Engine.update(engine);
59   //strokeWeight(4);
60   box1.display();
61   box2.display();
62   ground.display();
63   pig1.display();
64   pig1.score();
65   log1.display();
66
67   box3.display();
68   box4.display();
69   pig3.display();
```

We wrote a function score() inside pig class to increase the score as its visibility reduced.



```
1 class Pig extends BaseClass {
2   constructor(x, y){
3     super(x,y,50,50);
4     this.image = loadImage("sprites/enemy.png");
5     this.Visibility = 255;
6   }
7
8   display(){
9     //console.log(this.body.speed);
10    if(this.body.speed < 3){
11      super.display();
12    }
13    else{
14      World.remove(world, this.body);
15      push();
16      this.Visibility = this.Visibility - 5;
17      tint(255,this.Visibility);
18      image(this.image, this.body.position.x, this.body.position.y, 50, 50);
19      pop();
20    }
21  }
22
23  score(){
24    if (this.Visibility < 0 && this.Visibility > -1005){
25      score++;
26    }
27  }
28
29 }
```

To call the score function for each of the objects created in the game:

```
JS sketch.js x JS BaseClass.js base.png JS Bird.js JS Box.js JS Ground.js
JS sketch.js preload
55 fill("white");
56 text("Score " + score, width-300, 50)
57
58 Engine.update(engine);
59 //strokeWeight(4);
60 box1.display();
61 box2.display();
62 ground.display();
63 pig1.display();
64 pig1.score();
65 log1.display();
66
67 box3.display();
68 box4.display();
69 pig3.display();
70 pig3.score();
71 log3.display();
72
73 box5.display();
74 log4.display();
75 log5.display();
76
77 bird.display();
78 platform.display();
79 //log6.display();
80 slingshot.display();
81 }
82
83 function mouseDragged(){
84   if (gameState!="launched"){
85     Matter.Body.setPosition(bird.body, {x: mouseX , y: mouseY});
86   }
87 }
```



What's NEXT?

In the next class, you will be learning about debugging tips and tricks.

EXTEND YOUR KNOWLEDGE:

You can explore more about JSON data structure

https://www.w3schools.com/js/js_json.asp

WhiteHat Jr + WhiteHat Jr + WhiteHat Jr