# Tic-tac-toe API

Write and deploy a stateless API that correctly answers the question, “What move will result in the best outcome for O on this tic-tac-toe board?” (You may use any language.)

Tic-tac-toe can also be called naughts and crosses. Instructions for how to play the game are here if you’ve never played before: <http://www.exploratorium.edu/brain_explorer/tictactoe.html>

**Note:** We’d like to preserve our ability to reuse this project with other candidates without bias. Don’t share this spec or your code (if you want to use Github, please keep your repo private). Thanks!

## Specification

“The best outcome” means a win, if you can force one. Otherwise, play for a tie.

**Request Handling and Validation**

* Your server will be provided the current board in a GET request, using the 'board' parameter in the query string.
* If the board string doesn't represent a valid tic-tac-toe board, or it’s not plausibly o’s turn, your server should return an HTTP response code 400 (Bad Request)
* Your server always plays as o.
* Either player can go first.
* If the board is a valid tic-tac-toe board and it is plausibly o's turn, your server should return a string representation of the same board with one ‘o’ added.

**Strategy**

Please implement the optimal play algorithm outlined on Wikipedia, which we’ve pasted here (from <https://en.wikipedia.org/wiki/Tic-tac-toe#Strategy>):

1. Win: If the player has two in a row, they can place a third to get three in a row.
2. Block: If the opponent has two in a row, the player must play the third themselves to block the opponent.
3. Fork: Create an opportunity where the player has two threats to win (two non-blocked lines of 2).
4. Blocking an opponent's fork: If there is only one possible fork for the opponent, the player should block it. Otherwise, the player should block any forks in any way that simultaneously allows them to create two in a row. Otherwise, the player should create a two in a row to force the opponent into defending, as long as it doesn't result in them creating a fork. For example, if "X" has two opposite corners and "O" has the center, "O" must not play a corner in order to win. (Playing a corner in this scenario creates a fork for "X" to win.)
5. Center: A player marks the center. (If it is the first move of the game, playing on a corner gives the second player more opportunities to make a mistake and may therefore be the better choice; however, it makes no difference between perfect players.)
6. Opposite corner: If the opponent is in the corner, the player plays the opposite corner.
7. Empty corner: The player plays in a corner square.
8. Empty side: The player plays in a middle square on any of the 4 sides.

**Board representation**

* The board is encoded as a string of nine characters where each character is either 'o' (letter o), 'x’, or a space. The nine characters are the tic-tac-toe board read left to right, top to bottom -- for example:

x|o|

-+-+-

o| |

-+-+-

|x|

would be encoded with the string "xo o x ", and an empty board would be a string of nine spaces.

**Example**

If I run

curl YOUR\_URL?board=+xxo++o++

I should get the exact string oxxo o (that’s o-x-x-o-space-space-o-space-space) as the **entire** contents of the HTTP response body. If your api returns anything else, our unit tests will fail when run on your code.

**Deploying**

You can deploy anywhere on the public internet that's convenient for you. We want to be able to test your code by making HTTP requests to your API, so whatever way makes most sense for you to make that possible works for us. If you're not sure where to deploy here are a couple of good choices:

* Heroku is a solid choice that should get you up and running quickly and for free, and has plenty of documentation on how to get started (<https://devcenter.heroku.com/start>), and sample projects which you can clone and modify (feel free to use these!) If you haven't used Heroku before, we recommend creating a Heroku account and reading the getting-started instructions before you start the timed portion of the challenge.
* You can run the code locally on the computer you developed it on, and set up an ngrok tunnel (<https://ngrok.com/product>). You will run your code locally, and then run the ngrok program on your computer, and configure ngrok to forward requests to your public facing URL to your computer. This only works if your computer will be plugged in and running before our conversation so we can test!
* The deployment itself counts towards the two and a half hours, but it's okay to practice beforehand if you're worried about spending time fighting with the tools.

**Return to us**

* A copy of your code in zip or tar (tar cf code.tgz \* should do it, or git archive master -o code.tgz). Attach this to the email or upload it somewhere accessible by a private link and send through the greenhouse link
* The url at which we can test it
* If the project is successful, we'll schedule a call for ~30 minutes for a quick conversation about your code and a chance for you to ask questions about Wave engineering.

**What we are looking for**

* You can use whatever libraries and tools you like, except please don't use a library that actually represents a Tic-Tac-Toe board or plays Tic-Tac-Toe -- that part is for you to design and build. Similarly, feel free to use google for general programming / library usage questions, but don't use code from the internet which solves the game.
* How we will score your submission (most important first):
* Your api implements the specified interface
* Your api rejects invalid boards and accepts valid boards
* Your code is readable, easy to understand and short
* You successfully implement the strategy above

**Advice**

* Use tools that you’re comfortable with, as they’ll give you the best chance to get the work done.
* Do the deployment work first, since we need to hit your API to be able to test it.