

# How Events work

Bird's Eye View

How do programs  
run?  
TOP to BOTTOM

- Each step goes in order.
- Happens faster than you can see it!

```
JS script.js > ...
```

```
1 1.console.log("hello world from script js");
```

```
2
```

```
3 2.setProperty("waterAppHeader", "color", "red");
```

```
4
```

```
5 3.setProperty("waterAppHeader", "background-color", "blue");
```

```
6
```

```
7
```

# Pop Quiz:

- \* What will you see on the screen?
- \* What will this print?

```
2
3  var myFavColor = "red";
4
5  console.log("Favorite color is: " + myFavColor);
6  setProperty("waterAppHeader", "color", myFavColor);
7
8
9  myFavColor = "blue";
10 console.log("Favorite color is: " + myFavColor);
11 setProperty("waterAppHeader", "color", myFavColor);
12
13
14 myFavColor = "yellow";
15 console.log("Favorite color is: " + myFavColor);
16 setProperty("waterAppHeader", "color", myFavColor);
17
```


Functions change up the order.  
What color will the background be????????

```
1  var myFavColor = "red";
2  var mySecondFavColor = "blue";
3
4  setProperty("myContainer", "border-width", "10px");
5  setProperty("myContainer", "border-style", "solid");
6
7  function setBackground() {
8      console.log("hello from setBackground");
9      setProperty("myContainer", "background-color", myFavColor);
10     setProperty("myContainer", "border-color", mySecondFavColor);
11 }
12
13 myFavColor = "yellow";
14 mySecondFavColor = "green";
15
16 setBackground();
17
```

Functions change up the order.  
What color will the background be????????

```
1  var myFavColor = "red";
2  var mySecondFavColor = "blue";
3  1.
4  setProperty("myContainer", "border-width", "10px");
5  setProperty("myContainer", "border-style", "solid");
6
7  function setBackground() {
8      console.log("hello from setBackground");
9      3.
10     setProperty("myContainer", "background-color", myFavColor);
11     setProperty("myContainer", "border-color", mySecondFavColor);
12 }
13
14 2.
15 myFavColor = "yellow";
16 mySecondFavColor = "green";
17
18 setBackground();
```

# Events can fire off a function whenever you want using a CALLBACK function.

```
JS script.js >  onEvent("esButton", "click") callback
1  var myFavColor = "red";
2  var mySecondFavColor = "blue";
3
4  function setBackground() {
5      console.log("hello from setBackground");
6      setProperty("myContainer", "background-color", myFavColor);
7      setProperty("myContainer", "border-color", mySecondFavColor);
8  }
9
10 onEvent("esButton", "click", function () {
11     console.log("clicked");
12     setText("waterAppHeader", "Aqua");
13     setBackground();
14 });
15
16
17
18
```

# Pop Quiz: What color will the background be AFTER the esButton is clicked?

```
JS script.js > onEvent("esButton", "click") callback
1  var myFavColor = "red";
2  var mySecondFavColor = "blue";
3
4  function setBackground() {
5      console.log("hello from setBackground");
6      setProperty("myContainer", "background-color", myFavColor);
7      setProperty("myContainer", "border-color", mySecondFavColor);
8  }
9
10 onEvent("esButton", "click", function () {
11     console.log("clicked");
12     setText("waterAppHeader", "Aqua");
13     setBackground();
14 });
15
16
17 myFavColor = "aqua";
18 mySecondFavColor = "gold";
19
20
21
```

# Can you follow the thread?

```
JS script.js > onEvent("esButton", "click") callback
1  var myFavColor = "red";
2  var mySecondFavColor = "blue";
3
4  function setBackground() {
5      console.log("hello from setBackground");
6      setProperty("myContainer", "background-color", myFavColor);
7      setProperty("myContainer", "border-color", mySecondFavColor);
8  }
9
10 onEvent("esButton", "click", function () {
11     console.log("clicked");
12     setText("waterAppHeader", "Aqua");
13     setBackground();
14 });
15
16
17 myFavColor = "aqua";
18 mySecondFavColor = "gold";
19
20
21
```



# What about API calls? Where does the thread go?

```
JS script.js > fetchNewOrleansWeather
1
2   var temperature = 105;
3   function fetchNewOrleansWeather() {
4       console.log("hello from fetchNewOrleansWeather");
5       const requestOptions = {
6           method: "GET",
7           redirect: "follow"
8       };
9
10      fetch("https://api.open-meteo.com/v1/forecast?latitude=29.95653&longitude=-90.07374&current=temperature_2m,")
11          .then((response) => response.json())
12          .then(function (result) {
13              temperature = result.current.temperature_2m;
14              console.log(temperature);
15              setText("temp", temperature);
16          })
17          .catch((error) => console.error(error));
18
19      console.log("goodbye from fetchNewOrleansWeather");
20
21  }
22
23  // This will kick off the network call.
24  fetchNewOrleansWeather();
25
26
```

# What about API calls? Where does the thread go?

Put the letters in order from first to last, by time.

```
JS script.js > fetchNewOrleansWeather
1
2 A. var temperature = 105;
3 function fetchNewOrleansWeather() {
4     console.log("hello from fetchNewOrleansWeather");
5     const requestOptions = {
6 B.       method: "GET",
7         redirect: "follow"
8     };
9
10    fetch("https://api.open-meteo.com/v1/forecast?latitude=29.95653&longitude=-90.07374&current=temperature_2m,is
11        .then((response) => response.json())
12        .then(function (result) {
13            temperature = result.current.temperature_2m;
14 C.        console.log(temperature);
15            setText("temp", temperature);
16        })
17        .catch((error) => console.error(error));
18
19 D.    console.log("goodbye from fetchNewOrleansWeather");
20
21    }
22
23 // This will kick off the network call.
24 E. fetchNewOrleansWeather();
25
```

# What about API calls? Where does the thread go?

```
JS script.js > fetchNewOrleansWeather
1
2. var temperature = 105;
3 function fetchNewOrleansWeather() {
4     console.log("hello from fetchNewOrleansWeather");
5     3. const requestOptions = {
6         method: "GET",
7         redirect: "follow"
8     };
9
10    fetch("https://api.open-meteo.com/v1/forecast?latitude=29.95653&longitude=-90.07374&current=temperature_2m,")
11        .then((response) => response.json())
12        .then(function (result) {
13            5. temperature = result.current.temperature_2m;
14            console.log(temperature);
15            setText("temp", temperature);
16        })
17        .catch((error) => console.error(error));
18
19    4. console.log("goodbye from fetchNewOrleansWeather");
20
21
22
23 // This will kick off the network call.
24 2. fetchNewOrleansWeather();
25
26
```

# Pop Quiz: What will this print?

```
JS script.js > fetchNewOrleansWeather > then() callback
1
2 var temperature = 105;
3 function fetchNewOrleansWeather() {
4     console.log("hello from fetchNewOrleansWeather");
5     console.log(temperature);
6
7     const requestOptions = {
8         method: "GET",
9         redirect: "follow"
10    };
11
12    fetch("https://api.open-meteo.com/v1/forecast?latitude=29.95653&longitude=-90.07374&current=temperature_2m,is
13        .then((response) => response.json())
14        .then(function (result) {
15            temperature = result.current.temperature_2m;
16            console.log(temperature);
17            setText("temp", temperature);
18        })
19        .catch((error) => console.error(error));
20
21    console.log(temperature);
22    console.log("goodbye from fetchNewOrleansWeather");
23
24 }
25
26 // This will kick off the network call.
27 console.log(temperature);
28 fetchNewOrleansWeather();
29 console.log(temperature);
30
31
```

# Overview

- Programs run in order
- Programs define functions and assign variables (in order).
- Programs have functions that change the flow.
- Events add a way to run a function when someone clicks (or other events). This is called “listening for an event”
- Fetching an API takes time.
- When the API returns the value (ex: weather), that is another kind of event.
- The `` .then(response => { }) `` part of the program is listening for the data to return.

# Cheat Sheet for Hugging Face generated code

```
JS fireworks.js > query > response > headers
1  async function query(data) {
2    const response = await fetch(
3      "https://router.huggingface.co/v1/chat/completions",
4      {
5        headers: {
6          Authorization: `Bearer ${process.env.HF_TOKEN}`,
7          "Content-Type": "application/json",
8        },
9        method: "POST",
10       body: JSON.stringify(data),
11     }
12   );
13   const result = await response.json();
14   return result;
15 }
16
17 query({
18   messages: [
19     {
20       role: "user",
21       content: "What is the capital of France?",
22     },
23   ],
24   model: "meta-llama/Llama-3.1-70B-Instruct:fireworks-ai",
25 }).then((response) => {
26   console.log(JSON.stringify(response));
27 });
28
```

```
JS fireworks.js > query > response > headers
1  var botReply = "";
2  async function query(data) {
3    const response = await fetch(
4      "https://router.huggingface.co/v1/chat/completions",
5      {
6        headers: {
7          Authorization: `Bearer ${HF_TOKEN}`,
8          "Content-Type": "application/json",
9        },
10       method: "POST",
11       body: JSON.stringify(data),
12     }
13   );
14   const result = await response.json();
15   return result;
16 }
17
18 query({
19   messages: [
20     {
21       role: "user",
22       content: "What is the capital of Mississippi?",
23     },
24   ],
25   model: "meta-llama/Llama-3.1-70B-Instruct:fireworks-ai",
26 }).then((response) => {
27   console.log(JSON.stringify(response));
28   botReply = response.choices[0].message.content;
29   console.log(botReply);
30   setText("output-display", botReply);
31 });
```

# Cheat Sheet for Postman API generated code

```
JS script.js > ...
1  const requestOptions = {
2    method: "GET",
3    redirect: "follow"
4  };
5
6  fetch("https://api.open-meteo.com/v1/forecast?latitude=52.52&lon=13.41")
7    .then((response) => response.text())
8    .then((result) => console.log(result))
9    .catch((error) => console.error(error));
10
11
12
13
```

```
JS script.js > ...
1  const requestOptions = {
2    method: "GET",
3    redirect: "follow"
4  };
5
6  fetch("https://api.open-meteo.com/v1/forecast?latitude=52.52&lon=13.41")
7    .then((response) => response.json())
8    .then((result) => console.log(result))
9    .catch((error) => console.error(error));
10
11
12
13
```

```
JS script.js > ...
Added lines
1  const requestOptions = {
2    method: "GET",
3    redirect: "follow"
4  };
5
6  fetch("https://api.open-meteo.com/v1/forecast?latitude=52.52&lon=13.41")
7    .then((response) => response.json())
8    .then((result) => console.log(result))
9    .catch((error) => console.error(error));
10
11
12
13
```

```
JS script.js > ...
1  const requestOptions = {
2    method: "GET",
3    redirect: "follow"
4  };
5
6  var temperature = 0;
7  fetch("https://api.open-meteo.com/v1/forecast?latitude=52.52&lon=13.41")
8    .then((response) => response.json())
9    .then(function (result) {
10      console.log(result);
11      temperature = result.current.temperature_2m;
12      console.log(temperature);
13    })
14    .catch((error) => console.error(error));
15
16
```