Object types

Normally we won't be dealing with a single field, but a collection of them. For example, we may want to create an address, containing a street address, the city and a postal code. To specify such an object in JSON schema we use type: "object", and then we specify a map from the names of the properties to their specifications.

Each property is also a JSON schema, so we can specify these exactly as we have already demonstrated for basic types.

```
{
  type: “object”,
  properties: {
    streetAddress {
      type: "string",
      title: "Street address"
    },
    city: {
      type: "string",
      title: "City"
    },
    postalCode: {
      type: "string",
      title: "Postcode"
    }
  }
}
```

In this case our form might render as:

Nested objects

Because the properties are also JSON schemas this is a recursive data structure, allowing the specification of nested object structures. For example, we might have a user profile with a nested address object:

```
{
  fullname: “...”
  address: {
    streetAddress: “...”,
    city: “...”
    postalCode: “...”
  }
}
```

This would be represented in JSON schema as:

```
{
  type: "object",
  properties: {
    fullname: {
      type: "string",
      title: "Full name"
    },
    address: {
      type: "object",
      title: "Address details",
      properties: {
        streetAddress {
          type: "string",
          title: "Street address"
        },
        city: {
          type: "string",
          title: "City"
        },
        postalCode: {
          type: "string",
          title: "Postcode"
        }
      }
    }
  }
}
```

An example UI may render this as:

Full name

Address details

Street address

City

Postcode