---

title: Web SDK

subtitle: Integrate Vapi into your web application.

slug: sdk/web

---

The Vapi Web SDK provides web developers a simple API for interacting with the realtime call functionality of Vapi.

### Installation

Install the package:

```bash

yarn add @vapi-ai/web

```

or w/ npm:

```bash

npm install @vapi-ai/web

```

### Importing

Import the package:

```javascript

import Vapi from "@vapi-ai/web";

```

Then, create a new instance of the Vapi class, passing one of the following as a parameter to the constructor:

- your \*\*Public Key\*\*

- a generated \*\*JWT\*\*

```javascript

const vapi = new Vapi("your-public-key-or-jwt");

```

You can find your public key in the [Vapi Dashboard](https://dashboard.vapi.ai/account).

You can generate a JWT on the backend following [JWT Authentication](/customization/jwt-authentication) instructions.

---

## Usage

### `.start()`

You can start a web call by calling the `.start()` function. The `start` function can either accept:

1. \*\*a string\*\*, representing an assistant ID

2. \*\*an object\*\*, representing a set of assistant configs (see [Create Assistant](/api-reference/assistants/create-assistant))

The `start` function returns a promise that resolves to a call object. For example:

```javascript

const call = await vapi.start(assistantId);

// { "id": "bd2184a1-bdea-4d4f-9503-b09ca8b185e6", "orgId": "6da6841c-0fca-4604-8941-3d5d65f43a17", "createdAt": "2024-11-13T19:20:24.606Z", "updatedAt": "2024-11-13T19:20:24.606Z", "type": "webCall", ... }

```

#### Passing an Assistant ID

If you already have an assistant that you created (either via [the Dashboard](/quickstart/dashboard) or [the API](/api-reference/assistants/create-assistant)), you can start the call with the assistant's ID:

```javascript

vapi.start("79f3XXXX-XXXX-XXXX-XXXX-XXXXXXXXce48");

```

#### Passing Assistant Configuration Inline

You can also specify configuration for your assistant inline.

This will not create a [persistent assistant](/assistants/persistent-assistants) that is saved to your account, rather it will create an ephemeral assistant only used for this call specifically.

You can pass the assistant's configuration in an object (see [Create Assistant](/api-reference/assistants/create-assistant) for a list of acceptable fields):

```javascript

vapi.start({

transcriber: {

provider: "deepgram",

model: "nova-2",

language: "en-US",

},

model: {

provider: "openai",

model: "gpt-3.5-turbo",

messages: [

{

role: "system",

content: "You are a helpful assistant.",

},

],

},

voice: {

provider: "playht",

voiceId: "jennifer",

},

name: "My Inline Assistant",

...

});

```

#### Overriding Assistant Configurations

To override assistant settings or set template variables, you can pass `assistantOverrides` as the second argument.

For example, if the first message is "Hello `{{name}}`", set `assistantOverrides` to the following to replace `{{name}}` with `John`:

```javascript

const assistantOverrides = {

transcriber: {

provider: "deepgram",

model: "nova-2",

language: "en-US",

},

recordingEnabled: false,

variableValues: {

name: "Alice",

},

};

vapi.start("79f3XXXX-XXXX-XXXX-XXXX-XXXXXXXXce48", assistantOverrides);

```

### `.send()`

During the call, you can send intermediate messages to the assistant (like [background messages](/assistants/background-messages)).

- `type` will always be `"add-message"`

- the `message` field will have 2 items, `role` and `content`.

```javascript

vapi.send({

type: "add-message",

message: {

role: "system",

content: "The user has pressed the button, say peanuts",

},

});

```

<Info>

Possible values for role are `system`, `user`, `assistant`, `tool` or

`function`.

</Info>

### `.stop()`

You can stop the call session by calling the `stop` method:

```javascript

vapi.stop();

```

This will stop the recording and close the connection.

### `.isMuted()`

Check if the user's microphone is muted:

```javascript

vapi.isMuted();

```

### `.setMuted(muted: boolean)`

You can mute & unmute the user's microphone with `setMuted`:

```javascript

vapi.isMuted(); // false

vapi.setMuted(true);

vapi.isMuted(); // true

```

### `say(message: string, endCallAfterSpoken?: boolean)`

The `say` method can be used to invoke speech and gracefully terminate the call if needed

```javascript

vapi.say("Our time's up, goodbye!", true)

```

## Events

You can listen on the `vapi` instance for events. These events allow you to react to changes in the state of the call or user speech.

#### `speech-start`

Occurs when your AI assistant has started speaking.

```javascript

vapi.on("speech-start", () => {

console.log("Assistant speech has started.");

});

```

#### `speech-end`

Occurs when your AI assistant has finished speaking.

```javascript

vapi.on("speech-end", () => {

console.log("Assistant speech has ended.");

});

```

#### `call-start`

Occurs when the call has connected & begins.

```javascript

vapi.on("call-start", () => {

console.log("Call has started.");

});

```

#### `call-end`

Occurs when the call has disconnected & ended.

```javascript

vapi.on("call-end", () => {

console.log("Call has ended.");

});

```

#### `volume-level`

Realtime volume level updates for the assistant. A floating-point number between `0` & `1`.

```javascript

vapi.on("volume-level", (volume) => {

console.log(`Assistant volume level: ${volume}`);

});

```

#### `message`

Various assistant messages can be sent back to the client during the call. These are the same messages that your [server](/server-url) would receive.

At [assistant creation time](/api-reference/assistants/create-assistant), you can specify on the `clientMessages` field the set of messages you'd like the assistant to send back to the client.

Those messages will come back via the `message` event:

```javascript

// Various assistant messages can come back (like function calls, transcripts, etc)

vapi.on("message", (message) => {

console.log(message);

});

```

#### `error`

Handle errors that occur during the call.

```javascript

vapi.on("error", (e) => {

console.error(e);

});

```

---

## Resources

<CardGroup cols={2}>

<Card

title="NPM"

icon="fa-brands fa-npm"

iconType="solid"

href="https://www.npmjs.com/package/@vapi-ai/web"

>

View the package on NPM.

</Card>

<Card

title="GitHub"

icon="fa-brands fa-github"

iconType="solid"

href="https://github.com/VapiAI/web"

>

View the package on GitHub.

</Card>

</CardGroup>

<CardGroup cols={1}>

<Card

title="Try Our Quickstart"

icon="bolt"

iconType="solid"

href="/quickstart/web"

>

Get up and running quickly with the Web SDK.

</Card>

</CardGroup>