

# Building a Block Protocol Block

Before we get started...

- **Get** these slides here: <https://blockprotocol.org/workshop/slides.pdf>
- **Get** node & npm: <https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>
- **Sign up** for an account on <https://blockprotocol.org> (to publish your block)
- **Join** the **#workshop-blockprotocol** channel in the Strange Loop Slack
- **Check** that the template works for you by doing the following in a terminal:
  - **npx create-block-app@latest your-block-name**
  - **cd your-block-name**
  - **npm install && npm run dev** or **yarn install && yarn dev**
  - visit <http://localhost:63212> – you should see 'Hello, World!' UI

# You should see something like this

The screenshot displays the Dock v0.0.26 interface. At the top, there is a navigation bar with the Dock logo and version number, a 'Blocks /' breadcrumb, and utility buttons for 'Docs', 'Dark Mode' (OFF), and 'Preview Mode' (ON). The main content area shows a block with the text 'Hello, World!' and a description: 'The entityId of this block is my-entity-1. Use it to update its data, e.g. by calling updateEntity.' Below this is a blue 'Update Name' button. At the bottom, there is a 'Properties' tab selected, showing 'Entity Properties' with a 'Read-only mode' toggle. The JSON schema for the block is displayed in a code editor, showing the structure of the block's data. To the right, there is a 'Block Schema' section with an 'upload schema' button. A 'React Block' button is visible in the bottom right corner of the interface.

Dock v0.0.26 Blocks / Docs Dark Mode OFF Preview Mode ON

Hello, World!

The entityId of this block is my-entity-1. Use it to update its data, e.g. by calling updateEntity.

Update Name

Properties Datastore Logs React Block

Entity Properties Read-only mode Block Schema

```
▼ "blockEntity" : { 3 items
  "entityId" : string "my-entity-1"
  "entityTypeId" : string "block-type-1"
  ▼ "properties" : { 1 item
    "name" : string "World"
  }
}
```

upload schema

# Building a Block Protocol Block

## Strange Loop

Ciaran Morinan & Jude Allred, 22 September 2022

# Housekeeping

## 1. Introductions

## 2. Format

- Split into two broad sections: talk then work, talk then work
- While you're working, we will be available for help

## 3. Guidelines

- ask questions any time, often, out loud or in Slack
- feel free to ignore info broadcasts

# What we'll cover – part 1

1. Block Protocol overview
2. Principles of block building
3. Developing your block

# What we'll cover – part 2

4. Publishing your block

5. Show & tell

6. After the workshop

# 1. Block Protocol overview

# Why build a block?

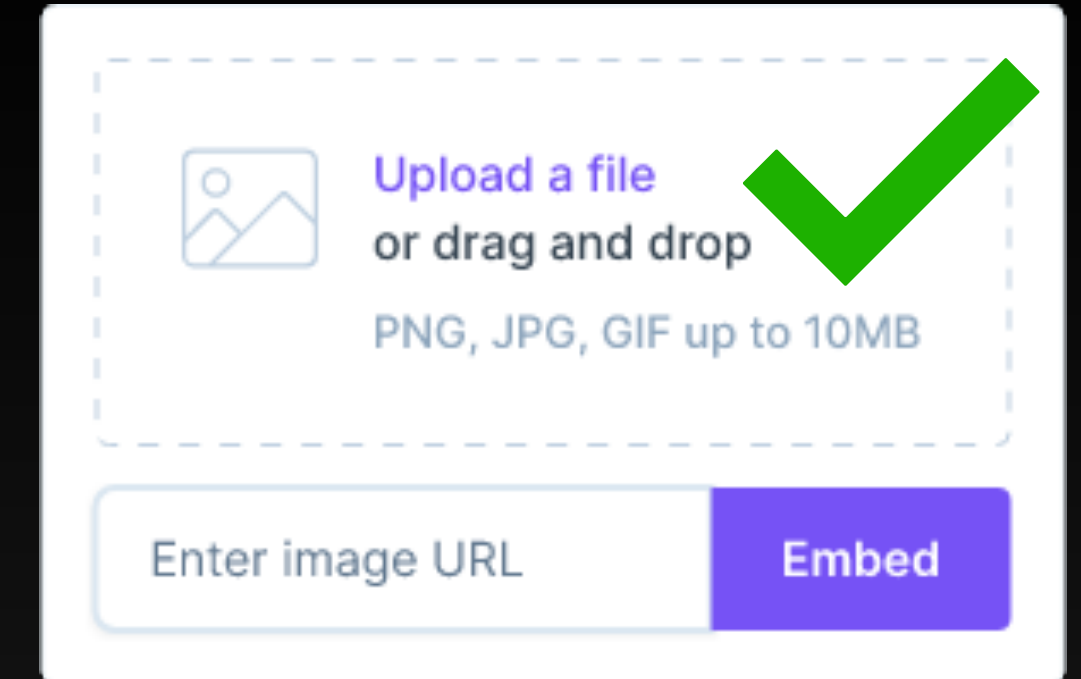
- Your block will work anywhere that implements the Protocol
- Write your block once, use it in many places
- The Block Protocol is a plugin framework for the *whole web* – not bespoke to a specific application





# What is a block?

Not Lego



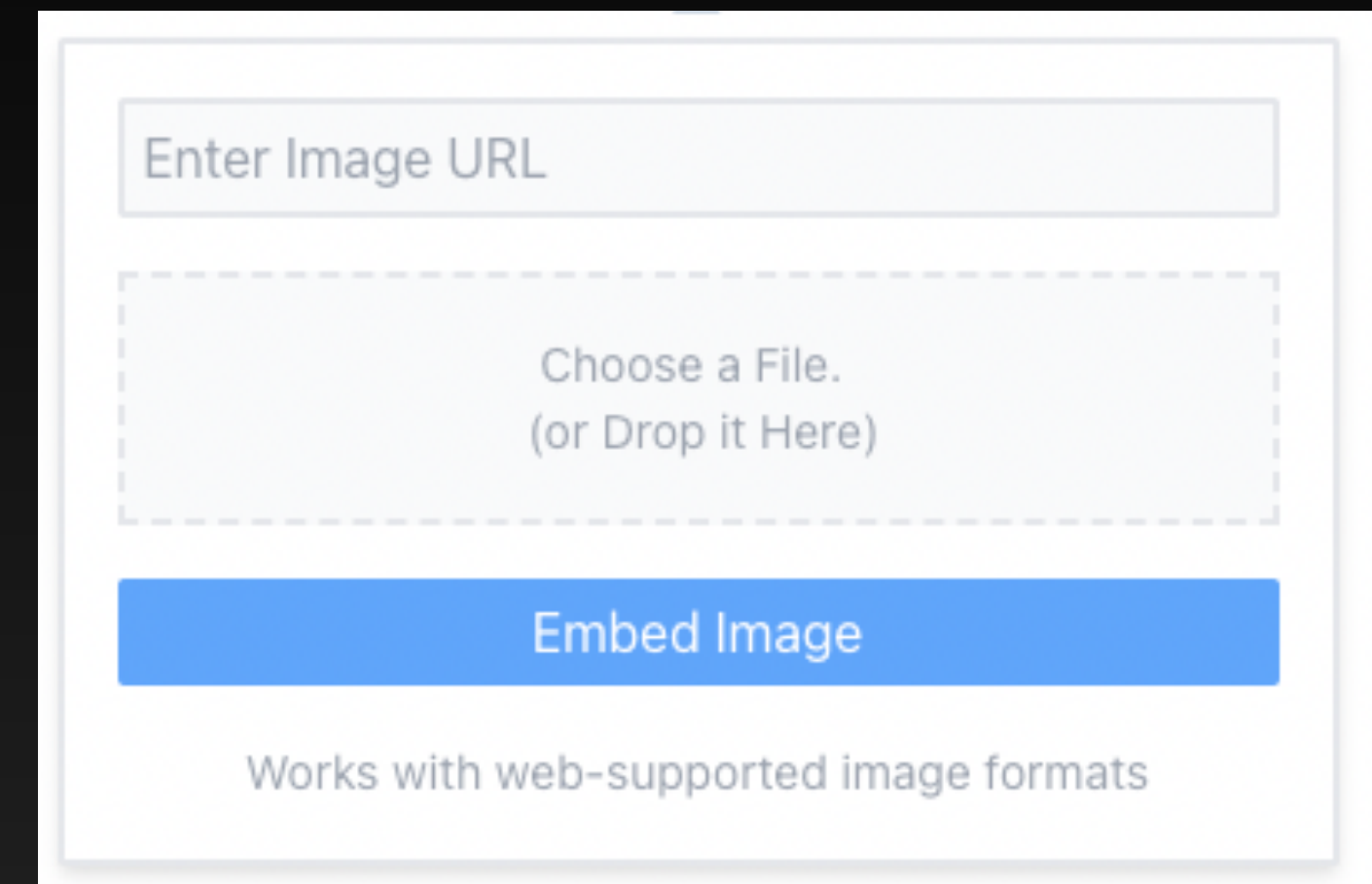
- Discrete **building blocks** for a web editor/page (or other application) - as seen in WordPress, Notion, etc
- Sometimes users can choose which block to insert, and compose a block out of pages (when building a page in a CMS)
- Sometimes a page is a set of preconfigured, inert blocks (when viewing a page someone has previously built)

# Examples

## Paragraph block

+ :: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

## Image block



Enter Image URL

Choose a File.  
(or Drop it Here)

Embed Image

Works with web-supported image formats

```
JavaScript Copy Caption  
  
function debounce(func, timeout = 300){  
  let timer;  
  return (...args) => {  
    clearTimeout(timer);  
    timer = setTimeout(() => { func.apply(this, args); }, timeout);  
  };  
}
```

## Code block

Age	CreatedAt	Email	Name	UpdatedAt	Username
38	2022-06-13T14:57:18.965Z	Michael@example.com	Michael	2022-06-13T14:57:18.965Z	michael
50	2022-06-13T14:57:18.965Z	Christopher@example.com	Christopher	2022-06-13T14:57:18.965Z	christopher
92	2022-06-13T14:57:18.965Z	Jessica@example.com	Jessica	2022-06-13T14:57:18.965Z	jessica
33	2022-06-13T14:57:18.965Z	Matthew@example.com	Matthew	2022-06-13T14:57:18.965Z	matthew

## Table block

# So we already have blocks?

- Blocks as we know them have one of two significant flaws:
  - They are written to be used in a particular application. They don't follow an open protocol and cannot be easily re-used elsewhere
  - They can't communicate data (a) back to the application or (b) with each other

# Block Protocol goal

Use any block in any application, without configuration

- **Easier to add new functionality** to and **share functionality** across applications
  - allow blocks to be easily re-used across multiple applications
  - allow applications to use blocks without knowing anything about them
- **Describe the structure of data that is available or expected**
  - dynamic validation of user input
  - easy to assess block-data compatibility / suitability
- **How** it does it
  - Defines a **contract** between **embedding application** and **block**



# The BP meets those goals by specifying:

- **Message transport:** *how* messages and requests are sent between blocks and apps
- **Message content:** *what* messages are sent between blocks and apps
  - Messages are defined under *services*
  - First such service is the **graph service**:
    - Defines the concept of 'entities' which are described by 'entity types'
    - Entities can refer to each other through 'links' (forming a knowledge graph)
    - Defines the messages via which entities, types and links can be discovered, created, and edited by blocks

# What the Block Protocol specifies (2)

- **Data structure:** how entity data is described
  - blocks must describe the shape of data they expect
  - embedder should provide schemas for every entity sent to blocks
  - Allows for programmatic compatibility checks / validation of input
  - uses *JSON Schema*

Property Name	Expected Type	Required
Products	Number	<input checked="" type="checkbox"/>
Platform	Text 0-120ch Multiple	<input checked="" type="checkbox"/>
Investors	Person Multiple	<input checked="" type="checkbox"/>
Founding date	Date DD-MM-YYYY	<input type="checkbox"/>
Estimated user base	Number 0-12,000	<input type="checkbox"/>
Monthly subscription	Number 0-100 Multiple	<input type="checkbox"/>
Product screenshots	Image Multiple	<input type="checkbox"/>

# What the Block Protocol specifies (3)

- You don't need to memorize or even be familiar with all of the protocol
- You are going to learn and use parts of it when implementing your block
- The protocol's scope is kept deliberately narrow, to maintain focus on standardizing the block <> application interface

# What it **doesn't** specify

Some common concerns that the Block Protocol leaves up to the block or application:

- **what happens to data inside the embedding application**
  - the application may store data any way it chooses, or not at all
  - data may be synced with other services/stores
- **what happens to data inside the block**
  - blocks may hold local state, if it chooses
  - blocks may communicate with external services



# Recap

1. **The Block Protocol wants to make any block usable in any application**
2. **It does this by specifying:**
  1. **a set of messages that applications and blocks can exchange to make requests and pass data between one another**
  2. **A standard data model for describing entities and the links between them**
3. **It doesn't care about what happens outside of block-application interaction**

## **2. Principles of block building**

# What will your block do?

- We're all going to build the same block: a **Portfolio Block**
- The aim of the block is to show off a **Person** and one or more **Projects** they've worked on
- We will make our block in the following steps:
  1. Display info about a person
  2. Display info about projects linked to the person
  3. Allow editing of the data (whether the person, projects, or links)

# Thinking in blocks

- A block is a discrete piece of functionality on a webpage
- should complement other blocks in a page
- should be properly encapsulated – we expect blocks to be sandboxed, but you should apply normal best practices in case they aren't (scope styling, JavaScript, etc)

# Choosing your fields

- What fields will your deal display from the Person and Projects?
- Best to start small and add more later
- A block that displays a small number of common fields can be used by many different types of entities

# What will your block look like?

- Think about how your block fits into a wider application
- Use minimal styling that works in a variety of contexts
- Think how it would look in a flow of blocks on a page

# 3. Developing your block

# First steps

- Visit <https://blockprotocol.org/workshop> for all the details we're going to talk through, and code snippets



# Let's get to work!

1. Display a Person
2. Display their linked projects
3. Going further:
  - Editing of the Person or Project
  - Respecting readonly mode

# 4. Publish your block

# Publish for the first time

## In your browser

- Sign up for an account at <https://blockprotocol.org> if you haven't already
- Create an API key and copy it: <https://blockprotocol.org/settings/api-keys>

## In your editor

- run `npx blockprotocol@latest publish` to generate a `.blockprotocol.rc` file in your project
- Replace the placeholder key in that file with your API key
- Run `npm run build` or `yarn build`
- now run `npx blockprotocol@latest publish`
- See your block on the Hub using the provided link!
  - Here's an example block page: <https://blockprotocol.org/@hash/blocks/person>

# Publish again

- Run `npm run build && npx blockprotocol@latest publish` or `yarn build && npx blockprotocol@latest publish`

## Things to try

# 5. Show and tell

# Show off your block, if you want!

- What functionality would you add next?
- What other blocks would you build to complement it?

# 6. What's next?

# For you

- **Expand** your block
- **Build** a new block
- **Please give us feedback** on the workshop and the Block Protocol!
  - Either via email: [c@hash.ai](mailto:c@hash.ai) & [j@hash.ai](mailto:j@hash.ai)
  - or form: <https://sohostrategy.typeform.com/blockfeedback>
  - We know there are a lot of potential improvements we can make to block DX – let us know what your highest priorities would be
- **Check out** our open RFCs at <https://github.com/blockprotocol/blockprotocol/discussions> and get involved in any that interests you



# For the Block Protocol

- The Block Protocol is constantly evolving – we’ve just covered using version 0.2, but there are several big updates on the way, which improve:
  - The expressiveness and power of the type system
    - not just entity types, but also individual property types, data types, link types
      - see <https://github.com/blockprotocol/blockprotocol/discussions/418>
    - The Block Protocol site will act as a hub for type hosting and discovery
  - New functionality, e.g.
    - for working with users
    - for working with rich text
    - more at [https://blockprotocol.org/docs/spec/rfcs\\_and\\_roadmap](https://blockprotocol.org/docs/spec/rfcs_and_roadmap)

# Embedding applications

- Places to embed your block
  - WordPress plugin – allow blocks to be inserted into any WordPress application
  - Production version of HASH, including use of the type system to map data to blocks
- We will email you with updates

**Thank you!**